aws

デベロッパーガイド

Amazon Simple Email Service

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Simple Email Service: デベロッパーガイド

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは、Amazon のものではない製品またはサービスにも関連して、お客様に混乱を招いたり Amazon の信用を傷つけたり失わせたりするいかなる形においても使用することはできません。Amazon が所有しない商標はすべてそれぞれの所有者に所属します。所有者は必ずしも Amazon との提携や関連があるわけではありません。また、Amazon の支援を受けているとはかぎりません。

Table of Contents

Amazon SES とは

Amazon Simple Email Service (SES) は、独自の E メールアドレスとドメインを使用して E メールを送受信するための簡単で費用対効果の高い方法を提供する E メールプラットフォームです。

例えば、特価販売などのマーケティング E メールや、注文確認などの取引 E メール、ニュースレターなどのその他のタイプの通信文の送信に使用できます。Amazon を使用してメールSESを受信すると、E メール自動応答者、E メールのサブスクリプション解除システム、受信 E メールからカスタマーサポートチケットを生成するアプリケーションなどのソフトウェアソリューションを開発できます。

Amazon に関連するトピックの詳細についてはSES、AWS メッセージングとターゲット設定のブログを参照してください。

利点

大規模な E メールソリューションを構築することは、ビジネスにとってしばしば複雑で高コストな課題となります。E メールサーバーの管理、ネットワーク構成、IP アドレスの評価といったインフラストラクチャの課題に対処する必要があります。さらに、多くのサードパーティの E メールソリューションでは、契約、料金交渉、およびかなりの初期コストが必要となります。Amazon はこれらの課題SESを排除し、Amazon.com が独自の大規模な顧客ベースに対応するために構築した長年の経験と高度な E メールインフラストラクチャを活用できます。

関連サービス

Amazon は、他の AWS 製品とシームレスにSES統合します。例えば、以下のことが可能です。

- 任意のアプリケーションに E メール送信機能を追加します。
- Amazon から E メールを送信するには、 EC2 を使用するかAWS SDK、Amazon SESSMTPインターフェイスを使用するか、Amazon SES APIに直接呼び出します。
- AWS Elastic Beanstalk を使用して、Amazon を使用して顧客にニュースレターSESを送信するプログラムなどの E メール対応アプリケーションを作成します。
- Amazon Simple Notification Service (Amazon SNS) をセットアップして、バウンスした E メール、苦情が発生した E メール、受信者のメールサーバーに正常に配信された E メールを通知します。Amazon を使用して E メールSESを受信すると、E メールコンテンツを Amazon SNSトピックに発行できます。

- を使用して Easy AWS Management Console をセットアップします。これはDKIM、E メールを認証する方法です。どのDNSプロバイダーDKIMでも Easy を使用できますが、Route 53 でドメインを管理する場合は特に簡単にセットアップできます。
- E メール送信に対するユーザーアクセスを管理するには、AWS Identity and Access Management (IAM) を使用します。
- Amazon Simple Storage Service (Amazon S3) に受信する E メールを保管します。
- AWS Lambda 関数をトリガーすることで、受信する E メールに対してアクションを実行します。
- AWS Key Management Service (AWS KMS)を使用して、Amazon S3 バケットで受信する E メールをオプションで暗号化します。
- AWS CloudTrail コンソールまたは Amazon を使用して行った Amazon SESAPIコールをログに記録するにはSES、 を使用しますAPI。
- E メール送信イベントを Amazon CloudWatch または Amazon Data Firehose に発行します。E メール送信イベントを Firehose に発行する場合は、Amazon Redshift、Amazon OpenSearch Service、または Amazon S3 でアクセスできます。

料金

Amazon ではSES、送受信された E メールの量に基づいてお支払いいただきます。詳細については、「Amazon のSES料金」を参照してください。

リージョンと Amazon SES

SES は、世界中のいくつかの AWS リージョンで利用できます。AWS は各リージョンで、複数のアベイラビリティーゾーンを管理しています。これらのアベイラビリティーゾーンは物理的に相互に分離されていますが、低レイテンシーで高スループットの冗長性に優れたプライベートネットワーク接続で統合されています。これらのアベイラビリティーゾーンにより、非常に高いレベルの可用性と冗長性を提供しながら、レイテンシーを最小限に抑えています。

すべてのリージョンの SES エンドポイントの一覧については、「AWS 全般のリファレンス」の「Amazon Simple Email Service のエンドポイントとクォータ」を参照してください。各リージョンで利用できるアベイラビリティーゾーンの数の詳細については、「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。

このセクションには、複数の AWS リージョン リージョンで SES を使用する場合に把握しておくべき情報が記載されています。必要事項は以下のとおりです。

- SES のリージョンとエンドポイント
- サンドボックスの削除と送信制限の引き上げ
- E メールアドレスおよびドメインの検証
- Easy DKIM
- アカウントレベルのサプレッションリスト
- フィードバック通知
- SMTP 認証情報
- 送信承認
- カスタム MAIL FROM ドメインに使用されるフィードバックエンドポイント
- E メールの受信
- (MX) レコードの設定

AWS リージョンの一般的な情報については、「AWS 全般のリファレンス」の「AWS サービスエンドポイント」を参照してください。

SES のリージョンとエンドポイント

SES を使用して E メールを送信する場合、SES API または SMTP インターフェイス用のエンドポイントを提供する URL に接続します。AWS 全般のリファレンス には、SES を介してメールを送受信するために使用するエンドポイントの完全なリストが記載されています。詳細については、「AWS 全般のリファレンス」の「Amazon Simple Email Service のエンドポイントとクォータ」を参照してください。具体的なセクションは、以下のとおりです。

- API エンドポイント – SES を介して E メールを送信する際は、この表に記載されている URL を使用して SES API に HTTPS リクエストを送信できます。
- SMTP エンドポイント – SMTP インターフェイスを使用する際は、この表に記載されている URL を使用して、E メールを送信できます。
- E メール受信エンドポイント – ドメインに送信される E メールを受信するように SES を設定している場合は、ドメインの DNS 設定でメールエクスチェンジャー (MX) レコードを設定する際に、このテーブルに記載されているインバウンド SMTP エンドポイント URL を使用できます。

> **Note**
>
> インバウンド SMTP URL は、IMAP サーバーアドレスではありません。つまり、Outlook などのアプリケーションを使用した E メールの受信に使用することはできません。受信

メール用の IMAP サーバーを提供するサービスについては、Amazon WorkMailを参照してください。

サンドボックスの削除と送信制限の引き上げ

アカウントのサンドボックスのステータスは、AWS リージョン リージョン間で異なる場合があります。つまり、アカウントが米国西部 (オレゴン) リージョンのサンドボックスから削除されても、米国東部 (バージニア北部) リージョンのサンドボックスからも削除しない限り、米国東部のリージョンのサンドボックスに維持される場合があります。

送信制限も、AWS リージョン リージョンごとに異なる場合があります。例えば、欧州 (アイルランド) リージョンではアカウントが毎秒 10 件のメッセージを送信できるとしても、別のリージョンでは送信できるメッセージ数が上下する場合があります。

サンドボックスからアカウントを削除するリクエストを送信する場合、またはアカウントの送信クォータを引き上げるリクエストを送信する場合、リクエストの適用対象となるすべての AWS リージョンを選択していることを確認する必要があります。単一のサポートセンターのケースで複数のリクエストを送信できます。

E メールアドレスおよびドメインの検証

SES を使用して E メールを送信する前に、E メールの送信元となるアドレスまたはドメインを所有していることを検証する必要があります。E メールアドレスおよびドメインの検証ステータスも、AWS リージョン リージョン間で異なります。例えば、米国西部 (オレゴン) リージョンでドメインを検証しても、米国東部 (バージニア北部) リージョンでドメインの検証プロセスをもう一度完了するまで、そのドメインを米国東部 (バージニア北部) リージョンでの E メールの送信に使用することはできません。E メールアドレスとドメインの検証については、「Amazon SES の検証済み ID」を参照してください。

Easy DKIM

Easy DKIM を使用する各 AWS リージョンで、Easy DKIM のセットアッププロセスを実行する必要があります。つまり、各リージョンで、SES コンソールまたは SES API を使用して CNAME レコードを生成する必要があります。次に、すべての CNAME レコードを、ドメインの DNS 設定に追加する必要があります。Easy DKIM のセットアップの詳細については、Amazon DKIMで簡単 SESを参照してください。

すべての AWS リージョンがデフォルトの SES DKIM ドメインである dkim.amazonses.com を使用するとは限りません。現在利用しているリージョンで、リージョン固有の DKIM ドメインが使用されているかを確認するには、「AWS 全般のリファレンス」の「DKIM domains table」を確認してください。

アカウントレベルのサプレッションリスト

SES アカウントレベルのアカウントレベルのサプレッションリストは、現在利用している AWS リージョン内の AWS アカウントにのみ適用されます。SES API v2 またはコンソールを使用して、アカウントレベルのサプレッションリストからアドレスを個別にまたは一括で、手動で追加または削除できます。アカウントレベルのサプレッションリストの使用の詳細については、「Amazon SESアカウントレベルのサプレッションリストの使用」を参照してください。

フィードバック通知

複数の AWS リージョンでフィードバック通知をセットアップする際に注意すべき重要な点が 2 つあります。

- フィードバックを E メールで受信するか SNS を介してのみ受け取るかなど、検証済み ID 設定は、設定を行ったリージョンのみに適用されます。例えば、user@example.com を 米国西部 (オレゴン) リージョンおよび 米国東部 (バージニア北部) リージョンで検証して、バウンスした E メールは SNS 通知を介して受信する場合、SNS API または SES コンソールを使用して、user@example.com の SNS フィードバック通知を両方のリージョンで設定する必要があります。
- フィードバック転送に使用する SNS トピックは、SES を使用するのと同じリージョンにある必要があります。

フィードバック通知による送信アクティビティのモニタリングの詳細については、「Amazon SES のイベント通知の設定」を参照してください。

SMTP 認証情報

SES SMTP インターフェイスを介してメールを送信するために使用する認証情報は、各 AWS リージョンに固有のものです。SES SMTP インターフェイスを使用して複数のリージョンに E メールを送信する場合は、各リージョンで SMTP 認証情報セットを生成する必要があります。

> **Note**
>
> 2019 年 1 月 10 日以前に SMTP 認証情報を作成した場合、古いバージョンの AWS 署名を使用して SMTP 認証情報が作成されました。セキュリティのため、この日付よりも前に作成した認証情報を削除して、新しい認証情報に置き換える必要があります。古い認証情報は、IAM コンソールを使用して削除できます。

カスタム MAIL FROM ドメインに使用されるフィードバックエンドポイント

カスタム MAIL FROM ドメインを使用する場合、SES では、ドメインが E メールプロバイダーから送信されるバウンス通知と苦情通知を受信できるように、MX レコードを発行する必要もあります。バウンス通知と苦情通知はリージョン固有のフィードバックエンドポイントに送信されるため、異なる AWS リージョンの検証済み ID に同じカスタム MAIL FROM ドメインを使用できます。

カスタム MAIL FROM ドメインを設定すると、SES はカスタム MAIL FROM が設定されているリージョンの適切なフィードバックエンドポイントを自動的に指定します。このエンドポイントは、ドメインの DNS 設定に発行 (追加) できるように、MX レコードの値フィールドに提供されています。

カスタム MAIL FROM のセットアッププロセスについては、「カスタムMAILFROMドメインの使用」を参照してください。リファレンスとして、SES がさまざまな AWS リージョンに使用するフィードバックエンドポイントは、「AWS 全般のリファレンス」の「Feedback endpoints」表に記載されています。

送信承認

代理の送信者は、ID 所有者の ID が検証済みの AWS リージョンからのみ、E メールを送信できます。代理送信者にアクセス許可を付与する承認の送信ポリシーは、そのリージョンのアイデンティティにアタッチする必要があります。送信承認の詳細については、「Amazon SES での送信承認の使用」を参照してください。

E メールの受信

Amazon S3 バケットを除き、SES で E メールの受信に使用するすべての AWS リソースは、SES エンドポイントと同じ AWS リージョン リージョンに配置されている必要があります。例えば、米国西部(オレゴン) リージョンで SES を使用する場合、使用する SNS トピック、KMS キー、Lambda 関数も米国西部(オレゴン) リージョン内に配置されている必要があります。同様に、

あるリージョン内で SES を使用して E メールを受信するには、そのリージョン内にアクティブな受信ルールセットを作成する必要があります。E メール受信の概念とセットアッププロセスについては、「Amazon SES を使用した E メールの受信」で説明されています。

「AWS 全般のリファレンス」の「E メール受信エンドポイント」表には、SES が E メール受信をサポートするすべての AWS リージョン の E メール受信エンドポイントが一覧表示されています。

Amazon のサービスクォータ SES

以下のセクションでは、Amazon SESリソースとオペレーションに適用されるクォータを一覧表示して説明します。クォータによっては、引き上げることができないものもあります。クォータの引き上げをリクエストできるかどうかを判断するには、[Adjustable] (増加の対象) 列を参照してください。

> **Note**
>
> SES クォータは AWS リージョン 、 で使用する ごとに設定されます AWS アカウント。

E メール送信クォータ

以下のクォータは、SES 経由の E メール送信に適用されます。

送信クォータ

クォータは、メッセージ数ではなく、受信者の数に基づきます。

リソース	デフォルトのクォータ	調整可能
24 時間ごとに送信可能な E メールの数	アカウントがサンドボックスにある場合は、24 時間あたり最大 200 通の E メールを送信できます。 アカウントがサンドボックスの外にある場合、この数字は具体的なユースケースによって異なります。	可能

リソース	デフォルトのクォータ	調整可能
1 秒あたりに送信できる E メールの数 (送信レート)	アカウントがサンドボックスにある場合、1 秒あたり 1 通の E メールを送信できます。 アカウントがサンドボックスの外にある場合、このレートは具体的なユースケースによって異なります。	可能

メッセージのクォータ

リソース	デフォルトのクォータ	調整可能
SES v1 APIの使用 - 最大メッセージサイズ (添付ファイルを含む)	メッセージあたり 10 MB (base64 エンコーディング後)	いいえ (メッセージサイズが 10MB を超えるワークロードの場合は、SESv2 APIへの移行を検討してください)。
SES v2 APIまたは の使用 SMTP - 最大メッセージサイズ (添付ファイルを含む)	メッセージあたり 40 MB (base64 エンコーディング後)	いいえ

> **Note**
>
> 10 MB を超えるメッセージは帯域幅スロットリングの対象となり、送信レートによっては最低 40 MB/s に調整される場合があります。例えば、40 MB のメッセージを 1 秒あたり 1 メッセージ、または 20 MB のメッセージを 1 秒間に 2 つ送信できます。

送信者と受信者のクォータ

リソース	デフォルトのクォータ	調整可能
メッセージあたりの受取人の最大数	メッセージあたり 50 人の受取人。 **Note** 受信者とは、「To」、「CC」、または BCC「」のアドレスです。	受信者の制限は調整できません。以下のメモを読んでから、 AWS アカウントマネージャーに連絡してこの機能をリクエストしてください。
検証できるアイデンティティの最大数	あたり 10,000 ID AWS リージョン。 **Note** ID とは、SES 経由で E メールを送信するために使用するドメインまたはE メールアドレスです。	AWS アカウントマネージャーに連絡して、ユースケースを相談してください。
専用 IP プールの最大数 (マネージド IP プールと標準 IP プールの両方を含む)	50	いいえ

Note

メッセージごとの受信者制限の引き上げをリクエストする前に、このブログを読み、メッセージごとの受信者数 50 人というデフォルトの制限を使用したり、個々の受信者にメッセージを送信しても、ユースケースに対応できない理由を詳しく説明できるようにしておいてください。メッセージ送信先に複数の受信者を定義すると、オブザーバビリティが低下

し、配信性能も低下する可能性があるため、ユースケースで特に必要とされない限り使用すべきではありません。

イベント発行に関連するクォータ

リソース	デフォルトのクォータ	調整可能
設定セットの最大数	10,000	いいえ
設定セット名の最大長	設定名には、最大で 64 文字までの英数字を含めることができます。この名前には、ハイフン (-) とアンダースコア (_) も使用できます。名前にスペース、アクセント文字、またはその他の特殊文字を含めることはできません。	いいえ
設定セットあたりのイベント送信先の最大数	10	いいえ
CloudWatch イベント送信先あたりのディメンションの最大数	10	いいえ

E メールテンプレートのクォータ

リソース	デフォルトのクォータ	調整可能
各 の E メールテンプレートの最大数 AWS リージョン	20,000	いいえ
最大テンプレートサイズ	500 KB	いいえ
各テンプレートの置換値の最大数	無制限	該当なし

リソース	デフォルトのクォータ	調整可能
テンプレート化された各 E メールの受信者の最大数	送信先 50 件。送信先は、「To」、「CC」、または BCC「」行の任意の E メールアドレスです。 Note への 1 回の呼び出しで連絡できる送信先の数は、アカウントの最大送信レートによって制限APIされる場合があります。	いいえ

E メール受信のクォータ

次の表に、SES による E メールの受信に関連するクォータを示します。

リソース	デフォルトのクォータ	調整可能
受信ルールセットあたりのルールの最大数	200	いいえ
受信ルールあたりのアクションの最大数	10	いいえ
受信ルールあたりの受取人の最大数	500	いいえ
あたりの受信ルールセットの最大数 AWS アカウント	40	いいえ
あたりの IP アドレスフィルターの最大数 AWS アカウント	100	いいえ

リソース	デフォルトのクォータ	調整可能
Amazon S3 バケットに保存できる E メールの最大サイズ (ヘッダーを含む)	40 MB	いいえ
Amazon SNS通知を使用して発行できる E メールの最大サイズ (ヘッダーを含む)	150 KB	いいえ
Amazon SNS通知を使用して発行できる E メールヘッダーの最大サイズ	10 KB	いいえ
AWS Lambda 機能を使用して発行できる E メールヘッダーの最大サイズ	50 KB	いいえ

Mail Manager のクォータ

Mail Manager に関連するクォータは、次の表のとおりです。

リソース	デフォルトのクォータ	調整可能
オープンイングレスエンドポイントの最大数	10	いいえ
承認済みのイングレスエンドポイントの最大数	50	いいえ
メッセージあたりの受信者の最大数	100	いいえ
E メールの最大サイズ (ヘッダーを含む)	40 MB	いいえ
トラフィックポリシーステートメントの最大数	20	いいえ

リソース	デフォルトのクォータ	調整可能
トラフィックポリシーステートメント条件の最大数	10	いいえ
リージョンあたりのトラフィックポリシーの最大数	100	いいえ
SMTP リレーの最大数	100	いいえ
ルールセットの最高数	40	いいえ
メッセージあたりのルールの実行最大数	200	いいえ
ルールあたりの条件の最大数	10	いいえ
ルールあたりのアクションの最大数	10	いいえ
ルールセットあたりのリレーアクションまたは送信アクションの最大数	10	いいえ
アクティブなアーカイブの最大数	10	いいえ
アーカイブ検索結果の最大数	1,000	いいえ
エクスポートされたアーカイブ検索結果の最大数	250000	いいえ
並行実行される検索リクエストの最大数	1	いいえ
並行実行されるエクスポートリクエストの最大数	1	いいえ
週あたりのアーカイブの保持変更の最大数	1	いいえ

一般的なクォータ

次の表は、SES 経由の E メールの送受信に適用されるクォータの一覧です。

SES API 送信クォータ

リソース	デフォルトのクォータ	調整可能
Amazon SESAPIアクションを呼び出すことができるレート	すべてのアクション（SendEmail 、SendRawEmail 、および SendTemplatedEmail を除く）は、1 秒あたり 1 つのリクエストに調整されます。	いいえ
MIME パート	500	いいえ

Amazon SES 認証情報の種類

Amazon Simple Email Service (Amazon SES) と対話するには、セキュリティ認証情報を使用して、自分の身元と、Amazon SES と対話するためのアクセス許可があることを確認します。認証情報にはさまざまな種類があり、何をするかによって使用する認証情報は異なります。たとえば、Amazon SES API を使用して E メールを送信する場合は AWS アクセスキーを使用し、Amazon SES SMTP インターフェイスを使用して E メールを送信する場合 は、SMTP 認証情報を使用します。

次の表に、目的に応じて Amazon SES で使用する可能性のある認証情報タイプを示します。

アクセス対象	使用する認証情報	認証情報の構成内容	認証情報の取得方法
Amazon SES API (Amazon SES API には、直接アクセスするか、AWS SDK、AWS Command Line	AWS アクセスキー	アクセスキー ID とシークレットアクセスキー	「AWS 全般のリファレンス」の「アクセスキー」を参照してください。

アクセス対象	使用する認証情報	認証情報の構成内容	認証情報の取得方法
Interface 、または AWS Tools for Windows PowerShell 経由で間接的にアクセスできます。)			Note セキュリティのベストプラクティスのため、AWS アカウントアクセスキーではなく AWS Identity and Access Management (IAM) ユーザーのアクセスキーを使用します。AWS アカウントの認証情報によってすべての AWS リソースへのフルアクセス権を付与するため、この認証情報は安全な場所に記憶しておき、AWS の日々の操作には IAM ユーザーの認証情報を使用してください。詳細については、「AWS 全般のリファレンス」の「ルートアカウント認証情報と IAM ユーザー認証情報」を参照してください。

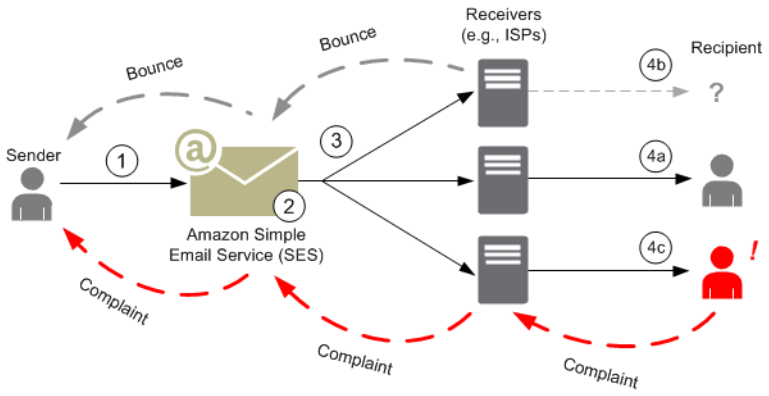
アクセス対象	使用する認証情報	認証情報の構成内容	認証情報の取得方法
Amazon SES SMTP インターフェイス	SMTP 認証情報	ユーザー名とパスワード	「Amazon SES SMTP 認証情報を取得」を参照してください **Note** Amazon SES SMTP 認証情報は、AWS アクセスキーと IAM ユーザーアクセスキーとは異なります。Amazon SES SMTP 認証情報は、実際には IAM 認証情報の一種です。IAM ユーザーは、Amazon SES SMTP 認証情報を作成できますが、ルートアカウント所有者は IAM ユーザーのポリシーによってユーザーに IAM アクション「iam:ListUsers」、「iam:CreateUser」、「iam:CreateAccessKey」、および「iam:PutUserPolicy」へのアクセス許可が付与されることを確認する必要があります。

アクセス対象	使用する認証情報	認証情報の構成内容	認証情報の取得方法
Amazon SES コンソール	IAM ユーザー名とパスワード または E メールアドレスとパスワード	IAM ユーザー名とパスワード または E メールアドレスとパスワード	「AWS 全般のリファレンス」の「IAM ユーザー名とパスワード」および「E メールアドレスとパスワード」を参照してください。 **Note** セキュリティのベストプラクティスとして、E メールアドレスとパスワードではなく、IAM ユーザー名とパスワードを使用してください。E メールアドレスとパスワードの組み合わせは AWS アカウント 用ですので、AWS の日々の操作に使用するのではなく、安全な場所に記録しておく必要があります。詳細については、「AWS 全般のリファレンス」の「ルートアカウント認証情報と IAM ユーザー認証情報」を参照してください。

さまざまなタイプの AWS セキュリティ認証情報 (Amazon SES でのみ使用される SMTP 認証情報は除く) の詳細については、「AWS 全般のリファレンス」の「AWS セキュリティ認証情報」を参照してください。

Amazon SES における E メール送信の仕組み

このトピックでは、SES で E メールを送信したときに実行される処理と、E メール送信後に想定されるさまざまな結果について説明します。次の図は、送信プロセスの概要を示しています。



1. クライアントアプリケーションは E メール送信者として動作し、1 人以上の受信者に E メールを送信するように SES にリクエストします。
2. リクエストが有効な場合、SES は E メールを受け入れます。
3. SES により、受取人の受信者宛てにインターネット経由でメッセージが送信されます。メッセージは SES に渡されると、通常は数ミリ秒内に発生する最初の配信の試行ですぐに送信されるのが一般的です。
4. この時点で、さまざまな可能性があります。以下に例を示します。

    a. ISP が、受取人の受信箱にメッセージを正常に配信します。

    b. 受信者の E メールアドレスが存在しないため、ISP が SES にバウンス通知を送信します。その後、SES は送信者に通知を転送します。

    c. 受取人がメッセージを受信したものの、そのメッセージをスパムと見なして、ISP に苦情を登録します。その ISP に対して SES でフィードバックループがセットアップされている場合は、苦情が SES に送信され、そこから送信者に転送されます。

以下のセクションでは、送信者が E メールリクエストを SES に送信した後と、SES が E メールメッセージを受信者に送信した後の想定される結果について個々に説明します。

送信者が SES に E メールリクエストを送信した後

送信者が SES に E メール送信のリクエストを送ると、呼び出しが成功または失敗します。以下のセクションでは、それぞれのケースで発生する状況を説明します。

リクエスト送信の成功

SES へのリクエストが成功すると、SES は成功のレスポンスを送信者に返します。このメッセージには、リクエストを一意に識別する文字列であるメッセージ ID が含まれます。メッセージ ID を使用して、送信した E メールを特定したり、送信中に発生した問題を追跡したりできます (ID と、E メールを受け入れたときに SES が渡す SES メッセージ ID 間で、独自のマッピングを保存する必要があります)。SES では、リクエストパラメータに基づいて E メールメッセージをアセンブルし、メッセージをスキャンして疑わしいコンテンツやウイルス感染を調べます。その後に、SMTP (Simple Mail Transfer Protocol) を使用してインターネット経由でメッセージを送信します。通常、メッセージは即座に送信されます。一般的に最初の配信の試行は数ミリ秒以内に発生します。

> **Note**
>
> SES は、送信者のリクエストを受け入れ、そのメッセージにウイルスが含まれていると判断した場合、メッセージの処理を停止し、そのメッセージを受信者のメールサーバーに配信しようとしません。

リクエスト送信の失敗

送信者が SES に送った E メール送信リクエストが失敗した場合、SES はエラーを含むレスポンスを送信者に返して E メールを削除します。リクエストの失敗にはいくつかの理由が考えられます。リクエストが適切にフォーマットされていない場合や、E メールアドレスが送信者によって検証されていない場合などがあります。

リクエストが失敗したかどうかを判断する方法は、SES を呼び出す方法によって異なります。次の例は、エラーと例外がどのように返されるかを示しています。

- クエリ (HTTPS) API (`SendEmail` または `SendRawEmail`) で SES を呼び出している場合、そのアクションでエラーが返されます。詳細については、「Amazon Simple Email Service API リファレンス」を参照してください。
- 例外を使用するプログラミング言語の AWS SDK を使用する場合は、SES の呼び出しによって、MessageRejectedException がスローされます。(SDK によってはこの例外の名前が多少異なる場合があります)。

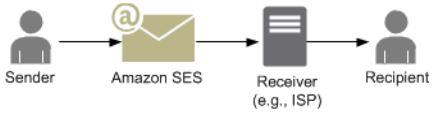
- SMTP インターフェイスを使用する場合、送信者は SMTP レスポンスコードを受け取りますが、エラーがどのように示されるかは送信者のクライアントによって異なります。一部のクライアントではエラーコードが表示されますが、表示されないクライアントもあります。

SES で E メールを送信するときに発生する可能性があるエラーについては、「Amazon SES の E メール送信エラー」を参照してください。

Amazon SES から E メールを送信した後

送信者が SES に送ったリクエストが成功した場合、SES が E メールを送信し、次の結果のいずれかが発生します。

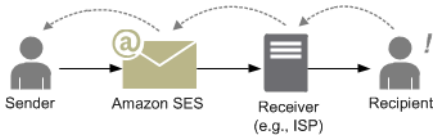
- 配信が成功し、受信者が E メールを拒否しない場合 – E メールは ISP により受け入れられ、ISP が E メールを受信者に配信します。次の図は、正常な配信を示しています。



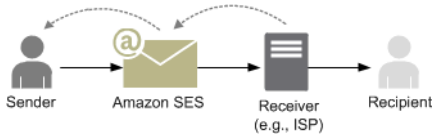
- ハードバウンス - E メールは、永続的な状態により ISP から拒否されるか、E メールアドレスが SES サプレッションリストに含まれているために SES から拒否されます。SES のいずれかの顧客で最近ハードバウンスの原因となった E メールアドレスは、SES のサプレッションリストに含まれます。ISP のハードバウンスは、受取人のアドレスが無効であるために発生する場合があります。ハードバウンスが発生すると ISP から SES に通知が送信され、さらに送信者に (送信者のセットアップに応じて) E メールまたは Amazon Simple Notification Service (Amazon SNS) で通知されます。SES は、同じ方法でサプレッションリストのバウンスを送信者に通知します。次の図は、ISP からのハードバウンスのパスを示しています。

- ソフトバウンス – ISP がリクエストを処理できないほどビジーである、受信者のメールボックスがいっぱいであるなどの一時的な状態のため、ISP が受信者に E メールを配信できません。ドメインが存在しない場合も、ソフトバウンスが発生することがあります。ISP は SES にソフトバウンス通知を返します。ドメインが存在しない場合、SES はドメインの E メールサーバーを見つけることができません。いずれの場合でも、SES は一定期間にわたって E メールの配信を再試行します。その期間内に E メールを配信できない場合、SES は E メールまたは Amazon SNS を通じてバウンス通知を送信者に送信します。再試行中に SES が E メールを受信者に配信できた場合、配信は成功です。次の図は、ソフトバウンスを示しています。この例では、SES は E メールの送信を再試行し、ISP は最終的に E メールを受信者に配信しています。

- 苦情 – E メールは ISP に受け入れられて受信者に配信されますが、受信者がそのメールを迷惑メールと見なし、E メールクライアントの [迷惑メールとしてマーク] などのボタンをクリックします。SES がその ISP に対してフィードバックループをセットアップしている場合は、苦情の通知が SES に送信され、そこから送信者に転送されます。ほとんどの ISP は苦情を申告した受信者の E メールアドレスを提供しません。したがって、SES から送信者に渡される苦情通知で示されるのは苦情を申告した可能性がある受信者のリストであり、元のメッセージの受信者全員と ISP から SES に渡された苦情の情報から推定されたものです。次の図は、苦情のパスを示しています。



- 自動応答 – E メールが ISP に受け入れられ、受信者に配信されます。ISP は、不在 (OOTO) メッセージなどの自動応答を SES に送信します。SES は、自動応答通知を送信者に転送します。次の図は、自動応答を示しています。



  SES に対応するプログラムで、自動応答が生成されるメッセージの送信を再試行しないようにしてください。

> **Tip**
>
> SES メールボックスシミュレーターを使用して、正常な配信、バウンス、苦情、OOTO をテストしたり、アドレスがサプレッションリストに含まれている場合にどうなるかをテストしたりできます。詳細については、「手動でメールボックスシミュレーターを使用する」を参照してください。

Amazon SESのE メール形式

クライアントが Amazon SES にリクエストを送ると、Amazon SES では、Internet Message Format 仕様 (RFC 5322) に準拠した E メールメッセージを構築します。以下に説明するように、E メールはヘッダー、本文、およびエンベロープで構成されます。

- ヘッダー - ルーティングの指示およびメッセージに関する情報が含まれています。例では、送信者のアドレス、受取人のアドレス、件名、および日付となっています。ヘッダーは、手紙の一番上に書かれる情報に似ていますが、ヘッダーにはその他の数多くのタイプの情報 (メッセージの形式など) を含めることができます。
- 本文 - メッセージ自体のテキストが含まれています。
- エンベロープ - SMTP セッション中に E メールクライアントとメールサーバーの間でやり取りされる実際のルーティング情報が含まれています。この E メールエンベロープ情報は、封筒に書かれる情報に似ています。E メールエンベロープのルーティング情報は、通常は E メールヘッダー内のルーティング情報と同じですが、異なる場合もあります。たとえば、ブラインドカーボンコピー (BCC) を送信した場合、エンベロープから取得される実際の受取人アドレスは、ヘッダーから取得されて受取人の E メールクライアントに表示される「To」アドレスと同じではありません。

E メールのシンプルな例を次に示します。ヘッダーの後には空の行が続き、その後に E メールの本文が続きます。エンベロープは E メール自体には含まれず、SMTP セッション中にクライアントとメールサーバーの間で通信されるため、ここには示されていません。

```
Received: from abc.smtp-out.amazonses.com (123.45.67.89) by in.example.com
 (87.65.43.210); Fri, 17 Dec 2010 14:26:22
From: "Andrew" <andrew@example.com>;
To: "Bob" <bob@example.com>
Date: Fri, 17 Dec 2010 14:26:21 -0800
Subject: Hello
Message-ID: <61967230-7A45-4A9D-BEC9-87CBCF2211C9@example.com>
Accept-Language: en-US
Content-Language: en-US
Content-Type: text/plain; charset="us-ascii"
Content-Transfer-Encoding: quoted-printable
MIME-Version: 1.0

Hello, I hope you are having a good day.
```

```
-Andrew
```

以下のセクションでは、E メールヘッダーと本文について説明し、Amazon SES を使用するときに提供する必要のある情報を示します。

E メールヘッダー

E メールメッセージごとに 1 個のヘッダーがあります。ヘッダーの各行にはフィールドが含まれ、その後にコロン、その後にフィールド本文が続きます。メールクライアントでメールを読むときに、通常、E メールクライアントには次のヘッダーフィールドの値が表示されます。

- To – メッセージの受信者の E メールアドレス。
- CC – メッセージのカーボンコピーの受信者の E メールアドレス。
- From – E メールの送信元の E メールアドレス。
- Subject - メッセージトピックの概要。
- Date - E メールが送信された日時。

これ以外にも、ルーティング情報を提供し、メッセージのコンテンツを説明するヘッダーフィールドが多数あります。E メールクライアントは、そのようなフィールドを、通常、ユーザーに表示しません。Amazon SES が受け入れるヘッダーフィールドの完全なリストについては、「Amazon SES ヘッダーフィールド」を参照してください。Amazon SES を使用する場合は、特に「From」、「Reply-To」、および「Return-Path」ヘッダーフィールドの違いを理解する必要があります。前述のように、「From」アドレスはメッセージ送信者の E メールアドレスですが、「Reply-To」および「Return-Path」は以下のとおりです。

- Reply-To - 返信が送信される E メールアドレス。デフォルトでは、返信は元の送信者の E メールアドレスに送信されます。
- Return-Path - メッセージのバウンスおよび苦情が送信される E メールアドレス。「Return-Path」は、「envelope from」、「envelope sender」、または「MAIL FROM」という名前になることもあります。

> **Note**
>
> Amazon SES を使用する場合は、常に「Return-Path」パラメータを設定して、バウンスを常時把握し、バウンスの発生時に適切なアクションを実行できるようにすることをお勧めします。

バウンスされたメッセージと受取人を簡単に照合するために、可変エンベロープリターンパス (VERP) を使用できます。VERP を使用することで、メッセージがバウンスされた場合にどの受取人からバウンスされたかを自動的に把握できるように、受取人ごとに異なる「Return-Path」を設定できます。これにより、バウンスメッセージを開いて解析する必要がなくなります。

E メール本文

E メール本文には、メッセージのテキストが含まれています。本文は、次の形式で送信できます。

- HTML – 受信者の E メールクライアントで HTML を解釈できる場合は、フォーマットされたテキストおよびハイパーリンクを本文に含めることができます。
- プレーンテキスト – 受信者の E メールクライアントがテキストベースの場合は、表示不可能な文字を本文に含めることはできません。
- HTML とプレーンテキスト – 1 つのメッセージで両方の形式を使用して同じコンテンツを送信すると、受信者の E メールクライアントでは機能に応じてどちらの形式で表示するかを決定します。

E メールメッセージを多数の受取人に送信する場合は、HTML とテキストの両方の形式で送信する方法が適しています。HTML 対応 E メールクライアントを使用している受取人は、メッセージに埋め込まれたハイパーリンクをクリックして表示できます。テキストベースの E メールクライアントを使用している受取人については、URL をコピーしてウェブブラウザで表示できるように、URL を記載しておく必要があります。

Amazon SES に提供する必要がある E メール情報

Amazon SES で E メールを送信する場合、指定する必要のある E メール情報は、Amazon SES をどのように呼び出すかによって異なります。最小限の情報を提供した場合は、Amazon SES ですべてのフォーマットが自動的に処理されます。また、添付ファイルの送信などやや高度な作業を実行する場合は、raw メッセージを自分自身で提供できます。以下のセクションでは、Amazon SES API、Amazon SES SMTP インターフェイス、または Amazon SES コンソールを使用して E メールを送信するときに提供する必要のある情報について説明します。

Amazon SES API

Amazon SES API を直接呼び出す場合は、`SendEmail` または `SendRawEmail` API を呼び出します。提供する必要がある情報の量は、呼び出す API によって異なります。

- `SendEmail` API では、送信元アドレス、宛先アドレス、メッセージの件名、メッセージ本文のみを提供するだけで済みます。必要に応じて、「Reply-To」アドレスを指定できます。この API を呼び出すと、Amazon SES では、E メールクライアントソフトウェアによる表示に最適化さ

れた、適切にフォーマット済みのマルチパート多目的インターネットメール拡張 (MIME) メールメッセージが自動的にアセンブルされます。詳細については、「Amazon SES API を使用してフォーマット済み E メールを送信する」を参照してください。

- `SendRawEmail` API には、ヘッダー、MIME パーツ、およびコンテンツの種類を指定し、独自の raw E メールメッセージをフォーマットして送信できる柔軟性があります。`SendRawEmail` は、通常は上級ユーザーに利用されています。メッセージの本文とすべてのヘッダーフィールドは、Internet Message Format 仕様 (RFC 5322) に準拠した形式で入力する必要があります。詳細については、「Amazon SES API v2 を使用した raw E メールの送信」を参照してください。

AWS SDK を使用して Amazon SES API を呼び出す場合は、対応する関数 (たとえば、Java の場合は `SendEmail` や `SendRawEmail`) に上記の情報を指定します。

Amazon SES API を使用した E メール送信の詳細については、「Amazon SES API を使用して E メールを送信する」を参照してください。

Amazon SES SMTP インターフェイス

SMTP インターフェイスで Amazon SES にアクセスする場合、SMTP クライアントアプリケーションでメッセージがアセンブルされるため、提供する必要のある情報は使用するアプリケーションによって異なります。少なくとも、クライアントとサーバー間の SMTP 交換には、送信元アドレス、宛先アドレス、およびメッセージデータが必要です。

Amazon SES SMTP インターフェイスを使用した E メール送信の詳細については、「Amazon SES SMTP インターフェイスを使用して E メールを送信」を参照してください。

Amazon SES コンソール

Amazon SES コンソールを使用してメールを送信する場合、提供する必要のある情報の量は、フォーマットされたメールまたは raw Eメールのどちらを送信するかによって異なります。

- フォーマットされた E メールを送信するには、送信元アドレス、宛先アドレス、メッセージの件名、メッセージ本文を指定する必要があります。Amazon SES では、E メールクライアントソフトウェアによる表示に最適化された、適切にフォーマット済みのマルチパート MIME メールメッセージが自動的にアセンブルされます。また、Reply-To および Return-Path フィールドを指定することもできます。
- raw E メールを送信する場合、送信元アドレス、宛先アドレス、およびメッセージコンテンツを指定します。このコンテンツには、Internet Message Format 仕様 (RFC 5322) に準拠した、メッセージ本文およびすべてのヘッダーフィールドを含める必要があります。

Amazon SES における E メール配信可能性の概要

送信者が望むのは、受取人が E メールを読み、有益であると見なし、スパムのラベルを付けないことです。つまり、E メールの配信性能 (メールが受信者の受信箱に到達する割合) を最大限に高める必要があります。このトピックでは、Amazon SES を使用する場合に理解を深めておく必要のある、E メールの配信性能の概念について説明します。

メールの配信可能性を最大限に高めるには、E メール配信の問題を理解して、問題を回避するステップを積極的に実施し、送信メールのステータス情報を継続的に入手することが必要です。さらに、配信が成功する確立を高めるために、必要に応じてメール送信プログラムを改善する必要があります。以下のセクションでは、これらのステップの基本となる概念について説明し、このプロセスの中で Amazon SES がどのように役立つかを示します。

E メール配信問題の理解

ほとんどの場合、メッセージは対象の受取人に正常に配信されます。ただし、配信が失敗する場合や、送信されたメールの受け取りを受取人が拒否する場合があります。以下のセクションで説明するバウンス、苦情、およびサプレッションリストは、これらの配信問題に関連しています。

バウンス

受信者のレシーバー (E メールプロバイダーなど) がメッセージを受信者に配信できなかった場合、レシーバーはそのメッセージを Amazon SES にバウンスします。Amazon SES は、システムのセットアップ内容に応じて、バウンスされた E メールに関して、E メールまたは Amazon Simple Notification Service (Amazon SNS) を介して通知します。詳細については、「Amazon SES のイベント通知の設定」を参照してください。

以下に示すように、ハードバウンスとソフトバウンスがあります。

- ハードバウンス - 永続的な E メール配信の障害。たとえば、メールボックスは存在しません。Amazon SES は、DNS ルックアップの失敗以外では、ハードバウンスを再試行しません。ハードバウンスの原因となっている E メールアドレスに対して配信の試行を繰り返さないことを強くお勧めします。
- ソフトバウンス - 一時的な E メール配信の障害。たとえば、メールボックスがいっぱいである、接続が多すぎる (スロットリングとも呼ばれる)、または接続がタイムアウトになった場合です。Amazon SES はソフトバウンスを複数回再試行します。それでも E メールを配信できない場合、Amazon SES は再試行を停止します。

Amazon SES は、以後再試行されないハードバウンスおよびソフトバウンスについて通知します。ただし、Amazon SES コンソールまたは `GetSendStatistics` API を使用して取得されるバウンス率とバウンスメトリクスに反映されるのはハードバウンスのみです。

バウンスには同期または非同期があります。同期バウンスは、送信者と受取人の E メールサーバーがアクティブに通信している間に発生します。非同期バウンスは、最初に受取人が E メールメッセージの配信を承諾した後で、受取人への配信に失敗した場合に発生します。

苦情

ほどんどの E メールクライアントプログラムには、メッセージをスパムフォルダに移動して E メールプロバイダーに転送するためのボタン ([Mark as Spam (スパムとしてマーク)] など) が用意されています。また、ほとんどの E メールプロバイダーでは、ユーザーが不要な E メールメッセージを転送して E メールプロバイダーによる防止策をリクエストできる迷惑メール用アドレス (abuse@example.net など) を用意しています。どちらの場合も、受取人が苦情を申し立てます。E メールプロバイダーがスパム送信者であると判断し、Amazon SES により E メールプロバイダーにフィードバックループが設定されている場合、E メールプロバイダーは Amazon SES に苦情を送信します 。Amazon SES はそのような苦情を受け取ると、スパム発信者と見なされた送信者のシステムのセットアップに応じて、E メールまたは Amazon SNS 通知でその送信者に苦情を転送します。

詳細については、「Amazon SES のイベント通知の設定」を参照してください。苦情を申し立てている E メールアドレスに対して配信の試行を繰り返さないことをお勧めします。

グローバルサプレッションリスト

Amazon SES グローバルサプレッションリストは、SES 共有 IP プール内のアドレスの評判を保護するために SES が所有および管理するもので、最近いずれかの SES 顧客に対してハードバウンスを引き起こした受信者メールアドレスが含まれています。サプレッションリストに登録されているアドレスに SES 経由で E メールを送信しようとした場合、SES の呼び出しは成功しますが、SES は、この E メールを送信せず、代わりにハードバウンスとして扱います。ハードバウンスと同じく、ブラックリストのバウンスは送信量上限やバウンス率にカウントされます。E メールアドレスは、サプレッションリストに最大 14 日まで残ります。送信しようとしている電子メールアドレスが有効であることが確認されている場合は、アドレスがアカウントレベルのサプレッションリストにリストされていないことを確認して、グローバル抑制リストを上書きできます。SES は引き続き配信を試みますが、バウンスした場合、バウンスはあなた自身の評価に影響しますが、他のユーザーは、彼ら自身のアカウントレベルのサプレッションリストを使用していない場合そのメールアドレスには送信できないため、誰もバウンスを受信しません。アカウントレベルのサプレッションリストの詳細については、「Amazon SESアカウントレベルのサプレッションリストの使用」を参照してください。

積極的な対応

インターネット経由の E メールに関する最大の問題の 1 つは、未承諾一括 E メール (スパム) です。E メールプロバイダーは、顧客にスパムが送信されないように広範な防止策を実行しています。また、Amazon SES では、E メールプロバイダーがお客様の E メールをスパムと見なす可能性を減らすための措置を講じています。Amazon SES は、検証、認証、送信クォータ、およびコンテンツフィルタリングを使用します。Amazon SES は、E メールプロバイダーとの信頼された評価も維持しており、高品質 E メールの送信を求めます。Amazon SES では、これらの作業の一部を自動的に処理します (コンテンツのフィルタリングなど)。その他の場合、ツールが提供したり (認証など)、適切な指示で助言します (クォータの送信)。以下のセクションでは、それぞれの概念の詳細情報を提供します。

検証

残念ながらスパムの発信者は、E メールヘッダーを改ざんし、元のメールアドレスを偽装して、E メールが別の送信元から送られたように見せかけることができます。E メールプロバイダーと Amazon SES の信頼関係を維持するために、Amazon SES は送信者が本人であることを確認する必要があります。したがって、送信者は送信アイデンティティを保護するために、Amazon SES 経由で送信する E メールの送信元となるすべての E メールアドレスを確認する必要があります。E メールアドレスを検証するには、Amazon SES コンソールまたは Amazon SES API を使用します。ま

た、ドメイン全体を確認することもできます。詳細については、Eメールアドレス ID の作成およびドメイン ID の作成を参照してください。

アカウントがAmazon SES サンドボックスにまだある場合は、Amazon SES のメールボックスシミュレーターから提供されているアドレスを除く、すべての受信者のアドレスも確認する必要があります。サンドボックスの外への移動については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。メールボックスシミュレーターの詳細については、「手動でメールボックスシミュレーターを使用する」を参照してください。

認証

認証は、本人であることを E メールプロバイダーに示す、もう 1 つの方法です。メールを認証するときに、自分がアカウントの所有者であり、E メールが転送中に変更されていないという証拠を提供します。場合によっては、E メールプロバイダーは、認証されない E メールの送信を拒否します。Amazon SES では、送信者ポリシーフレームワーク (SPF) とドメインキーアイデンティファイドメール (DKIM) の 2 つの認証方法がサポートされています。詳細については、「Amazon SES での ID の設定」を参照してください。

送信クォータ

E メールプロバイダーでは、E メールの数量または頻度が突然急増する現象を検出すると、その送信者がスパム発信者であることを疑い、E メールをブロックする場合があります。したがって、すべての Amazon SES アカウントには、送信クォータのセットがあります。これらのクォータは、24 時間内に送信できる E メールの数と、1 秒あたりに送信できる E メールの数を制限します。これらの送信クォータは、E メールプロバイダーとの信頼性の保護に役立ちます。

ほとんどの場合、新規ユーザーは、Amazon SES で毎日少量の E メールを送信できます。送信したメールが E メールプロバイダーに受け入れられる場合は、自動的にこのクォータが引き上げられます。送信クォータは時間の経過とともに次第に引き上げられ、より短い間隔でより多くの E メールを送信できるようになります。また、SES 送信制限の増加ケースを作成して、追加のクォータ引き上げをリクエストします。

送信クォータの詳細およびクォータを引き上げる方法については、「Amazon SES 送信制限の管理」を参照してください。

コンテンツのフィルタリング

多くの E メールプロバイダーでは、コンテンツのフィルタリングを使用して、受信 E メールがスパムであるかどうかを判断します。コンテンツフィルターは、疑わしいコンテンツを探し、E メールがスパムのプロファイルに適合する場合はその E メールをブロックします。Amazon SES で

は、コンテンツフィルタも使用します。アプリケーションから Amazon SES にリクエストを送ると、Amazon SES で自動的に E メールメッセージがアセンブルされ、メッセージヘッダーおよび本文がスキャンされて、E メールプロバイダーでスパムと見なされる可能性のあるコンテンツが含まれているかどうかが判断されます。Amazon SES で使用されているコンテンツフィルタによってメッセージにスパムの疑いが生じた場合は、Amazon SES による評価が下がります。

また、Amazon SES はすべてのメッセージをスキャンして、ウイルスが含まれていないかを確認します。メッセージにウイルスが含まれている場合、Amazon SES はそのメッセージを受信者のメールサーバーに配信しようとしません。

評価

E メール送信に関して、評価は IP アドレス、E メールアドレス、または送信ドメインがスパムの発信元ではないことを確認する測定指標であり、非常に重要です。送信者の E メールが 受信者の受信トレイに確実に配信されるように、Amazon SES は E メールプロバイダーからの高い評価を維持しています。同様に、送信者も Amazon SES からの高信頼の評価を維持する必要があります。高品質なコンテンツを送信することにより、Amazon SES との信頼関係を構築します。常に高品質なコンテンツを送信することで信頼性の評価が徐々に高くなり、Amazon SES によって送信クォータが引き上げられます。バウンスおよび苦情が増えすぎると評価が下がり、Amazon SES でのアカウントの送信クォータが引き下げられたり、Amazon SES アカウントが停止されたりする場合があります。

高い評価を維持する方法の 1 つは、システムをテストするときに、自分で作成した E メールアドレスに送信するのではなくメールボックスシミュレーターを使用することです。メールボックスシミュレーターに送信された E メールは、バウンスや苦情のメトリクスには影響しません。メールボックスシミュレーターの詳細については、「手動でメールボックスシミュレーターを使用する」を参照してください。

高品質 E メール

高品質 E メールは、受取人が価値を認めて受信を希望する E メールです。値は、受信者によって異なるものを意味し、その形式にはオファー、注文確認、領収書、ニュースレターなどがあります。E メールプロバイダーで品質が低いと見なされた E メールはブロックされるため、最終的に配信性能は送信する E メールの品質に左右されます。

継続的な情報の入手

配信が失敗しても、受取人が E メールに対して苦情を申し立てても、Amazon SES によって E メールが受取人のメールサーバーに正常に配信されても、Amazon SES の使用統計を簡単に監視できる機能と通知を使用して問題を追跡することができます。

通知

E メールがバウンスされると、E メールプロバイダーは Amazon SES に通知し、Amazon SES はユーザーに通知します。Amazon SES は、以後再試行しないハードバウンスおよびソフトバウンスについて通知します。多くのE メールプロバイダーも苦情を転送しています。Amazon SES では、主要なメールプロバイダと苦情フィードバックループを設定しているため、必要ありません。Amazon SES では、バウンス、苦情、正常な配信について 2 つの方法で通知します。アカウントを設定して Amazon SNS を介して通知する方法と、E メールで通知する方法 (バウンスおよび苦情のみ) です。詳細については、「Amazon SES のイベント通知の設定」を参照してください。

使用統計

Amazon SES の使用統計を利用して、失敗した配信を調べ、根本原因を判断して解決することができます。使用統計を表示するには、Amazon SES コンソールを使用するか、Amazon SES API を呼び出します。配信、バウンス、苦情、およびウイルス感染により拒否された E メールの数を調べたり、送信クォータを超えていないかを確認したりできます。

E メール送信プログラムの向上

多数のバウンスや苦情が発生する場合は、E メール送信戦略を再評価する必要があります。返送、苦情、および質の低い E メール送信の試行が多すぎると、悪用を構成し、終了のリスクにさらされ、AWS アカウント が停止されるおそれがあることを忘れないでください。基本的に、Amazon SES を使用して高品質 E メールを送信し、受信を希望する受取人にのみ E メールを送信する必要があります。

少なくとも 1 回の配信

Amazon SES 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、メッセージの受信時に、そのメッセージコピーをもう一度受け取る場合があります。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 SES を使用した E メール送信のベストプラクティス

お客様とのメール通信を管理する方法は、メールプログラムと呼ばれます。メールプログラムの成否を分けるいくつかの要因があります。これらの要因に対しては、当初、複雑さや戸惑いを感じること

があるかもしれません。しかし、メールの配信方法を理解し、特定のベストプラクティスに従うことで、受取人の受信トレイにメールが正常に届くようになります。

トピック

- E メールプログラムの成功のメトリクス
- 良好な送信者評価の維持

E メールプログラムの成功のメトリクス

メールプログラムの成功度は、いくつかのメトリクスを使って測定することができます。

このセクションでは、以下のメトリクスについて説明します。

- バウンス
- 苦情
- メッセージの品質

バウンス

メールが目的の受取人に届かないと、バウンスが発生します。ハードバウンスとソフトバウンスの 2 種類があります。ハードバウンスは、メールの不達の原因が、E メールアドレスが存在しないなどの永続的な問題である場合に発生します。ソフトバウンスは、メールの不達の原因が一時的な問題である場合に発生します。たとえば、受信者の受信トレイが満杯である場合や受信サーバーが一時的に使用できない場合にソフトバウンスが発生します。Amazon SES では、ソフトバウンスの対処方法として、一定期間、メールの再配信を試行します。

メールプログラムでハードバウンス数をモニタリングすること、および受信者リストからハードバウンスの原因である E メールアドレスを削除することが重要です。E メールレシーバーは、ハードバウンス率が高いことを検出すると、送信者が受信者をのよく知らないものとみなします。その結果、高いハードバウンス率によりメールメッセージの配信性能が低下する場合があります。

バウンスをなくし、送信者の評価を向上させるために、以下のガイドラインを参考にしてください。

- ハードバウンス率は 5% 未満に抑える。メールプログラムのハードバウンス数が少ないほど、ISP がメッセージを正当で重要なものと判断する確率が高まります。このレートは妥当で十分に達成可能な目標ですが、すべての ISP に採用されている統一ルールというわけではありません。
- メールリストのレンタルや購入は絶対にしないこと。レンタルや購入したリストには、大量の無効なアドレスが含まれている場合があり、ハードバウンス率が大幅に増える原因となります。さら

に、これらのリストには、スパムトラップが含まれている可能性があります。それは違法な送信者を捕まえるために限定的に使用される E メールアドレスです。メッセージがスパムタラップに陥ると、配信率と送信者の評価は決定的に損なわれる場合があります。

- リストを最新状態に保つこと。受信者に久しぶりにメールを送る場合は、他の手段 (ウェブサイトのログインアクティビティや購入履歴など) で受信者のステータスを確認します。
- お客様のステータスを確認する方法がない場合は、ウィンバックメールの送信を検討する。一般的なウィンバックメールでは、お客様からの連絡が途絶えていることをお客様に伝え、引き続きメールの受信を望む場合は確認の返信をしてもらいます。ウィンバックメールの送信後に、返信がなかったすべての受信者をリストから消去します。

バウンスを受信したら、以下のルールに従って適切に対応する必要があります。

- ハードバウンスの原因がメールアドレスである場合は、そのアドレスをリストから即座に削除します。ハードバウンスしたアドレスにメッセージを再送信しないでください。ハードバウンスの繰り返しは加算されて、最終的に受信者の ISP から悪い評価を受けることになります。
- バウンス通知の受信に使用するアドレスで E メールを受信できることを確認します。バウンスと苦情通知の設定の詳細については、「Amazon SES のイベント通知の設定」を参照してください。
- 受信メールが独自の社内サーバーではなく ISP から届いている場合は、大量のバウンス通知がスパムフォルダに入るか、完全に廃棄される可能性があります。バウンスの受信にはホステッド型の E メールアドレスを使用しないのが理想的です。それでも使用する必要がある場合は、スパムフォルダを頻繁に確認してください。また、バウンスメッセージをスパムとして報告しないでください。Amazon SES で、バウンス通知の送信先のアドレスを指定できます。
- 通常、バウンスでは配信を拒否しているメールボックスのアドレスが提供されます。ただし、受信者のアドレスを特定の E メールキャンペーンにマッピングするための詳細なデータが必要な場合は、自社の追跡システムまでたどれるように X ヘッダーに適切な値を含めます。詳細については、「Amazon SES ヘッダーフィールド」を参照してください。

苦情

メールの受信者がウェブベースの E メールクライアントで [スパムとしてマーク] (または同等) のボタンをクリックすると、苦情が発生します。これらの苦情が大量に累積すると、ISP からスパムの送信元とみなされます。これにより、配信性能と送信者の評価が低下します。いくつか、全てではありませんが、苦情が報告されたときに通知する ISP があります。これはフィードバックループとして知られています。Amazon SES は、フィードバックループを提供する ISP からの苦情を自動的に転送します。

苦情をなくし、送信者の評価を向上させるために、以下のガイドラインを参考にしてください。

- 苦情率は 0.1% 未満に抑える。メールプログラムの苦情数が少ないほど、ISP がメッセージを正当で重要なものと判断する確率が高まります。このレートは妥当で十分に達成可能な目標ですが、すべての ISP に採用されている統一ルールというわけではありません。
- マーケティングメールに関してお客様から苦情が寄せられた場合は、そのお客様に対するマーケティングメールの送信を即座に停止します。ただし、メールプログラムに他のタイプの E メール (通知メールやトランザクションメールなど) が含まれている場合、これらのタイプのメッセージは引き続き苦情元の受信者に送信することが許容されます。
- ハードバウンスと同様に、久しぶりにメールを送信する受信者のリストがある場合は、メッセージを送信する理由を確実に受信者に伝えます。受信者に送信者を思い出してもらうためと、連絡する理由を知らせるための「ようこそ」メッセージを送信することをお勧めします。

苦情を受けたら、以下のルールに従って適切に対応する必要があります。

- 苦情通知の受信に使用するアドレスで E メールを受信できることを確認します。バウンスと苦情通知の設定の詳細については、「Amazon SES のイベント通知の設定」を参照してください。
- 苦情通知が ISP またはメールシステムによってスパムとして報告されていないことを確認します。
- 通常、苦情通知には E メールの本文が含まれます。この点は、E メールのヘッダーのみが含まれるバウンス通知とは異なります。ただし、苦情通知では、苦情元の個人の E メールアドレスは削除されます。苦情元の E メールアドレスを特定できるように E メール本文に X ヘッダーや特別な識別子を埋め込むことをお勧めします。この方法を使用すると、苦情元のアドレスをより簡単に識別して、受信者リストから削除できるようになります。

メッセージの品質

E メールレシーバーは、メッセージの真偽を見分けるために、コンテンツフィルタを使用してメッセージ内の特定の属性を検出します。これらのコンテンツフィルタは、メッセージの内容を自動的に確認し、悪意あるメッセージの一般的な特性を識別します。Amazon SES では、コンテンツのフィルタリングテクノロジーを使用し、スパムやマルウェアを含むメッセージを送信前に検出してブロックします。

E メールレシーバーのコンテンツフィルタで、メッセージ内にスパムや悪意のある E メールの特性が検出されると、フラグが付けられて受取人の受信トレイから除外されます。

E メールを設計するときは、以下の点に留意してください。

- 最近のコンテンツフィルタはインテリジェントであり、絶えず適応し、変わり続けています。定義済みのルールのセットには依存しません。ReturnPath や Litmus などのサードパーティーのサービスを使用すると、どのような内容の E メールがコンテンツフィルタをトリガーする可能性があるかを確認できます。
- E メールにリンクが含まれている場合は、それらの URL が URIBL.com や SURBL.org などにある DNS ブラックリストに登録されていないかチェックします。
- リンク短縮ツールの使用を避けます。悪意のある送信者は、リンク短縮ツールを使用して実際のリンク先を隠す場合があります。ISP は、不正な目的でリンク短縮サービス - 最も評判の高いものであっても - が使用されていると判断すると、これらのサービス全体へのアクセスを拒否する場合があります。拒否リストに登録されたリンク短縮サービスへのリンクが含まれている E メールは、お客様の受信トレイに届かず、E メールキャンペーンの成功は危くなります。
- E メール内のすべてのリンクをテストし、目的のページを参照することを確認します。
- ウェブサイトにプライバシーポリシーと利用規約のドキュメントが含まれていること、これらのドキュメントが最新の内容であることを確認します。これらのドキュメントへのリンクは、送信する E メールごとに提供するのが適切です。これらのドキュメントへのリンクを提供することで、顧客に対する隠し事がないことを示し、信頼関係を築くのに役立ちます。
- 頻繁に変わるコンテンツ (「今日のお買い得情報」メッセージなど) を送信する場合は、E メールの内容がデプロイごとに異なることを確認します。頻繁に内容が変わるメッセージを送信する場合は、これらのメッセージがタイムリーで関連性が高いことを確認し、同じ繰り返しで煩わしさを感じさせないようにします。

良好な送信者評価の維持

Amazon SES における送信者評価とは、E メールプロバイダーやスパムフィルターによって認識される E メール送信者の信頼性や信用性を指します。送信者評価とは、E メールが正当と見なされ、受信者の受信トレイに正常に配信される可能性の尺度です。

以下のセクションでは、送信者評価を維持しながら、E メール通信が意図した対象者に確実に配信されるように、E メール送信に際して注意すべき重要な原則について説明します。

ドメインおよび「From」アドレスに関する考慮事項

- E メールの差出人のアドレスについてよく考えます。「From」アドレスは、受信者が最初に接する情報の 1 つであり、第一印象は長く残る場合があります。さらに、「From」アドレスと評価の関連付けを行う ISP もあります。
- 通信のタイプ別に異なるサブドメインを使用することを検討します。たとえば、ドメイン example.com を使用してマーケティングメッセージとトランザクションメッセージの両方を送信

する場合を考えます。この場合、すべてのメッセージを example.com から送信するのではなく、マーケティングメッセージは marketing.example.com などのサブドメインから、トランザクションメッセージは orders.example.com などのサブドメインから送信します。評価はサブドメイン別に確立されます。例えば、マーケティングメッセージがスパムトラップにかかったり、コンテンツフィルターをトリガーしたりした場合、サブドメインを使用していると、自社の評価が損なわれるリスクを軽減できます。

- 大量のメッセージを送信する場合は、sender@hotmail.com などの ISP ベースのアドレスから送信しません。sender@hotmail.com から大量のメッセージが送信されていることを ISP が気付くと、その E メールは自社所有の送信メール用ドメインから送信される E メールとは別の方法で処理されます。
- ドメインレジストラーと協力してドメインの WHOIS 情報が正しいことを確認します。正直かつ最新の WHOIS レコードを維持することは、透過性を重要視していることを示します。また、ユーザーはドメインの正当性を簡単に確認できます。
- 「From」アドレスまたは「Reply-to」アドレスとして no-reply@example.com などの no-reply アドレスを使用しないようにします。no-reply@ メールアドレスを使用すると、受信者に返信する方法を封じていること、受信者のフィードバックに関心がないことが明確なメッセージとして伝わります。

認証

- ドメインを SPF および SenderID で認証します。これらの認証方法により、各 E メールの送信元のドメインが真正のものであることを受取人が確認できます。
- 送信メールに DKIM で署名します。このステップにより、コンテンツが送受信の当事者間での転送中に変更されていないことを受取人が確認できます。
- SPF と DKIM の両方の認証設定をテストするには、個人の Gmail アカウントや Hotmail アカウントなどの ISP ベースの個人のメールアドレスに E メールを送信し、メッセージのヘッダーを確認します。ヘッダーを見て、メッセージの認証と署名が成功したかどうかがわかります。

リストの構築とメンテナンス

- ダブルオプトイン戦略を実装します。ユーザーがサインアップして E メールを受信する場合は、確認リンクを記載したメッセージを送信し、ユーザーがそのリンクをクリックしてアドレスを確認するまでは E メールの送信を開始しません。ダブルオプトイン戦略は、入力ミスによるハードバウンスの数を減らすのに役立ちます。

- ウェブベース形式でメールアドレスを集める場合は、収集時にそれらのアドレスに対して最小限の検証を実施します。たとえば、収集したアドレスの形式が正しいこと (recipient@example.com の形式になっていること)、および有効な MX レコードのドメインを参照していることを確認します。
- ユーザー定義の入力を無検査で Amazon SES に渡すことを許可するときは注意が必要です。フォーラムの登録やフォームの提出には、それなりのリスクがあります。コンテンツ全体をユーザーが生成し、スパム送信者が独自の内容をフォームに記入する可能性があるためです。高品質のコンテンツの E メールのみを送信することは、お客様の責任です。
- 標準エイリアス (postmaster@、abuse@、noc@ など) が意図的に E メールにサインアップするようなことは考えられません。メッセージは、実際に希望する相手にのみ送信します。このルールは、E メールのウォッチドッグとして通例予約されている標準エイリアスに特に該当します。これらのエイリアスは、評価を損なう目的で一種の悪意のある破壊活動としてリストに追加される場合があります。

コンプライアンス

- E メールの送信先の国や地域で適用される E メールマーケティングとスパム対策に関する法律および規制に注意してください。送信する E メールがこれらの法律に準拠していることを確認するのは、お客様の責任です。このガイドではこれらの法律については触れていませんので、お客様側で調べることが重要です。法律のリストについては、Wikipedia の「Email Spam Legislation by Country」を参照してください。
- 法的なアドバイスを受けるには、弁護士に相談してください。

SES での Amazon の使用 AWS SDK

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

SDK ドキュメント	コードの例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例

SDK ドキュメント	コードの例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	PowerShell コード例のツール
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

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

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

Amazon Simple Email Service の開始方法

この章では、Amazon SES の初期セットアップに必要なタスクと、開始に役立つチュートリアルについて説明します。

トピック

- Amazon Simple Email Service を設定する
- 別の E メール送信ソリューションから Amazon SES への移行
- 本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)

Amazon Simple Email Service を設定する

Amazon SES の使用を開始する前に、次のステップを完了する必要があります。

タスク

- にサインアップする AWS
- SES アカウントの設定
- プログラムによりアクセス許可を付与する (コンソールの外部で SES とやり取りするするため)
- AWS SDK のダウンロード (SES APIs を使用する場合)

にサインアップする AWS

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。

SES アカウントの設定

SES の使用を開始します。そのためには、E メールアドレスと送信ドメインを検証し、SES アカウント設定ウィザードを使用することにより SES 経由で E メールの送信を開始して、アカウントの本稼働アクセスをリクエストできるようにします。

SES アカウント設定ウィザードを使用してアカウントを設定する

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/ses/ で Amazon SES コンソールを開きます。
2. SES コンソールのホームページから [開始する] を選択すると、ウィザードが SES アカウントの設定ステップを順を追って案内します。

SES アカウント設定ウィザードは、SES で ID (E メールアドレスまたはドメイン) をまだ作成していない場合にのみ表示されます。

プログラムによりアクセス許可を付与する (コンソールの外部で SES とやり取りするするため)

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、 が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 アイデンティティセンター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 CLI、 AWS SDKs、または AWS APIs。	使用するインターフェイスの指示に従ってください。 • については AWS CLI、「 AWS Command Line Interface ユーザーガイド」の「 を使用する AWS CLI ように を設定する AWS

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
		IAM Identity Center」を参照してください。 • AWS SDKs、ツール、API については、 AWS APIs「 SDK およびツール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 CLI、 AWS SDKs、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 CLI、 AWS SDKs、または AWS APIs。	使用するインターフェイスの指示に従ってください。 • 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用した認証」を参照してください。 AWS SDKs • API AWS APIs「IAM 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。

AWS SDK のダウンロード (SES APIs を使用する場合)

raw HTTP リクエストの組み立てなどの低レベルの詳細を処理せずに SES APIs を呼び出すには、AWS SDK を使用できます。 AWS SDKsSES およびその他の AWS サービスの機能をカプセル化する関数とデータ型を提供します。 AWS SDK をダウンロードするには、 SDKsに移動します。SDK をダウンロードしたら、共有認証情報ファイルを作成し、 AWS アクセスキーを指定します。

別の E メール送信ソリューションから Amazon SES への移行

このトピックでは、E メール送信ソリューションを、オンプレミスや Amazon EC2 インスタンスでホストされているソリューションから Amazon SES に移動する場合に必要な手順について概要を示します。

このセクションのトピック:

- ステップ 1. ドメインの検証
- ステップ 2. 本番稼働用アクセスのリクエスト
- ステップ 3. ドメイン認証システムの設定
- ステップ 4. SMTP 認証情報の生成
- ステップ 5. SMTP エンドポイントへの接続
- 次のステップ

ステップ 1. ドメインの検証

Amazon SES を使用して E メールを送信する前に、E メールの送信元の ID を検証する必要があります。Amazon SES の場合、ID は E メールアドレスまたはドメイン全体です。ドメインを検証すると、Amazon SES を使用してそのドメインの任意のアドレスから E メールを送信できます。ドメインの検証の詳細については、「ドメイン ID の作成」を参照してください。

ステップ 2. 本番稼働用アクセスのリクエスト

初めて Amazon SES の使用を開始した時点では、アカウントはサンドボックス環境内にあります。アカウントがサンドボックスにある間は、検証済みアドレスにのみ E メールを送信できます。さらに、1 日に送信できるメッセージの数と、1 秒あたりに送信できるメッセージの数に制限があります。本番稼働用アクセスのリクエストの詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。

ステップ 3. ドメイン認証システムの設定

DKIM や SPF などの認証システムを使用するようにドメインを設定できます。このステップは技術的にはオプションです。ただし、ドメインに DKIM または SPF (またはその両方) を設定することで、E メールの配信性能を向上させ、顧客からの信頼度を高めることができます。SPF の設定の詳細については、「Amazon SES における SPF を使った E メールの認証」を参照してください。DKIM の設定の詳細については、「Amazon DKIMでの による E メールの認証 SES」を参照してください。

ステップ 4. SMTP 認証情報の生成

SMTP を使用するアプリケーションを使用して E メールを送信する場合は、SMTP 認証情報を生成する必要があります。SMTP 認証情報は、通常の AWS 認証情報とは異なります。これらの認証情報は、各 AWS リージョンでも一意です。SMTP 認証情報を生成する方法の詳細については、「Amazon SES SMTP 認証情報を取得」を参照してください。

ステップ 5. SMTP エンドポイントへの接続

postfix や sendmail などのメッセージ転送エージェントを使用する場合は、アプリケーションが Amazon SES SMTP エンドポイントを参照するように設定を更新する必要があります。SMTP エンドポイントの詳細なリストについては、「Amazon SES SMTP エンドポイントへの接続」を参照してください。前のステップで作成した SMTP 認証情報は、特定の AWS リージョンに関連付けられていることに注意してください。SMTP 認証情報を作成したリージョンの SMTP エンドポイントに接続する必要があります。

次のステップ

この時点で、Amazon SES を使用して E メールの送信を開始する準備が整いました。ただし、オプションとして実行できるステップがいくつかあります。

- 設定セットを作成できます。これは、送信する E メールに適用されるルールのセットです。たとえば、設定セットを使用して、E メールを配信したときや、受取人がメッセージを開いたりメッセージ内のリンクをクリックしたりしたとき、E メールがバウンスしたとき、受取人が E メールをスパムとしてマークしたときに、通知を送信する先を指定できます。詳細については、「Amazon SES の設定セットの使用」を参照してください。
- Amazon SES 経由で E メールを送信するときは、アカウントのバウンスや苦情をモニタリングすることが重要です。Amazon SES には、アカウントの返送やクレームを追跡するために使用できる評価メトリクスコンソールページが含まれています。詳細については、「評価メトリクスを使用

して返送率と苦情率を追跡する」を参照してください。また、これらのレートが高くなりすぎた場合に警告する CloudWatch アラームを作成することもできます。CloudWatch アラームの作成の詳細については、「CloudWatch を使用して評価モニタリングアラームを作成する」を参照してください。

- 大量の E メールを送信するユーザーや、自分の IP アドレスの評価を完全に制御したいユーザーは、追加の月額料金を支払って専用 IP アドレスをリースできます。詳細については、「Amazon 専用 IP アドレス SES」を参照してください。

本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)

不正使用や悪用を防止し、送信者としての評価を保つため、新しい Amazon SES アカウントには一定の制限が適用されます。

すべての新しいアカウントが Amazon SES サンドボックスに配置されます。アカウントのサンドボックスステータスは、それぞれ一意です AWS リージョン。アカウントはサンドボックスにありますが、Amazon SES のすべての機能を使用することができます。ただし、アカウントがサンドボックスにある場合、アカウントに次の制限が適用されます。

- E メールの送信先は、検証済み E メールアドレスおよびドメイン、または Amazon SES メールボックスシミュレーターに制限されます。
- 最大で 24 時間あたり 200 メッセージを送信できます。
- 最大で 1 秒あたり 1 メッセージを送信できます。
- 送信承認については、ユーザーも代理送信者も、検証されていない E メールアドレスに E メールを送信することはできません。
- アカウントレベルのサプレッションでは、サプレッションリスト管理に関連する一括アクションと SES API コールが無効になります。

アカウントをサンドボックスから本番環境に移行すると、受信者のアドレスまたはドメインの検証状況を問わず、任意の受信者に E メールを送信できます。 ただし、「From」、「Source」、「Sender」または「Return-Path」アドレスとして使用するすべての ID は引き続き検証する必要があります。

アカウントをサンドボックスから削除して本番環境に移行することをリクエストするには、このセクションの手順を実行します。

Tip

- 新規のお客様で、まだ ID を作成していない場合は、コンソールの SES アカウントの設定ウィザードが有効になり、使用開始のサポートを提供します。ウィザードへのアクセス方法については、「SES アカウントをセットアップする」を参照してください。
- 既に単一または複数の ID を作成済みの場合は、アカウントの設定ウィザードの代わりに、[セットアップを開始] ページが表示されます。
  - 既存の ID のいずれかが検証済みドメインである場合、[セットアップを開始] ページから本稼働アクセスのリクエストを直接実行することもできます。これは、本稼働アクセスのリクエストを行う前に SES でドメインを確認することがベストプラクティスであり、本稼働アクセスのリクエストをより迅速に承認し、直ちに E メールの送信を開始できる方法です。

Note

- Amazon SES を使用して Amazon EC2 インスタンスから E メールを送信する場合、Amazon EC2 インスタンスのポート 25 からスロットルを削除するようにリクエストする必要がある場合もあります。詳細については、「 AWS ナレッジセンター」のEC2 インスタンスからポート 25 のスロットルを削除する方法」を参照してください。

AWS Management Consoleを使用し、(サンドボックスからアカウントを削除して) 本稼働アクセスのリクエストを行うには

1. Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションパネルで、[アカウントダッシュボード] を選択します。
3. コンソール上部の「Amazon SES アカウントはサンドボックスにあります」という警告ボックスの右側で、[セットアップを開始ページを表示する]、[本稼働アクセスのリクエスト] の順に選択します。
4. アカウントの詳細モーダルで、送信するメールの大部分を最もよく表すマーケティング またはトランザクション ラジオボタンのいずれかを選択します。
   - マーケティングメール - 購入、ダウンロード情報など、マーケティングおよびプロモーションコンテンツを含み、見込み客または顧客のターゲットリストに、1 対多で送信されます。

- トランザクションメール - 各受信者に固有の 1 対 1 で送信され、通常、ウェブサイトの購入、パスワードのリセットリクエストなどのユーザーアクションによってトリガーされます。

5. ウェブサイトの URL で、ウェブサイトの URL を入力して、送信予定のコンテンツの種類について分かるようにします。
6. 追加の連絡先 で、アカウントに関するコミュニケーションの受信場所をお知らせください。最大 4 つの E メールアドレスをカンマで区切って指定できます。
7. [優先する交信言語] で、本件に関して連絡を受ける際の言語として、[英語] か [日本語] を選択します。
8. 確認 で、明示的にリクエストした個人にのみメールを送信することに同意するチェックボックスをオンにし、バウンスや苦情の通知を処理するプロセスが設定されていることを確認します。
9. リクエストの送信ボタンを選択すると、-リクエストが送信されたことを確定し、現在審査中であることを示すバナーが表示されます。

アカウントの詳細のレビューを送信すると、レビューが完了するまで詳細を編集できなくなります。AWS サポート チームは 24 時間以内にリクエストに最初の応答を提供します。

迷惑なコンテンツや悪意のあるコンテンツを送信するためにシステムが悪用されないように、各リクエストを慎重に検討する必要があります。可能であれば、24 時間以内にリクエストを承認します。ただし、お客様から追加情報を取得する必要がある場合は、お客様のリクエストの解決に時間がかかる場合があります。

オプションで、 を使用して本番稼働用アクセスのリクエストを送信することもできます AWS CLI。を使用してリクエストを送信する AWS CLI と、多数の ID の本稼働アクセスをリクエストする場合や、Amazon SES のセットアッププロセスを自動化する場合に役立ちます。

AWS CLIを使用して、アカウントが Amazon SES サンドボックスから削除されるようにリクエストするには

1. 前提条件 : AWS CLIをインストールして設定する必要があります。詳細については、AWS Command Line Interface ユーザーガイドを参照してください。
2. コマンドラインで以下のコマンドを入力します。

```
aws sesv2 put-account-details \
--production-access-enabled \
--mail-type TRANSACTIONAL \
--website-url https://example.com \
--additional-contact-email-addresses info@example.com \
```

```
--contact-language EN
--use-case-description "Use-case description"
```

上記のコマンドで、次の操作を行います。

a. ########を Amazon SES 経由で送信する予定の E メールのタイプに置き換えます。TRANSACTIONAL または MARKETING のどちらかを指定できます。複数の値が当てはまる場合は、送信する E メールの大部分に当てはまるオプションを指定します。

b. *https://example.com* を、ウェブサイトの URL に置き換えます。この情報の提供は、お客様が送信を予定しているコンテンツのタイプを正しく理解するために役立ちます。

c. *info@example.com* を、アカウントに関するお知らせの受信先となるメールアドレスに置き換えます。最大 4 つの E メールアドレスをカンマで区切って指定できます。

d. *EN* 任意の言語を使用します。英語の場合は EN を、日本語の場合は JA を指定します。

e. #########をユースケースの説明に置き換えます。

アカウントの詳細のレビューを送信すると、レビューが完了するまで詳細を編集できなくなります。AWS サポート チームは 24 時間以内にリクエストに最初の応答を提供します。

迷惑なコンテンツや悪意のあるコンテンツを送信するためにシステムが悪用されないように、各リクエストを慎重に検討する必要があります。可能であれば、24 時間以内にリクエストを承認します。ただし、お客様から追加情報を取得する必要がある場合は、お客様のリクエストの解決に時間がかかる場合があります。

Amazon SES 送信制限の管理

Amazon SES アカウントには、送信できる E メールメッセージの数と送信レートを規制するための送信クォータのセットがあります。送信クォータは、Amazon SES と E メールプロバイダーとの間の信頼関係を保つという意味で、Amazon SES を利用するすべてのお客様に利益をもたらします。E メール送信のボリュームやレートが不意に急増すると、E メールプロバイダーによって E メールがブロックされますが、送信クォータがあることで、送信アクティビティを徐々に増やすことができ、その可能性を低く抑えることができます。

以下のクォータは、Amazon SES 経由の E メール送信に適用されます。

- 送信クォータ - 24 時間当たりに送信できる E メールの最大数。このクォータは、ローリング期間で計算されます。メールを送信しようとするたびに、Amazon SES は過去 24 時間以内に送信したメールの数を決定します。過去 24 時間以内に送信した E メールの総数が 1 日の最大数より少ない限り、送信リクエストは受け入れられ、E メールが送信されます。

  メッセージの送信がアカウントの 1 日の最大数を超える場合、Amazon SES への呼び出しは拒否されます。

- 送信レート - Amazon SES が 1 秒あたりにアカウントから受け付ける E メールの最大数。このクォータを瞬間的に超えることはできますが、制限を超えた状態が長時間続くことは許可されません。

  > **Note**
  >
  > ご使用のアカウントで、Amazon SES のメッセージ受信レートが、この最大送信レートよりも少なくなる場合があります。

- 最大メッセージサイズ (MB)-送信できる E メールの最大サイズ。これには、MIME エンコード後の E メールの一部である画像と添付ファイルが含まれます。たとえば、5 MB のファイルをアタッチすると、MIME エンコード後の E メールの添付ファイルのサイズは、約 6.85MB (元のファイルサイズの約 137%) になります。

  > **Note**
  >
  > 添付ファイルを cloud drive にアップロードし、cloud drive の添付ファイルの URL を含めることをお勧めします。これにより、メールのサイズを小さくして配信可能性を向上させ

ます。SES では、異なるメールサーバーのサイズに基づくポリシーが異なるため、大きなメールが受信者のメールボックスに届くことを保証できません。

Amazon SES 送信クォータは、AWSリージョンごとに独立しています。複数のAWSリージョンで Amazon SES を使用する方法については、「リージョンと Amazon SES」を参照してください。

アカウントが Amazon SES サンドボックス内にある場合、 24 時間あたり 200 メッセージのみ送信でき、最大送信レートは毎秒 1 メッセージです。サンドボックスからアカウントを削除するリクエストを送信するとき、同時にクォータの引き上げをリクエストすることもできます。アカウントをサンドボックスから削除する方法については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。

アカウントがサンドボックスから削除されたら、AWS サポートセンターで新しいケースを作成することで、いつでもさらにクォータの引き上げをリクエストできます。詳細については、「Amazon SES 送信クォータの引き上げ」を参照してください。

> **Note**
>
> メッセージ数ではなく、受取人数に基づいた送信クォータがあります。たとえば、受取人数が 10 人である E メールは、クォータに対しては 10 通とカウントされます。ただし、`SendEmail` API オペレーションの 1 回の呼び出しで複数の受信者にメールを送信することはお勧めしません。呼び出しが失敗すると、メール全体が拒否されるためです。`SendEmail` は、受取人ごとに 1 回呼び出すようお勧めします。

- 送信クォータの引き上げについては、「Amazon SES 送信クォータの引き上げ」を参照してください。
- Amazon SES コンソールまたは Amazon SES API を使用して送信クォータをモニタリングするには、「Amazon SES 送信クォータのモニタリング」を参照してください。
- 送信クォータに達したときにアプリケーションが受け取るエラーについては、「Amazon SES アカウントの送信クォータに関するエラー」を参照してください。

Amazon SES 送信クォータの引き上げ

お客様のアカウントには、現在のリージョンごとに引き上げることができる以下のクォータがあります。

リソース	デフォルトのクォータ	説明
送信クォータ	200	現在のAWS リージョンで、このアカウントで 24 時間に送信できるEメールの最大数。
送信レート	1	Amazon SES が 1 秒あたりに現在のAWS リージョンのこのアカウントに対して受け付ける E メールの最大数。

送信クォータの自動引き上げ

アカウントがサンドボックスの外にあり、質の高い E メールを送信しようとする場合、本稼働アカウント用に送信クォータが自動的に引き上げられる可能性があります。多くの場合、これらのクォータを実際に引き上げる必要が生じる前に、クォータは自動的に引き上げられます。

レートが自動的に引き上げられる対象になるには、次のすべての条件が満たされている必要があります。

- 受取人が受信を希望する高品質のコンテンツを送信する – 受取人の希望や期待に沿ったコンテンツを送信します。E メールを希望しないお客様へのメール送信を停止します。
- 実際の本稼働コンテンツを送信する フェイクの E メールアドレスにテストメッセージを送信する – バウンス率や苦情率に悪い影響が出る可能性があります。また、内部の受取人にのみメッセージを送信すると、お客様が受信を希望するコンテンツを送信しているかどうかの判断が難しくなります。ただし、内部以外の受信者に本稼働メッセージを送信する場合は、E メール送信プラクティスを正確に評価できます。
- 現在のクォータに近いボリュームを送信する - 自動クォータ引き上げの対象となるには、1 日あたりの E メール量がアカウントの 1 日あたりの最大メール量を超えることはないが定期的にそれに近くなっている必要があります。
- バウンス率と苦情率が低い – 受信するバウンスと苦情の件数は最小限に抑えます。バウンスと苦情の件数の多さが、送信クォータに悪影響を及ぼすことがあります。

ユーザーが要求した送信クォータの増加

現在の送信クォータがニーズに見合っていないが、自動的に引き上げられていない場合は、引き上げをリクエストできます。

- 送信クォータまたは送信レート — これらのいずれかに対する引き上げリクエストは、AWS Service Quotas コンソールを通じて送信できます。

Service Quotasコンソールを使用して Amazon SES 送信クォータの増加をリクエストするには

1. Service Quotas コンソールを開きます。
2. コンソールの右上隅 (アカウント番号の横) にあるドロップダウンを使用して、増やしたいリージョンを選択します。
3. ナビゲーションペインで、[AWS のサービス] を選択します。
4. Amazon Simple Email Service (SES) を選択します。
5. クォータを選択して、指示に従ってクォータの引き上げをリクエストします。

> **増加リクエストタイプについての AWS サポート チーム SLA**
>
> 迷惑なコンテンツや悪意のあるコンテンツを送信するためにシステムが悪用されないように、各リクエストを慎重に検討する必要があります。可能な場合は、次のリストにある指定された時間内に、増加のタイプについてのリクエストを承認します。ただし、お客様から追加情報を取得する必要がある場合は、お客様のリクエストの解決に時間がかかる場合があります。お客様のユースケースが当社の方針に沿わない場合、当社はリクエストを承認しない権利を留保します。
>
> - 送信クォータまたは送信レート: 最長 24 時間。

> **Note**
>
> Service Quotas コンソールは多くの言語で利用できますが、実際のサポートは英語でのみ提供されます。

Amazon SES 送信クォータのモニタリング

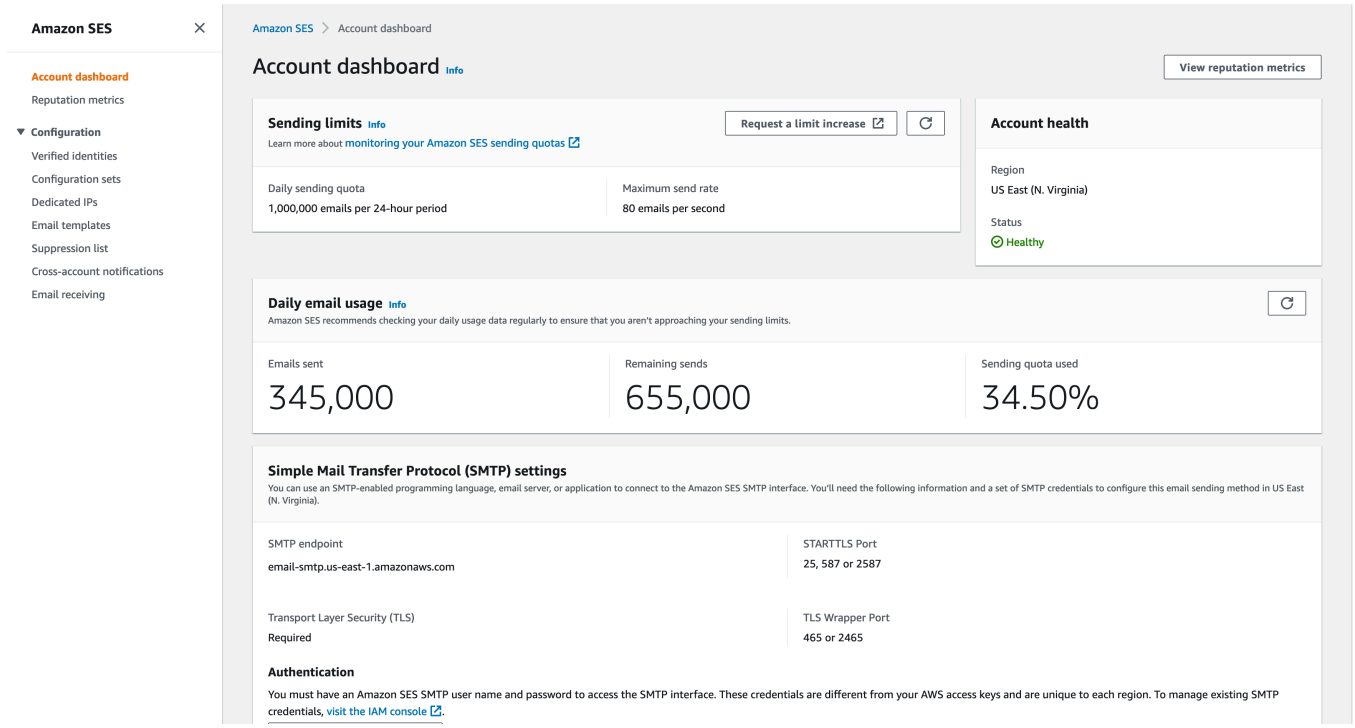
送信クォータをモニタリングするには、Amazon SES コンソールまたは Amazon SES API を使用します。これを行うには、クエリ (HTTPS) インターフェイスを直接呼び出すか、AWSSDK、AWS Command Line Interface 、または AWS Tools for Windows PowerShell を使用して間接的に呼び出します。

> **Important**
>
> 送信クォータに達しないように、頻繁に送信統計を確認することをお勧めします。送信クォータに近付いた場合は、クォータを引き上げる方法について「Amazon SES 送信クォータの引き上げ」を参照してください。送信クォータに達する前に、クォータの引き上げを検討してください。

Amazon SES コンソールを使用した送信クォータのモニタリング

次の手順では、Amazon SES コンソールを使用した送信クォータの表示方法を示します。

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインで、[アカウントダッシュボード] を選択します。送信クォータが [送信制限] に表示されます。送信済み E メールの合計数、残りの送信数、送信クォータの使用割合は、日次 E メール使用量に表示されます。



3. 表示を更新するには、日次 E メール使用量ボックスの右上隅にある更新アイコンを選択します。

Amazon SES API を使用した送信クォータのモニタリング

Amazon SES API には送信クォータを返す `GetSendQuota` アクションがあります。`GetSendQuota` アクションを呼び出すと、以下の情報が返されます。

- 過去 24 時間に送信した E メールの数
- 現在の 24 時間の送信クォータ
- 最大送信レート

> **Note**
>
> `GetSendQuota` の説明については、「Amazon Simple Email Service API リファレンス」を参照してください。

Amazon SES アカウントの送信クォータに関するエラー

1 日あたりの送信クォータ (24 時間以内に送信できる E メールの最大数) や最大送信レート (1 秒間あたりに送信できるメッセージの最大数) に達した後で E メールを送信しようとすると、Amazon SES はメッセージを削除し、再配信を試行しません。また、Amazon SES は問題を説明するエラーメッセージを提供します。このエラーメッセージを Amazon SES で生成する方法は、E メールの送信を試行した方法によって異なります。このトピックでは、Amazon SES API と SMTP インターフェイスを介して受信したメッセージに関する情報を示します。

最大送信レートに達したときに使用できる手法については、AWSメッセージングとターゲティングブログの「”Throttling – Maximum sending rate exceeded (スロットリング – 最大送信レートの超過)”の対処法」を参照してください。

Amazon SES API で送信制限に達した場合

Amazon SES API (またはAWS SDK) を使用して E メールを送信しようとしたときに、アカウントの送信制限をすでに超えている場合は、API によってThrottlingException工ラーが生成されます。このエラーメッセージには、以下のいずれかのメッセージが含まれます。

- `Daily message quota exceeded`
- `Maximum sending rate exceeded`

スロットリングエラーが発生した場合は、最大 10 分間待機した後で送信リクエストを再試行するようにアプリケーションをプログラムする必要があります。

SMTP で送信制限に達した場合

Amazon SES SMTP を使用して E メールを送信しようとしたときに、アカウントの送信制限をすでに超えている場合、SMTP クライアントは以下のいずれかのエラーを表示する場合があります。

- `454 Throttling failure: Maximum sending rate exceeded`
- `454 Throttling failure: Daily message quota exceeded`

これらのエラーの処理方法は、SMTP クライアントごとに異なります。

Amazon SES での E メール送信のセットアップ

Amazon Simple Email Service (Amazon SES) では、Amazon SES コンソール、Amazon SES Simple Mail Transfer Protocol (SMTP) インターフェイス、または Amazon SES API を使用して E メールを送信することができます。通常、テスト E メールを送信して送信アクティビティを管理する場合は、コンソールを使用します。一括 E メールを送信する場合には、SMTP インターフェイスまたは API を使用します。Amazon SES の料金に関する情報については、「Amazon SES の料金」を参照してください。

- SMTP 対応のソフトウェアパッケージ、アプリケーション、またはプログラミング言語を使用し、Amazon SES を介して E メールを送信する場合や、Amazon SES を既存のメールサーバーに統合する場合には、Amazon SES SMTP インターフェイスを使用します。詳細については、「プログラミングで Amazon SES の SMTP インターフェイスを介して E メールを送信する」を参照してください。
- 未処理の HTTP リクエストを使用して、Amazon SES を呼び出す場合には、Amazon SES API を使用します。詳細については、「Amazon SES API を使用して E メールを送信する」を参照してください。

> **Important**
>
> 複数の受取人 (受取人は「To」、「CC」、「BCC」のアドレス) にメールを送信する場合、Amazon SES の呼び出しに失敗すると、E メール全体が拒否されて、どの受信者も目的のメールを受信できません。そのため、1 回につき 1 人の受信者に E メールを送信することをお勧めします。

Amazon SES SMTP インターフェイスを使用して E メールを送信

Amazon SES を介して本稼働 E メールを送信する場合には、Simple Mail Transfer Protocol (SMTP) インターフェイスまたは Amazon SES API を使用できます。Amazon SES API の詳細については、「Amazon SES API を使用して E メールを送信する」を参照してください。このセクションでは、SMTP インターフェイスについて説明します。

Amazon SES は、インターネットで最も一般的な E メールプロトコルである SMTP を使用してメールを送信します。SMTP 対応の各種プログラミング言語やソフトウェアを使用して Amazon SES SMTP インターフェイスに接続することで、Amazon SES を介して E メールを送信できます。この

セクションでは、Amazon SES を介して E メールを送信するために、Amazon SES の SMTP 認証情報を取得する方法、SMTP インターフェイスを使用して E メールを送信する方法、および、各種のソフトウェアおよびメールサーバーの設定方法を説明します。

SMTP インターフェイスを介した Amazon SES の使用に伴う一般的な問題の解決策については、「Amazon SES SMTP の問題」を参照してください。

SMTP 経由で E メールを送信するための要件

Amazon SES SMTP インターフェイスを使用して E メールを送信するには、次の参照が必要です。

- SMTP エンドポイントアドレス。Amazon SES　SMTP エンドポイントのリストについては、Amazon SES SMTP エンドポイントへの接続を参照してください。
- SMTP インターフェイスのポート番号。ポート番号は接続方法によって変わります。詳細については、「Amazon SES SMTP エンドポイントへの接続」を参照してください。
- SMTP ユーザー名とパスワード。SMTP 認証情報は、各 AWS 地域に固有です。複数の AWS リージョンで SMTP インターフェイスを使用してメールを送信する予定がある場合は、リージョンごとに SMTP 認証情報が必要です。

> **Important**
>
> SMTP 認証情報は、Amazon SES コンソールへのサインインに使用する AWS アクセスキーまたは認証情報と同じではありません。SMTP 認証情報を生成する方法については、「Amazon SES SMTP 認証情報を取得」を参照してください。

- Transport Layer Security (TLS) を使用して通信できるクライアントソフトウェア。詳細については、Amazon SES SMTP エンドポイントへの接続を参照ください。
- Amazon SES で検証済みの E メールアドレス。詳細については、Amazon SES の検証済みIDを参照ください。
- 大量の E メールを送信する場合は、送信クォータの引き上げが必要です。詳細については、「Amazon SES 送信制限の管理」を参照ください。

SMTP 経由で E メールを送信する方法

SMTP 経由で E メールを送信するには、次のいずれかの方法を使用できます。

- SMTP 対応ソフトウェアを、Amazon SES SMTP インターフェイスを介して E メールを送信するように設定するには、「ソフトウェアパッケージを使用し、Amazon SES を介して E メールを送信します」を参照ください。
- Amazon SES を介して E メールを送信するようにアプリケーションをプログラムするには、「プログラミングで Amazon SES の SMTP インターフェイスを介して E メールを送信する」を参照ください。
- Amazon SES を介してすべての送信 E メールを送信するために、既存の E メールサーバーを設定する方法については、「Amazon SES を既存の E メールサーバーと統合します」を参照してください。
- テストを行うときに有効なコマンドラインを使用して、Amazon SES SMTP インターフェイスを操作するためには、「コマンドラインを使用して、Amazon SES SMTP インターフェイスへの接続をテストする」を参照ください。

SMTP 応答コードのリストについては、「Amazon SES から返される SMTP 応答コード」を参照ください。

提供する E メール情報

SMTP インターフェイスで Amazon SES にアクセスする場合、SMTP クライアントアプリケーションでメッセージがアセンブルされるため、提供する必要のある情報は使用するアプリケーションによって異なります。少なくとも、クライアントとサーバー間の SMTP 交換には、次が必要になります。

- 送信元アドレス
- 送信先アドレス
- メッセージデータ

SMTP インターフェイスを使用していて、フィードバック転送が有効になっている場合、バウンス、苦情、配信通知は "MAIL FROM" アドレスに送信されます。指定した、いずれの "Reply-To" アドレスは使用されません。

Amazon SES SMTP 認証情報を取得

SES SMTP インターフェイスにアクセスするには、Amazon SES SMTP 認証情報が必要です。

SES SMTP インターフェイスを介して E メールを送信するために使用する認証情報は、各 AWS リージョンに固有です。SES SMTP インターフェイスを使用して複数のリージョンでメールを送信する場合は、使用しようとしている各リージョンで SMTP 認証情報のセットを生成する必要があります。

SMTP パスワードは、 AWS シークレットアクセスキーとは異なります。認証情報の詳細については、「Amazon SES 認証情報の種類」を参照してください。

> **Note**
>
> SMTP エンドポイントは現時点では、アフリカ (ケープタウン)、アジアパシフィック (ジャカルタ)、欧州 (ミラノ)、イスラエル (テルアビブ)、中東 (バーレーン) では使用できません。

SES コンソールを使用して SES SMTP 認証情報を取得する

要件

IAM ユーザーは SES SMTP 認証情報を作成できますが、SES SMTP 認証情報の作成には IAM が使用されるため、ユーザーのポリシーによって IAM 自体を使用するアクセス許可がユーザーに付与される必要があります。IAM ポリシーで許可する必要がある IAM アクションは、`iam:ListUsers`、`iam:CreateUser`、`iam:CreateAccessKey`、および`iam:PutUserPolicy`です。コンソールを使用して SES SMTP 認証情報を作成しようとしたときに、IAM ユーザーにこれらのアクセス許可がない場合、アカウントが「iam:ListUsers."

> **Important**
>
> 上記の IAM アクションには、サービスでリソースのアクセス許可を付与または変更するアクセス許可が付与されるため、IAM レベルが最も高いアクセス許可管理アクセスレベルがあります。したがって、 AWS アカウントのセキュリティを向上させるために、アクセス許可管理アクセスレベル分類を含むこれらのポリシーを制限または定期的にモニタリングすることを強くお勧めします。

SMTP 認証情報を作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/ses/ で Amazon SES コンソールを開きます。

2. 左のナビゲーションペインで [SMTP settings] (SMTP 設定) を選択します。[Simple Mail Transfer Protocol (SMTP) settings] (Simple Mail Transfer Protocol (SMTP) の設定) ページが開きます。
3. 右上の [Create SMTP Credentials] (SMTP 認証情報の作成) を選択します。IAM コンソールが開きます。
4. (オプション) 既に作成した SMTP ユーザーを表示、編集、または削除する必要がある場合は、右下の [Manage my existing SMTP credentials] (既存の SMTP 認証情報の管理) を選択します。IAM コンソールが開きます。SMTP 認証情報の管理の詳細は、次の手順に従って表示されます。
5. [SMTP のユーザーを作成] で、[ユーザー名] フィールドに SMTP ユーザーの名前を入力します。または、このフィールドに提供されているデフォルト値を使用できます。完了したら、右下隅の [ユーザーを作成] を選択します。
6. [SMTP パスワード] で [表示] を選択します。SMTP 認証情報が画面に表示されます。
7. [.csv ファイルをダウンロード] を選択してこれらの認証情報をダウンロードし、安全な場所に保管します。このダイアログボックスを閉じると、認証情報の表示や保存はできなくなります。
8. [SES コンソールに戻る] を選択します。

この手順で作成した SMTP 認証情報を一覧表示するには、IAM コンソールの [Access management] (アクセス管理) で [Users] (ユーザー) を選択し、検索バーを使用して SMTP 認証情報を割り当てたすべてのユーザーを見つけます。

IAM コンソールを使用して、既存の SMTP ユーザーを削除することもできます。ユーザーの削除の詳細については、IAM 入門ガイドの「IAM ユーザーの管理」を参照してください。

SMTP パスワードを変更する場合は、IAM コンソールで既存の SMTP ユーザーを削除します。次に、前の手順を完了して、新しい SMTP 認証情報のセットを作成します。

既存の認証情報を変換して SES SMTP AWS 認証情報を取得する

IAM インターフェイスを使用して設定したユーザーがいる場合は、ユーザーの SES SMTP 認証情報をその AWS ユーザーの認証情報から取得できます。

> **Important**
>
> 一時的な AWS 認証情報を使用して SMTP 認証情報を取得しないでください。SES SMTP インターフェイスは、一時的なセキュリティ認証情報から生成された SMTP 認証情報をサポートしていません。

IAM ユーザーが SES SMTP インターフェイスを使用してメールを送信できるようにするには

1. 以下の手順に従って、このセクションで提供されるアルゴリズムを使用して、認証情報からユーザーの SMTP AWS 認証情報を取得します。

   AWS 認証情報から開始するため、SMTP ユーザー名は AWS アクセスキー ID と同じであるため、SMTP パスワードを生成するだけで済みます。
2. 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。
3. [アクセス管理] で、[ポリシー]、[ポリシーの作成] の順に選択します。
4. [ポリシーエディター] で、[JSON] を選択して、エディタ内のコード例をすべて削除します。
5. 以下の許可ポリシーをエディタに貼り付けます。

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

6. [次へ] をクリックし、[ポリシー名] フィールドに `AmazonSesSendingAccess` と入力して、[ポリシーの作成] をクリックします。
7. [アクセス管理] で、[ユーザーグループ]、[グループの作成] の順に選択します。
8. [ユーザーグループ名] フィールドに、`AWSSESSendingGroupDoNotRename` と入力します。
9. [ユーザーをグループに追加] テーブルから SMTP ユーザーを選択して、グループに追加します。

10. 以前に作成した AmazonSesSendingAccess ポリシーを [許可ポリシーをアタッチ] テーブルから選択してアタッチし、[ユーザーグループを作成] をクリックします。

IAM での SES の使用の詳細については、「Amazon での Identity and Access Management SES」を参照してください。

> **Note**
>
> SES SMTP 認証情報はどの IAM ユーザーに対しても生成できますが、SMTP 認証情報を生成するときには、個別の IAM ユーザーを作成することをお勧めします。目的別にユーザーを作成することが推奨される理由については、「IAM のベストプラクティス」を参照ください。

次の擬似コードは、 AWS シークレットアクセスキーを SES SMTP パスワードに変換するアルゴリズムを示しています。

```
// Modify this variable to include your AWS secret access key
key = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY";

// Modify this variable to refer to the AWS Region that you want to use to send email.
region = "us-west-2";

// The values of the following variables should always stay the same.
date = "11111111";
service = "ses";
terminal = "aws4_request";
message = "SendRawEmail";
version = 0x04;

kDate = HmacSha256(date, "AWS4" + key);
kRegion = HmacSha256(region, kDate);
kService = HmacSha256(service, kRegion);
kTerminal = HmacSha256(terminal, kService);
kMessage = HmacSha256(message, kTerminal);
signatureAndVersion = Concatenate(version, kMessage);
smtpPassword = Base64(signatureAndVersion);
```

一部のプログラミング言語に含まれているライブラリを使用して、IAM シークレットアクセスキーを SMTP パスワードに変換できます。このセクションでは、Python を使用して AWS シークレットアクセスキーを SES SMTP パスワードに変換するために使用できるコード例を示します。

> **Note**
>
> 次の例では、Python 3.6 で導入されたf 文字列を使用しています。 古いバージョンでは使用できません。
> 現在、Python SDK (Boto3) は公式に 2.7 と 3.6 (またはそれ以降) をサポートしています。ただし、2.7 のサポートは廃止され、2021 年 7 月 15 日に削除されるため、少なくとも 3.6 にアップグレードする必要があります。

Python

```
#!/usr/bin/env python3

import hmac
import hashlib
import base64
import argparse

SMTP_REGIONS = [
    "us-east-2",  # US East (Ohio)
    "us-east-1",  # US East (N. Virginia)
    "us-west-2",  # US West (Oregon)
    "ap-south-1",  # Asia Pacific (Mumbai)
    "ap-northeast-2",  # Asia Pacific (Seoul)
    "ap-southeast-1",  # Asia Pacific (Singapore)
    "ap-southeast-2",  # Asia Pacific (Sydney)
    "ap-northeast-1",  # Asia Pacific (Tokyo)
    "ca-central-1",  # Canada (Central)
    "eu-central-1",  # Europe (Frankfurt)
    "eu-west-1",  # Europe (Ireland)
    "eu-west-2",  # Europe (London)
    "eu-south-1",  # Europe (Milan)
    "eu-north-1",  # Europe (Stockholm)
    "sa-east-1",  # South America (Sao Paulo)
    "us-gov-west-1",  # AWS GovCloud (US)
    "us-gov-east-1",  # AWS GovCloud (US)
]
```

```
# These values are required to calculate the signature. Do not change them.
DATE = "11111111"
SERVICE = "ses"
MESSAGE = "SendRawEmail"
TERMINAL = "aws4_request"
VERSION = 0x04


def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()


def calculate_key(secret_access_key, region):
    if region not in SMTP_REGIONS:
        raise ValueError(f"The {region} Region doesn't have an SMTP endpoint.")

    signature = sign(("AWS4" + secret_access_key).encode("utf-8"), DATE)
    signature = sign(signature, region)
    signature = sign(signature, SERVICE)
    signature = sign(signature, TERMINAL)
    signature = sign(signature, MESSAGE)
    signature_and_version = bytes([VERSION]) + signature
    smtp_password = base64.b64encode(signature_and_version)
    return smtp_password.decode("utf-8")


def main():
    parser = argparse.ArgumentParser(
        description="Convert a Secret Access Key to an SMTP password."
    )
    parser.add_argument("secret", help="The Secret Access Key to convert.")
    parser.add_argument(
        "region",
        help="The AWS Region where the SMTP password will be used.",
        choices=SMTP_REGIONS,
    )
    args = parser.parse_args()
    print(calculate_key(args.secret, args.region))


if __name__ == "__main__":
    main()
```

このスクリプトを使用して SMTP パスワードを取得するには、上記のコードをsmtp_credentials_generate.pyとして保存します。コマンドラインから、以下のコマンドを実行します。

```
python path/to/smtp_credentials_generate.py wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY us-east-1
```

上記のコマンドで、次の操作を行います。

- *path/to/*を保存した場所へのパスに置き換えますsmtp_credentials_generate.py。
- *wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY* を、SMTP パスワードに変換するシークレットアクセスキーに置き換えます。
- *us-east-1* を SMTP 認証情報を使用する AWS リージョンに置き換えます。

このスクリプトが正常に実行されると、SMTP パスワードだけが出力されます。

SMTP ユーザーの既存のインラインポリシーからグループポリシーへの移行 (セキュリティに関する推奨事項)

> **Important**
>
> 2024 年 9 月 6 日より前に SES SMTP 認証情報を作成した場合、インラインポリシーとタグが SMTP ユーザーにアタッチされています。SES ではインラインポリシーの使用を排除する方向であり、セキュリティに関する推奨事項と同じ方法を採用することをお勧めします。

既存のインラインポリシーからグループポリシーに SMTP ユーザーを移行する前に、まず SES アクセス許可ポリシーを使用して IAM ユーザーグループを作成し、インラインポリシーの代わりにする必要があります。この IAM ユーザーグループを既に作成している場合、または 2024 年 9 月 6 日以降に作成した SMTP 認証情報用にユーザーグループ自動的に作成されている場合は、以下の手順のステップ 10 に直ちにスキップできます。

既存のインラインポリシーからマネージドグループに移行するには

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. [アクセス管理] で、[ポリシー]、[ポリシーの作成] の順に選択します。

3. [ポリシーエディター] で、[JSON] を選択して、エディタ内のコード例をすべて削除します。
4. 以下の許可ポリシーをエディタに貼り付けます。

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

5. [次へ] をクリックし、[ポリシー名] フィールドに AmazonSesSendingAccess と入力して、[ポリシーの作成] をクリックします。
6. [アクセス管理] で、[ユーザーグループ]、[グループの作成] の順に選択します。
7. [ユーザーグループ名] フィールドに、AWSSESSendingGroupDoNotRename と入力します。
8. [ユーザーをグループに追加] テーブルから SMTP ユーザーを選択して、グループに追加します。
9. 以前に作成した AmazonSesSendingAccess ポリシーを [許可ポリシーをアタッチ] テーブルから選択してアタッチし、[ユーザーグループを作成] をクリックします。

   SES アクセス許可ポリシーを使用して IAM ユーザーグループを作成したら、残りのステップで説明されるとおり、SMTP ユーザーを現在のインラインポリシーからこのグループポリシーに移行できます。
10. [アクセス管理] で、[ユーザー] をクリックして、移行する SMTP ユーザーを選択します。
11. [グループ] タブをクリックして、[ユーザーをグループに追加] を選択します。
12. AWSSESSendingGroupDoNotRename グループを選択してから、[ユーザーをグループに追加] をクリックします。
13. [許可] タブをクリックして、[ポリシー名] 列に AmazonSesSendingAccess が 2 行表示されており、1 つはインラインで、もう 1 つには [次を経由してアタッチ] 列に *AWSSESSendingGroupDoNotRename* グループが表示されていることを確認します。
14. [ポリシー名] 列に AmazonSesSendingAccess が含まれ、[次を経由してアタッチ] 列にインラインが含まれている行のみを選択して、[削除] をクリックし、[ポリシーを削除] をクリックして確定します。

[次を経由してアタッチ] 列に *AWSSESSendingGroupDoNotRename* グループが含まれる行が残っているかを検証します。

15. [タグ] タブをクリックして、[タグを管理] をクリックします。
16. [キー] 列に「InvokedBy」が含まれ、[値] 列に「SESConsole」が含まれる行の横にある [削除] をクリックしてから、[変更を保存] をクリックします。

> **Important**
>
> 送信に影響が及ばないように、`AmazonSesSendingAccess` ポリシー (インラインポリシーまたはグループポリシー、またはその両方) は、SMTP ユーザーにアタッチされたままにしておく必要があります。インラインポリシーは、グループポリシーがユーザーにアタッチされた後にのみ削除します。

Amazon SES SMTP エンドポイントへの接続

Amazon SES SMTP インターフェイスを使用して E メールを送信するには、SMTP エンドポイントに接続します。Amazon SES SMTP エンドポイントの完全なリストについては、「AWS 全般のリファレンス」の「Amazon Simple Email サービスエンドポイントとクォータ」を参照してください。

Amazon SES SMTP エンドポイントでは、すべての接続が Transport Layer Security (TLS) を使用して暗号化されている必要があります。(TLS は、以前のプロトコルの名前である「SSL」と呼ばれることが多いことに注意してください。) Amazon SES は、TLS で暗号化された接続を確立するために、STARTTLS および TLS ラッパーという 2 つのメカニズムをサポートしています。ソフトウェアが STARTTLS および TLS ラッパーをサポートしているかどうかは、ソフトウェアのドキュメントを参照してください。

Amazon Elastic Compute Cloud (Amazon EC2) では、デフォルトでポート 25 経由での E メールトラフィックを調整しています。EC2 から SMTP エンドポイントを介して E メールを送信する際のタイムアウトを回避するには、スロットルを削除するため、E メール送信制限の削除要求を送信します。または、別のポートを使用して E メールを送信するか、Amazon VPC エンドポイントを使用することもできます。

SMTP 接続の問題については、「SMTP に関する問題」を参照してください。

STARTTLS

STARTTLS とは、暗号化されていない接続を暗号化された接続にアップグレードする方法です。STARTTLS には、様々なプロトコルに対応したバージョンがあります。SMTP バージョンは、RFC 3207で定義されています。

STARTTLS 接続を設定する場合、SMTP クライアントは、ポート 25、587、または 2587 で Amazon SES SMTP エンドポイントに接続し、EHLO コマンドを発行します。次に、サーバーから STARTTLS SMTP 拡張機能をサポートしているという通知が来るのを待ちます。通知を受けたクライアントは、STARTTLS コマンドを発行し、TLS ネゴシエーションを開始します。ネゴシエーションが完了すると、クライアントが暗号化された新しい接続で EHLO コマンドを発行し、SMTP セッションが正常に進行します。

TLS ラッパー

TLS ラッパー (SMTPS またはハンドシェイクプロトコルとも呼ばれる) は、最初に暗号化されていない接続を確立するのではなく、最初から暗号化された接続を開始する方法です。TLS ラッパーを使用する場合、Amazon SES SMTP エンドポイントは TLS ネゴシエーションを実行しません。TLS を使用してエンドポイントに接続し、通信全体で TLS の使用を継続するのはクライアントの役割です。TLS ラッパーは古いプロトコルですが、数多くのクライアントが今もサポートしています。

TLS ラッパー接続を設定する場合、SMTP クライアントは、Amazon SES SMTP エンドポイントにポート 465 または 2465 で接続します。サーバーが自身の証明書を提示すると、クライアントが EHLO コマンドを発行し、SMTP セッションが正常に進行します。

ソフトウェアパッケージを使用し、Amazon SES を介して E メールを送信します

SMTP を介した E メール送信に対応している市販とオープンソースのソフトウェアパッケージは多数あります。次に例を示します。

- ブログプラットフォーム
- RSS アグリゲータ
- リスト管理ソフトウェア
- ワークフローシステム

上記の SMTP 対応ソフトウェアは、Amazon SES SMTP インターフェイスを介して E メールを送信するように設定できます。個々のソフトウェアパッケージの SMTP 設定手順については、そのソフトウェアのドキュメントを参照ください。

次の手順は、一般的な問題追跡ソリューションである JIRA で Amazon SES 送信を設定する方法を示しています。この設定により、ソフトウェアの問題のステータスが変化したときに JIRA からユーザーに E メールで通知できるようになります。

Amazon SES を介して E メールを送信するために、JIRA を設定するには

1. ウェブブラウザーを使用し、管理者認証情報で JIRA にログインします。
2. ブラウザのウィンドウで、[Administration] を選択します。
3. システムメニューで、メールを選択します。
4. Mail administration ページで、Mail Servers を選択します。
5. Configure new SMTP mail server を選択します。
6. Add SMTP Mail Server フォームで、次のフィールドに入力します。

    a. 名前 - このサーバーの記述名。

    b. From address – E メールの送信元アドレス。それから送信する前に、Amazon SES でこの E メールアドレスを検証する必要があります。検証の詳細については、Amazon SES の検証済みIDを参照ください。

    c. Email prefix – 送信前に、件名行に JIRA が付加する文字列。

    d. Protocol - SMTP を選択します。

    > **Note**
    >
    > この設定を使用して Amazon SES に接続できない場合は、SECURE_SMTP を試してください。

    e. ホスト名 - Amazon SES SMTP エンドポイントのリストについては、Amazon SES SMTP エンドポイントへの接続を参照ください。たとえば、米国西部 (オレゴン) 地域で Amazon SES エンドポイントを使用する場合、ホスト名はemail-smtp.us-west-2.amazonaws.com です。

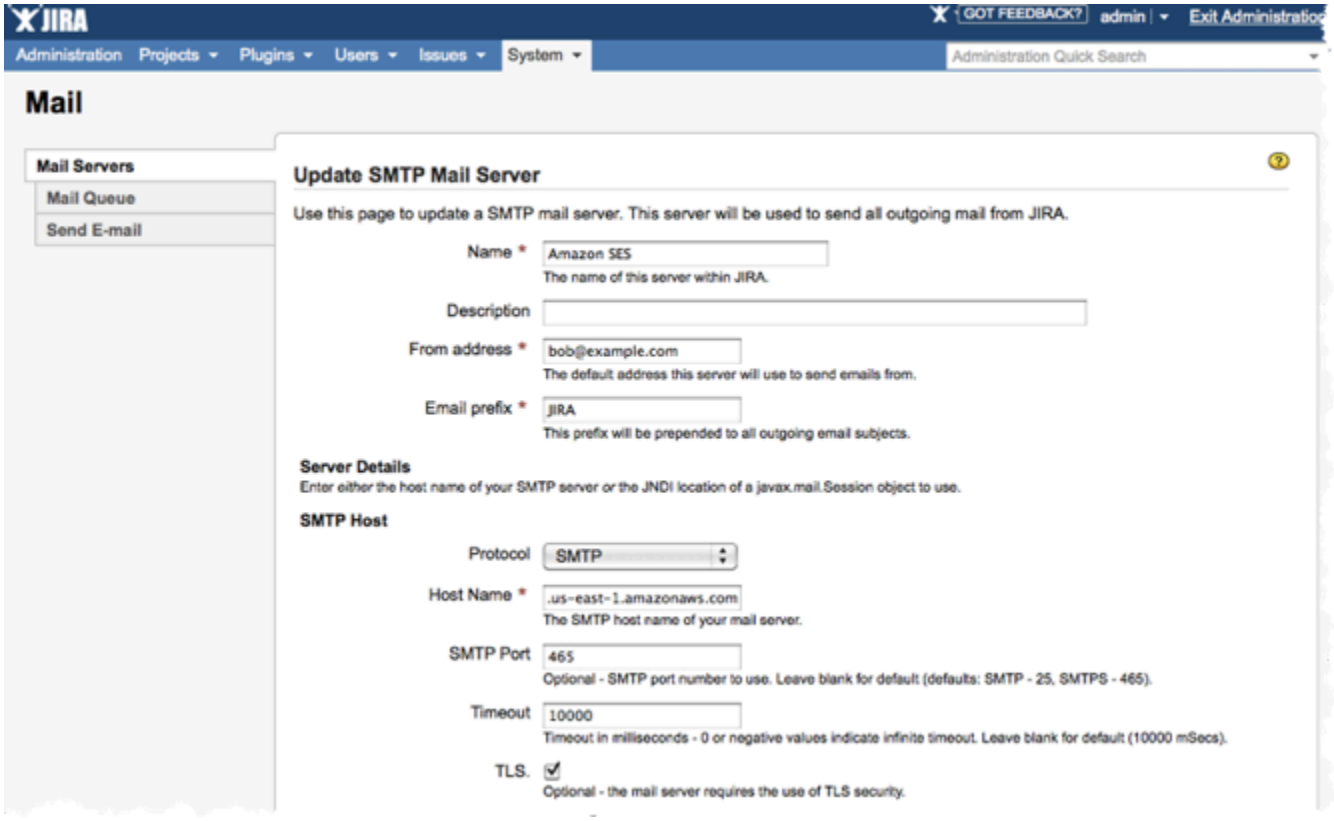
    f. SMTP Port - 25、587、2587 のいずれか (STARTTLS を使用して接続する場合)、または 465、2465 のいずれか (TLS ラッパーを使用して接続する場合)。

    g. TLS – このチェックボックスを選択します。

h. User Name - SMTP の ユーザー名。

i. Password - SMTP パスワード。

以下の画像に TLS ラッパーの設定が表示できます。



7. 接続のテストを選択します。Amazon SES を介して JIRA が送信したテスト E メールが正しく到着すれば、設定は完了です。

プログラミングで Amazon SES の SMTP インターフェイスを介して E メールを送信する

Amazon SES の SMTP インターフェイスで E メールを送信するには、SMTP 対応のプログラミング言語、E メールサーバー、またはアプリケーションを使用できます。開始する前に、Amazon Simple Email Service を設定するのタスクを完了します。また、以下の詳細を取得する必要があります。

- Amazon SES SMTP エンドポイントに接続するための Amazon SES SMTP 認証情報。Amazon SES SMTP 認証情報を取得するには、「Amazon SES SMTP 認証情報を取得」を参照してください。

> **Important**
>
> SMTP 認証情報は、 AWS 認証情報とは異なります。認証情報の詳細については、Amazon SES 認証情報の種類を参照ください。

- SMTP エンドポイントアドレス。Amazon SES における SMTP エンドポイントのリストについては、Amazon SES SMTP エンドポイントへの接続 を参照ください。
- Amazon SES SMTP インターフェイ接続端子番号。これは接続方法によって異なります。詳細については、Amazon SES SMTP エンドポイントへの接続を参照ください。

コードの例

SMTP 対応のプログラミング言語を使用して、Amazon SES の SMTP インターフェイスにアクセスできます。SMTP 認証情報と Amazon SES SMTP ホスト名およびポート番号を使用して、プログラミング言語の 一般的な SMTP 機能によって E メールを送信します。

Amazon Elastic Compute Cloud (Amazon EC2) では、デフォルトでポート 25 経由での E メールトラフィックを制限しています。Amazon EC2 から SMTP エンドポイントを介して E メールを送信する際のタイムアウトを回避するには、これらの制限を解除するようリクエストすることができます。詳細については、「 AWS ナレッジセンター」のAmazon EC2 インスタンスまたは AWS Lambda 関数からポート 25 の制限を削除する方法」を参照してください。

Java および PHP のこのセクションのコード例で、この問題を回避するためにポート 587 を使用します。

> **Note**
>
> このチュートリアルでは、受信を確認できるように自分宛に E メールを送信します。さらに詳しい実験や負荷テストには、Amazon SES メールボックスシミュレーターを使用してください。メールボックスシミュレーターに送信される E メールは、送信クォータに加算されず、バウンス率や苦情率の計算にも含まれません。詳細については、手動でメールボックスシミュレーターを使用するを参照ください。

プログラミング言語を選択して、その言語の例を表示します。

> **Warning**
>
> Amazon SES では、静的認証情報の使用はお勧めしません。ソースコードからハードコードされた認証情報を削除してセキュリティ体制を改善する方法については、「AWS Secrets Manager」を参照してください。このチュートリアルは、本番環境以外の環境で Amazon SES SMTP インターフェイスをテストする目的でのみ提供されています。

Java

この例では、Eclipse IDEとJavaMail APIを使用して、SMTP インターフェイスで Amazon SES から E メールを送信します。

以下の手順を実行する前に、Amazon Simple Email Service を設定するに記載されている作業を完了します。

Java で Amazon SES SMTP インターフェイスを使用して E メールを送信するには

1. ウェブブラウザで、JavaMail GitHub ページに移動します。[Assets] の下で、[javax.mail.jar] を選択して、JavaMail の最新バージョンをダウンロードします。

   > **Important**
   >
   > このチュートリアルには、JavaMail バージョン 1.5 以降が必要です。これらの手順は、JavaMail バージョン 1.6.1 を使用してテスト済みです。

2. ウェブブラウザで Jakarta Activation GitHub ページに移動して、[JavaBeans Activation Framework 1.2.1 Final Release] の下で、jakarta.activation.jar をダウンロードします。
3. 次のステップを実行し、Eclipse でプロジェクトを作成します。

   a. Eclipse を起動します。
   b. Eclipse で、ファイルを選択し、新規、Java Project の順に選択します。
   c. Create a Java Project ダイアログボックスで、プロジェクト名を入力し、次へを選択します。
   d. Java Settings ダイアログボックスのライブラリタブを選択します。
   e. [Classpath] を選択し、[Add External JARs] ボタンを使用して、javax.mail.jar と jakarta.activation.jar の 2 つの外部 jar ファイルを追加します。
   f. Add External JARs を選択します。

g. JavaMail をダウンロードした先のフォルダを参照します。javax.mail.jar ファイルを選択し、開くを選択します。

h. Java Settings ダイアログボックスの終了を選択します。

4. Eclipse で、Package Explorerウィンドウのプロジェクトを展開します。

5. プロジェクトの下の src ディレクトリを右クリックし、新規、クラスの順に選択します。

6. New Java Class ダイアログボックスの名前フィールドに AmazonSESSample と入力し、終了を選択します。

7. AmazonSESSample.java のコンテンツ全体を以下のコードに置き換えます。

```
import java.util.Properties;

import javax.mail.Message;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class AmazonSESSample {

    // Replace sender@example.com with your "From" address.
    // This address must be verified.
    static final String FROM = "sender@example.com";
    static final String FROMNAME = "Sender Name";

    // Replace recipient@example.com with a "To" address. If your account
    // is still in the sandbox, this address must be verified.
    static final String TO = "recipient@example.com";

    // Replace smtp_username with your Amazon SES SMTP user name.
    static final String SMTP_USERNAME = "smtp_username";

    // The name of the Configuration Set to use for this message.
    // If you comment out or remove this variable, you will also need to
    // comment out or remove the header below.
    static final String CONFIGSET = "ConfigSet";

    // Amazon SES SMTP host name. This example uses the US West (Oregon) region.
    // See https://docs.aws.amazon.com/ses/latest/DeveloperGuide/
regions.html#region-endpoints
    // for more information.
    static final String HOST = "email-smtp.us-west-2.amazonaws.com";
```

```
   // The port you will connect to on the Amazon SES SMTP endpoint.
   static final int PORT = 587;

   static final String SUBJECT = "Amazon SES test (SMTP interface accessed
using Java)";

   static final String BODY = String.join(
           System.getProperty("line.separator"),
           "<h1>Amazon SES SMTP Email Test</h1>",
           "<p>This email was sent with Amazon SES using the ",
           "<a href='https://github.com/javaee/javamail'>Javamail Package</a>",
           " for <a href='https://www.java.com'>Java</a>."
       );

   public static void main(String[] args) throws Exception {

       // Create a Properties object to contain connection configuration
information.
       Properties props = System.getProperties();
       props.put("mail.transport.protocol", "smtp");
       props.put("mail.smtp.port", PORT);
       props.put("mail.smtp.starttls.enable", "true");
       props.put("mail.smtp.auth", "true");

       // Create a Session object to represent a mail session with the
specified properties.
       Session session = Session.getDefaultInstance(props);

       // Create a message with the specified information.
       MimeMessage msg = new MimeMessage(session);
       msg.setFrom(new InternetAddress(FROM,FROMNAME));
       msg.setRecipient(Message.RecipientType.TO, new InternetAddress(TO));
       msg.setSubject(SUBJECT);
       msg.setContent(BODY,"text/html");

       // Add a configuration set header. Comment or delete the
       // next line if you are not using a configuration set
       msg.setHeader("X-SES-CONFIGURATION-SET", CONFIGSET);

       // Create a transport.
       Transport transport = session.getTransport();

       // Get the password
```

```
        String SMTP_PASSWORD = fetchSMTPPasswordFromSecureStorage();

        // Send the message.
        try
        {
            System.out.println("Sending...");

            // Connect to Amazon SES using the SMTP username and password you
 specified above.
            transport.connect(HOST, SMTP_USERNAME, SMTP_PASSWORD);

            // Send the email.
            transport.sendMessage(msg, msg.getAllRecipients());
            System.out.println("Email sent!");
        }
        catch (Exception ex) {
            System.out.println("The email was not sent.");
            System.out.println("Error message: " + ex.getMessage());
        }
        finally
        {
            // Close and terminate the connection.
            transport.close();
        }
    }

    static String fetchSMTPPasswordFromSecureStorage() {
        /* IMPLEMENT THIS METHOD */
        // For example, you might fetch it from a secure location or AWS Secrets
 Manager: https://aws.amazon.com/secrets-manager/
    }
}
```

8. AmazonSESSample.java で、以下のメールアドレスを実際の値に置き換えます。

> **Important**
>
> E メールアドレスでは、大文字と小文字は区別されます。検証したアドレスと完全に一致することを確認してください。

- *sender@example.com* – 「送信元」メールアドレスに置き換えます。このアドレスを確認してから、プログラムを実行してください。詳細については、「Amazon SES の検証済みID」を参照してください。
- *recipient@example.com* – 「送信先」メールアドレスに置き換えます。アカウントがサンドボックスにまだある場合は、このアドレスを使用前に確認する必要があります。詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。

9. AmazonSESSample.java で、以下を実際の値に置き換えます。

   - *smtp_username* – SMTP ユーザー名の認証情報に置き換えます。SMTP ユーザー名の認証情報は 20 文字の文字と数字の並びであり、意味のある名前ではありません。
   - *smtp_password* – パスワードを取得するために `fetchSMTPPasswordFromSecureStorage` を実装します。

10. (オプション) *email-smtp.us-west-2.amazonaws.com* 以外の AWS リージョン Amazon SES SMTP エンドポイントを使用する場合は、HOST 変数の値を実際に使用するエンドポイントに変更します。Amazon SES を使用できるリージョンのリストについては、「AWS 全般のリファレンス」の「Amazon Simple Email Service (Amazon SES)」を参照してください。
11. (オプション) この E メール送信時に設定セットを使用する場合は、*ConfigSet* 変数の値をその設定セット名に変更します。設定セットの詳細については、Amazon SES の設定セットの使用を参照ください。
12. AmazonSESSample.java を保存します。
13. プロジェクトを構築するため、プロジェクト、プロジェクトの構築の順に選択します。(このオプションが無効の場合、自動構築が有効になっている可能性があります。)
14. プログラムを開始して E メールを送信するため、実行を選択した後、もう一度実行を選択します。
15. 出力を確認します。E メールの送信が正常に完了すると、コンソールに「確認の E メールが送信されました」と表示されます。送信が失敗した場合は、エラーメッセージが表示されます。
16. 受信者のアドレスの E メールクライアントにサインインします。送信した E メールメッセージを確認します。

PHP

この例では PHPMailer クラスを使用し、SMTP インターフェイスで Amazon SES を介して E メールを送信します。

以下の手順を実行する前に、Amazon Simple Email Service を設定するのタスクを完了する必要があります。Amazon SES のセットアップに加えて、PHP で E メールを送信するには、次の前提条件を満たしている必要があります。

前提条件:

- PHP のインストール – PHP は、http://php.net/downloads.php で入手できます。PHP をインストールした後、コマンドプロンプトから PHP を実行できるように環境変数に PHP のパスを追加します。
- Composer 依存関係管理システムのインストール – Composer 依存関係管理システムのインストール後、PHPMailer クラスと依存関係をダウンロードしてインストールできます。Composer をインストールするには、https://getcomposer.org/download のインストール手順を参照してください。
- PHPMailer クラスのインストール – Composer のインストール後、以下のコマンドを実行して PHPMailer をインストールします。

```
path/to/composer require phpmailer/phpmailer
```

前のコマンドで、*path/to/*を、Composer をインストールしたパスに置き換えます。

PHP で Amazon SES の SMTP インターフェイスを使用して E メールを送信するには

1. amazon-ses-smtp-sample.php という名前のファイルを作成します。テキストエディタでファイルを開き、次のコードを貼り付けます。

```
<?php

// Import PHPMailer classes into the global namespace
// These must be at the top of your script, not inside a function
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

// If necessary, modify the path in the require statement below to refer to the
// location of your Composer autoload.php file.
```

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

// Replace sender@example.com with your "From" address.
// This address must be verified with Amazon SES.
$sender = 'sender@example.com';
$senderName = 'Sender Name';

// Replace recipient@example.com with a "To" address. If your account
// is still in the sandbox, this address must be verified.
$recipient = 'recipient@example.com';

// Replace smtp_username with your Amazon SES SMTP user name.
$usernameSmtp = 'smtp_username';

// Specify a configuration set. If you do not want to use a configuration
// set, comment or remove the next line.
$configurationSet = 'ConfigSet';

// If you're using Amazon SES in a region other than US West (Oregon),
// replace email-smtp.us-west-2.amazonaws.com with the Amazon SES SMTP
// endpoint in the appropriate region.
$host = 'email-smtp.us-west-2.amazonaws.com';
$port = 587;

// The subject line of the email
$subject = 'Amazon SES test (SMTP interface accessed using PHP)';

// The plain-text body of the email
$bodyText =  "Email Test\r\nThis email was sent through the
    Amazon SES SMTP interface using the PHPMailer class.";

// The HTML-formatted body of the email
$bodyHtml = '<h1>Email Test</h1>
    <p>This email was sent through the
    <a href="https://aws.amazon.com/ses">Amazon SES</a> SMTP
    interface using the <a href="https://github.com/PHPMailer/PHPMailer">
    PHPMailer</a> class.</p>';

$mail = new PHPMailer(true);

try {
    // Specify the SMTP settings.
    $mail->isSMTP();
    $mail->setFrom($sender, $senderName);
```

```
    $mail->Username   = $usernameSmtp;
    $mail->Password   = fetchSMTPPasswordFromSecureStorage();
    $mail->Host       = $host;
    $mail->Port       = $port;
    $mail->SMTPAuth   = true;
    $mail->SMTPSecure = 'tls';
    $mail->addCustomHeader('X-SES-CONFIGURATION-SET', $configurationSet);

    // Specify the message recipients.
    $mail->addAddress($recipient);
    // You can also add CC, BCC, and additional To recipients here.

    // Specify the content of the message.
    $mail->isHTML(true);
    $mail->Subject    = $subject;
    $mail->Body       = $bodyHtml;
    $mail->AltBody    = $bodyText;
    $mail->Send();
    echo "Email sent!" , PHP_EOL;
} catch (phpmailerException $e) {
    echo "An error occurred. {$e->errorMessage()}", PHP_EOL; //Catch errors from
 PHPMailer.
} catch (Exception $e) {
    echo "Email not sent. {$mail->ErrorInfo}", PHP_EOL; //Catch errors from
 Amazon SES.
}
function fetchSMTPPasswordFromSecureStorage() {
/* IMPLEMENT THIS METHOD */
// For example, you might fetch it from a secure location or AWS Secrets
 Manager: https://aws.amazon.com/secrets-manager/
}

?>
```

2. amazon-ses-smtp-sample.php で、以下を実際の値に置き換えます。

   - *sender@example.com* – Amazon SES で検証した E メールアドレスに置き換えます。詳細については、「検証済みID」を参照してください。Amazon SES では、E メールアドレスの大文字と小文字が区別されます。検証したアドレスと完全に一致するアドレスを入力してください。
   - *recipient@example.com* – 送信先アドレスに置き換えます。アカウントがサンドボックスにまだある場合は、このアドレスを使用前に確認する必要があります。詳細について

は、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。検証したアドレスと完全に一致するアドレスを入力してください。

- *smtp_username* – SMTP ユーザー名の認証情報に置き換えます。これは、Amazon SES コンソールの [SMTP 設定] ページで取得したものです。これは AWS アクセスキー ID とは異なります。SMTP ユーザー名の認証情報は 20 文字の文字と数字の並びであり、意味のある名前ではありません。
- *smtp_password* – パスワードを取得するために `fetchSMTPPasswordFromSecureStorage` を実装します。
- (オプション) *ConfigSet* – この E メール送信時に設定セットを使用する場合は、この値を設定セット名に置き換えます。設定セットの詳細については、Amazon SES の設定セットの使用を参照ください。
- (オプション) *email-smtp.us-west-2.amazonaws.com* – 米国西部 (オレゴン) 以外のリージョンの Amazon SES SMTP エンドポイントを使用する場合は、この値を使用するリージョンの Amazon SES SMTP エンドポイントに置き換えます。 AWS リージョン Amazon SES が利用可能な の SMTP エンドポイント URLs「」の「Amazon Simple Email Service (Amazon SES)」を参照してくださいAWS 全般のリファレンス。

3. amazon-ses-smtp-sample.php を保存します。
4. このプログラムを実行するには、amazon-ses-smtp-sample.php と同じディレクトリでコマンドプロンプトを開いて、php amazon-ses-smtp-sample.php と入力します。
5. 出力を確認します。E メールの送信が正常に完了すると、コンソールに「確認の E メールが送信されました」と表示されます。送信が失敗した場合は、エラーメッセージが表示されます。
6. 受信者のアドレスの E メールクライアントにサインインします。送信したメッセージを確認します。

Amazon SES を既存の E メールサーバーと統合します

現在、E メールサーバーをお客様が管理している場合には、Amazon SES SMTP エンドポイントを使用して、外部へのメールをすべて Amazon SES に送信できます。既存の E メールクライアントや E メールアプリケーションに変更を加える必要はありません。Amazon SES への変更は、これらに対して透過的です。

メール転送エージェント (MTA) の中には、SMTP リレーを介した E メールの送信をサポートしているものもあります。このセクションでは、一般的ないくつかの MTA について、Amazon SES

SMTP インターフェイスを使用して E メールを送信するための設定方法について、一般的なガイダンスを提供します。

Amazon SES SMTP エンドポイントでは、すべての接続が Transport Layer Security（TLS）を使用して暗号化されている必要があります。

トピック

- Amazon SES と Postfix の統合
- Amazon SES と Sendmail の統合
- Amazon SES を Microsoft Windows Server IIS SMTP に統合する

Amazon SES と Postfix の統合

Postfix は、広く使用されている Sendmail Message Transfer Agent (MTA) に代わる手段です。Postfix の詳細については、http://www.postfix.org を参照ください。このトピックの手順では、Linux、macOS、または Unix で動作します。

> **Note**
>
> Postfix はサードパーティー製アプリケーションであり、Amazon Web Services によって開発またはサポートされていません。このセクションの手順は情報提供のみを目的としており、予告なく変更される場合があります。

前提条件

このセクションの手順を完了する前に、以下のタスクを実行する必要があります。

- システムに Sendmail アプリケーションがインストール済みである場合、これをアンインストールします。このステップを完了する手順は使用するオペレーティングシステムによって異なります。

  > **Important**
  >
  > sendmail への参照の後で Postfix コマンド `sendmail` を参照し、Sendmail アプリケーションと混同しないようにします。

- Postfix をインストールします。このステップを完了する手順は使用するオペレーティングシステムによって異なります。

- SASL 認証パッケージをインストールします。このステップを完了する手順は使用するオペレーティングシステムによって異なります。たとえば、RedHat ベースのシステムを使用している場合は、`cyrus-sasl-plain` パッケージをインストールする必要があります。Debian または Ubuntu ベースのシステムを使用している場合は、`libsasl2-modules` パッケージをインストールする必要があります。
- E メールの送信に使用する E メールアドレスまたはドメインを確認します。詳細については、E メールアドレス ID の作成を参照ください。
- アカウントがまだサンドボックスにある場合、検証済み E メールアドレスにのみ E メールを送信できます。詳細については、本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)を参照ください。

Postfix の設定

Postfix を使用して Amazon SES 経由で E メールを送信するようにメールサーバーを設定するには、次の手順を完了します。

Postfix を設定するには

1. コマンドラインから、以下のコマンドを入力します。

```
sudo postconf -e "relayhost = [email-smtp.us-west-2.amazonaws.com]:587" \
"smtp_sasl_auth_enable = yes" \
"smtp_sasl_security_options = noanonymous" \
"smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd" \
"smtp_use_tls = yes" \
"smtp_tls_security_level = secure" \
"smtp_tls_note_starttls_offer = yes"
```

> **Note**
>
> 米国西部 (オレゴン) 以外の AWS リージョンで Amazon SES を使用する場合は、前述のコマンドの *email-smtp.us-west-2.amazonaws.com* を適切なリージョンの SMTP エンドポイントに置き換えます。詳細については、「the section called “リージョン”」を参照してください。

2. テキストエディタで、`/etc/postfix/master.cf` ファイルを開きます。次のエントリを検索します。

```
-o smtp_fallback_relay=
```

このエントリが見つかった場合は、行の先頭に # (ハッシュ) 文字を配置してコメントアウトします。保存してファイルを閉じます。

それ以外で、このエントリが存在しない場合、次の手順に進みます。

3. テキストエディタで、/etc/postfix/sasl_passwd ファイルを開きます。このファイルが存在していない場合は、作成します。
4. /etc/postfix/sasl_passwd に次の行を追加します。

```
[email-smtp.us-west-2.amazonaws.com]:587 SMTPUSERNAME:SMTPPASSWORD
```

> **Note**
>
> *SMTPUSERNAME* と *SMTPPASSWORD* を SMTP サインイン認証情報に置き換えてください。SMTP サインイン認証情報は、 AWS アクセスキー ID やシークレットアクセスキーと同じではありません。認証情報の詳細については、「the section called “SMTP 認証情報の取得”」を参照してください。
> 米国西部 (オレゴン) 以外の AWS リージョンで Amazon SES を使用する場合は、前の例の *email-smtp.us-west-2.amazonaws.com* を適切なリージョンの SMTP エンドポイントに置き換えます。詳細については、「the section called “リージョン”」を参照してください。

sasl_passwd を保存して閉じます。

5. コマンドプロンプトで次のコマンドを入力し、SMTP 認証情報を含む hashmap データベースファイルを作成します。

```
sudo postmap hash:/etc/postfix/sasl_passwd
```

6. (オプション) 前の手順で作成した/etc/postfix/sasl_passwd および /etc/postfix/sasl_passwd.db ファイルは暗号化されていません。これらのファイルには SMTP 認証情報が含まれているため、ファイルへのアクセスを制限するために、ファイルの所有権とアクセス許可を変更することをお勧めします。これらのファイルへのアクセスを制限するには。

   a. コマンドプロンプトで次のコマンドを入力して、ファイルの所有権を変更します。

```
sudo chown root:root /etc/postfix/sasl_passwd /etc/postfix/sasl_passwd.db
```

b. コマンドプロンプトで次のコマンドを入力してファイルのアクセス許可を変更して、ルートユーザーのみがこれらのファイルで読み込みや書き込みを実行できるようにします。

```
sudo chmod 0600 /etc/postfix/sasl_passwd /etc/postfix/sasl_passwd.db
```

7. Postfix が CA 証明書の場所を認識できるようにします (Amazon SES サーバー証明書を検証するために必要です)。このステップで使用するコマンドは、オペレーティングシステムによって異なります。

   - Amazon Linux、Red Hat Enterprise Linux、あるいは関連するディストリビューションを使用している場合には、次のコマンドを入力します。

     ```
     sudo postconf -e 'smtp_tls_CAfile = /etc/ssl/certs/ca-bundle.crt'
     ```

   - Ubuntu あるいは関連するディストリビューションを使用している場合には、次のコマンドを入力します。

     ```
     sudo postconf -e 'smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt'
     ```

   - macOS を使用する場合、システムのキーチェーンで証明書を生成できます。証明書を生成するには、コマンドラインで次のコマンドを入力します。

     ```
     sudo security find-certificate -a -p /System/Library/Keychains/
     SystemRootCertificates.keychain > /etc/ssl/certs/ca-bundle.crt
     ```

     証明書を生成したら、次のコマンドを入力します。

     ```
     sudo postconf -e 'smtp_tls_CAfile = /etc/ssl/certs/ca-bundle.crt'
     ```

8. 次のコマンドを入力して Postfix サーバーを起動します (サーバーがすでに起動している場合は、設定を再ロードしてください)。

   ```
   sudo postfix start; sudo postfix reload
   ```

9. コマンドラインで以下のように入力し、各行の最後で Enter キーを押して、テスト E メールを送信します。*sender@example.com* を送信元 E メールアドレスに置き換えます。送信元アドレスの Amazon SES での使用を確認する必要があります。*recipient@example.com* を送信

先アドレスに置き換えます。アカウントがサンドボックスにまだある場合は、受信者アドレスも確認する必要があります。最後に、メッセージの最後の行には 1 つのピリオドが (.) 含まれていることが必要です (他に何も含まない)。

```
sendmail -f sender@example.com recipient@example.com
From: Sender Name <sender@example.com>
Subject: Amazon SES Test
This message was sent using Amazon SES.
.
```

10. 受信者のアドレスに関連付けられているメールボックスを確認します。E メールが届かない場合は、迷惑メールフォルダを確認します。それでも E メールが見つからない場合には、E メールを送信するために使用したシステムのメールログで詳細を確認します (通常は、/var/log/maillog にあります)。

高度な使用例

この例は、設定セットを使用する E メール、および MIME マルチパートエンコードを使用してプレーンテキストと HTML バージョンのメッセージの両方を添付ファイルと共に送る E メールを送信する方法を示します。また、それにはクリックイベントの分類に利用できるリンクタグが含まれています。E メールのコンテンツは外部ファイルで指定されるため、Postfix セッションでは手動でコマンドを入力する必要はありません。

Postfix を使用してマルチパートの MIME E メールを送信するには

1. テキストエディタで、mime-email.txt という名前の新規ファイルを作成します。
2. テキストファイルで次のコンテンツを貼り付け、赤の値を自分のアカウントの適切な値に置き換えます。

```
X-SES-CONFIGURATION-SET: ConfigSet
From:Sender Name <sender@example.com>
Subject:Amazon SES Test
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="YWVhZDFlY2QzMGQ2N2U0YTZmODU"

--YWVhZDFlY2QzMGQ2N2U0YTZmODU
Content-Type: multipart/alternative; boundary="3NjM0N2QwMTE4MWQ0ZTg2NTYxZQ"

--3NjM0N2QwMTE4MWQ0ZTg2NTYxZQ
Content-Type: text/plain; charset=UTF-8
```

```
Content-Transfer-Encoding: quoted-printable

Amazon SES Test

This message was sent from Amazon SES using the SMTP interface.

For more information, see:
http://docs.aws.amazon.com/ses/latest/DeveloperGuide/send-email-smtp.html

--3NjM0N2QwMTE4MWQ0ZTg2NTYxZQ
Content-Type: text/html; charset=UTF-8
Content-Transfer-Encoding: quoted-printable

<html>
  <head>
</head>
  <body>
    <h1>Amazon SES Test</h1>
      <p>This message was sent from Amazon SES using the SMTP interface.</p>
      <p>For more information, see
      <a ses:tags="samplekey0:samplevalue0;samplekey1:samplevalue1;"
      href="http://docs.aws.amazon.com/ses/latest/DeveloperGuide/send-email-
smtp.html">
      Using the Amazon SES SMTP Interface to Send Email</a> in the <em>Amazon SES
      Developer Guide</em>.</p>
  </body>
</html>
--3NjM0N2QwMTE4MWQ0ZTg2NTYxZQ--
--YWVhZDFlY2QzMGQ2N2U0YTZmODU
Content-Type: application/octet-stream
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="customers.txt"

SUQsRmlyc3ROYW1lLExhc3ROYW1lLENvdW50cnkKMzQ4LEpvaG4sU3RpbGVzLENh
bmFkYQo5MjM4OSxKaWUsTGl1LENoaW5hCjczNCxTaGlybGV5LFJvZHJpZ3VleixV
bml0ZWQgU3RhdGVzCjI4OTMsQW5heWEsSXllbmdhcixJbmRpYQ==
--YWVhZDFlY2QzMGQ2N2U0YTZmODU--
```

保存してファイルを閉じます。

3. コマンドラインから、以下のコマンドを入力します。*sender@example.com* をユーザーの E メールアドレスに、*recipient@example.com* を受信者の E メールアドレスに置き換えます。

```
sendmail -f sender@example.com recipient@example.com < mime-email.txt
```

   コマンドが正常に実行された場合、何の出力もなく終了します。

4. E メールの受信箱を確認します。メッセージが配信されなかった場合は、システムのメールログを確認します。

Amazon SES と Sendmail の統合

Sendmail は 1980 年代の初めにリリースされ、それ以降、継続的に改善されてきました。これは、大規模なユーザーコミュニティを持つ柔軟で設定可能な Message Transfer Agent (MTA) です。Sendmail は 2013 年に Proofpoint により買収されましたが、Proofpoint は Sendmail のオープンソースバージョンの提供を続けています。Sendmail の open source バージョンは、Proofpoint のウェブサイトからダウンロードするか、ほとんどの Linux ディストリビューションのパッケージマネージャを経由してダウンロードできます。

このセクションの手順は、Amazon SES を介して E メールを送信するよう Sendmail を設定する方法を示しています。この手順は Ubuntu 18.04.2 LTS を実行しているサーバーでテストされています。

> **Note**
>
> Sendmail はサードパーティーのアプリケーションであり、Amazon Web Services によって開発またはサポートされていません。このセクションの手順は情報提供のみを目的としており、予告なく変更される場合があります。

前提条件

このセクションの手順を完了する前に、以下の手順を完了する必要があります。

- Sendmail パッケージをサーバーにインストールします。

> **Note**
>
> 使用しているオペレーティングシステムのディストリビューションに応じて、sendmail-cf、m4、および cyrus-sasl-plain のパッケージのインストールが必要になる場合があります。

- 差出人アドレスとして使用する ID を確認します。詳細については、「Eメールアドレス ID の作成」を参照してください。

  アカウントが Amazon SES サンドボックスにある場合は、E メールの送信先のアドレスも検証する必要があります。詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。

Amazon SES を使用して Amazon EC2 インスタンスから E メールを送信する場合は、次の手順も実行する必要があります。

- 受信する E メールプロバイダーが E メールを受信するように、Elastic IP アドレスを Amazon EC2 インスタンスに割り当てる必要があります。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 Elastic IP アドレス」を参照してください。
- Amazon Elastic Compute Cloud (Amazon EC2) では、デフォルトでポート 25 経由での E メールトラフィックを制限しています。Amazon EC2 から SMTP エンドポイントを介して E メールを送信する際のタイムアウトを回避するには、これらの制限を解除するようリクエストすることができます。詳細については、「 AWS ナレッジセンター」のAmazon EC2 インスタンスまたは AWS Lambda 関数からポート 25 の制限を削除する方法」を参照してください。

  または、このセクションの手順を変更して、ポート 25 ではなく、ポート 587 を使用することもできます。

Sendmail を設定する

このセクションの手順に従って、Amazon SES を使用して E メールを送信するように Sendmail を設定します。

> **Important**
>
> このセクションの手順では、米国西部 (オレゴン) で Amazon SES を使用することを前提としています AWS リージョン。別の地域を使用する場合は、この手順の email-smtp.us-

west-2.amazonaws.com のすべてのインスタンスをご希望地域の SMTP エンドポイントに置き換えてください。Amazon SES が使用可能な AWS リージョン の SMTP エンドポイント URL のリストについては、「AWS 全般のリファレンス」の「Amazon Simple Email Service (Amazon SES)」を参照してください。

Sendmail を設定するには

1. ファイルエディタで、ファイル /etc/mail/authinfo を開きます。このファイルが存在しない場合は、作成します。

   次の行を /etc/mail/authinfo に追加します。

   ```
   AuthInfo:email-smtp.us-west-2.amazonaws.com "U:root" "I:smtpUsername"
    "P:smtpPassword" "M:PLAIN"
   ```

   上の例に、以下の変更を加えます。

   - *email-smtp.us-west-2.amazonaws.com*を、使用する Amazon SES SMTP エンドポイントに置き換えます。
   - *smtpUsername* を Amazon SES SMTP ユーザー名に置き換えます。
   - *smtpPassword*を Amazon SES SMTP パスワードに置き換えます。

   > **Note**
   >
   > SMTP サインイン認証情報は、 AWS アクセスキー ID およびシークレットアクセスキーとは異なります。SMTP サインイン認証情報を取得する方法の詳細については、「Amazon SES SMTP 認証情報を取得」を参照してください。

   終了したら、authinfoを保存します。

2. コマンドラインで、以下のコマンドを入力して/etc/mail/authinfo.db ファイルを生成します。

   ```
   sudo sh -c 'makemap hash /etc/mail/authinfo.db < /etc/mail/authinfo'
   ```

3. コマンドラインで、以下のコマンドを入力して Amazon SES SMTP エンドポイントに中継するためのサポートを追加します。

```
sudo sh -c 'echo "Connect:email-smtp.us-west-2.amazonaws.com RELAY" >> /etc/mail/
access'
```

上のコマンドで、*email-smtp.us-west-2.amazonaws.com* を、使用する Amazon SES SMTP エンドポイントのアドレスに置き換えます。

4. コマンドラインで、次のコマンドを入力して /etc/mail/access.db を再生成します。

```
sudo sh -c 'makemap hash /etc/mail/access.db < /etc/mail/access'
```

5. コマンドラインで、次のコマンドを入力して sendmail.cf ファイルと sendmail.mc ファイルのバックアップを作成します。

```
sudo sh -c 'cp /etc/mail/sendmail.cf /etc/mail/sendmail_cf.backup && cp /etc/mail/
sendmail.mc /etc/mail/sendmail_mc.backup'
```

6. 以下の行を、/etc/mail/sendmail.mc ファイルのすべての MAILER() 定義の前に追加します。

```
define(`SMART_HOST', `email-smtp.us-west-2.amazonaws.com')dnl
define(`RELAY_MAILER_ARGS', `TCP $h 25')dnl
define(`confAUTH_MECHANISMS', `LOGIN PLAIN')dnl
FEATURE(`authinfo', `hash -o /etc/mail/authinfo.db')dnl
MASQUERADE_AS(`example.com')dnl
FEATURE(masquerade_envelope)dnl
FEATURE(masquerade_entire_domain)dnl
```

上のテキストで、次の操作を行います。

- *email-smtp.us-west-2.amazonaws.com*を、使用する Amazon SES SMTP エンドポイントに置き換えます。
- *example.com* を、E メールの送信に使用するドメインに置き換えます。

終了したら、ファイルを保存します。

> **Note**
>
> Amazon EC2 は、デフォルトでポート 25 経由の通信を制限します。Amazon EC2 インスタンスの Sendmail を使用する場合は、E メール送信制限解除申請に入力する必要があります。

7. コマンドラインで、以下のコマンドを入力してsendmail.cfを書き込み可能にします。

```
sudo chmod 666 /etc/mail/sendmail.cf
```

8. コマンドラインで、以下のコマンドを入力してsendmail.cfを再生成します。

```
sudo sh -c 'm4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf'
```

> **Note**
>
> 「コマンドが見つかりません」や「そのようなファイルまたはディレクトリはありません」などといったエラーが発生した場合は、m4パッケージとsendmail-cfパッケージがシステムにインストールされていることを確認します。

9. コマンドラインで、以下のコマンドを入力してsendmail.cfの権限を読み取り専用にリセットします。

```
sudo chmod 644 /etc/mail/sendmail.cf
```

10. コマンドラインで、以下のコマンドを入力して Sendmail を再起動します。

```
sudo /etc/init.d/sendmail restart
```

    Linux または Sendmail のバージョンによって、上記がうまくいかない場合は、以下を試してください。

```
sudo su service sendmail restart
```

11. 以下の手順を実行して、テスト E メールを送信します。

    a. コマンドラインで、以下のコマンドを入力します。

```
/usr/sbin/sendmail -vf sender@example.com recipient@example.com
```

*sender@example.com*を送信元 E メールアドレスに置き換えてください。*recipient@example.com* を送信先アドレスに置き換えます。終了したら、Enter キーを押します。

b. 以下のメッセージの内容を入力します。各行の終わりで Enter キーを押します。

```
From: sender@example.com
To: recipient@example.com
Subject: Amazon SES test email

This is a test message sent from Amazon SES using Sendmail.
```

E メールの内容を入力し終わったら、Ctrl+D キーを押して送信します。

12. 受取人の E メールクライアントで E メールをチェックします。E メールが見つからない場合は、迷惑メールフォルダを確認します。それでも E メールが見つからない場合は、メールサーバー上の Sendmail ログを確認します。通常、ログは /var/log/mail.log または /var/log/maillog にあります。

Amazon SES を Microsoft Windows Server IIS SMTP に統合する

Amazon SES を介して E メールを送信するように Microsoft Windows Server の IIS SMTP サーバーを設定します。これらの手順は、Amazon EC2 インスタンスで Microsoft Windows Server 2022 を使用して書かれています。Microsoft Windows Server 2016 でも同じ設定を使用できます。

> **Note**
>
> Windows Server はサードパーティーのアプリケーションであり、Amazon Web Services によって開発またはサポートされていません。このセクションの手順は情報提供のみを目的としており、予告なく変更される場合があります。

Amazon SES と Microsoft Windows Server IIS SMTP を統合するには

1. まず、次の手順を使用して Microsoft Windows Server 2022 をセットアップします。

a. Amazon EC2 マネジメントコンソールから、新しい Microsoft Windows Server 2022 Base Amazon EC2 インスタンスを起動します。

b. 「Amazon EC2 Windows インスタンスの使用開始」の手順に従って、リモート Desktop によりそのインスタンスに接続してアクセスします。

c. Server Manager のダッシュボードを起動します。

d. [Web Server] ロールをインストールします。[IIS 6 Management Compatibility tools] (IIS 6 管理互換性ツール) ([Web Server] (ウェブサーバー) チェックボックスの下にあるオプション) を含めることを忘れないでください。

e. [SMTP Server] 機能をインストールします。

2. 次に、以下の手順を使用して IIS SMTP サービスを設定します。

a. Server Manager のダッシュボードに戻ります。

b. ツールメニューの Internet Information Services (IIS) 6.0 Manager を選択します。

c. SMTP Virtual Server #1 を右クリックし、プロパティを選択します。

d. アクセスタブの Relay Restrictions で、Relay を選択します。

e. Relay Restrictions ダイアログボックスで、追加を選択します。

f. Single Computer で、IP アドレスとして 127.0.0.1 と入力します。これで、IIS SMTP サービスを介して E メールを Amazon SES に中継する権限がこのサーバーに割り当てられました。

この手順では、E メールがこのサーバーで生成されることを前提としています。E メールを生成するアプリケーションが別のサーバーで実行されている場合は、IIS SMTP でそのサーバーの中継アクセスを許可する必要があります。

> **Note**
>
> SMTP リレーをプライベートサブネットに拡張するには、[Relay Restriction] で [Single Computer] 127.0.0.1 と [Group of Computers] 172.1.1.0 ~ 255.255.255.0 (ネットマスクセクション) を使用します。[Connection] で、[Single Computer] 127.0.0.1 と [Group of Computers] 172.1.1.0 ~ 255.255.255.0 (ネットマスクセクション) を使用します。

3. 最後に、以下の手順を使用して、Amazon SES を介して E メールを送信するようにサーバーを設定します。

a. SMTP Virtual Server #1 Propertiesダイアログボックスに戻り、Delivery タブを選択します。

b. [Delivery] タブで、[Outbound Security] を選択します。

c. [Basic Authentication] (基本認証) を選択し、Amazon SES SMTP 認証情報を入力します。これらの認証情報は、「Amazon SES SMTP 認証情報を取得」の手順に従って Amazon SES コンソールから取得できます。

> **Important**
>
> SMTP 認証情報は、 AWS アクセスキー ID およびシークレットアクセスキーとは異なります。 AWS 認証情報を使用して SMTP エンドポイントに対して自分自身を認証しようとしないでください。認証情報の詳細については、「Amazon SES 認証情報の種類」を参照してください。

d. [TLS encryption] が選択されていることを確認します。

e. [Delivery] タブに戻ります。

f. [Outbound Connections] を選択します。

g. [Outbound Connections] ダイアログボックスで、ポートが 25 または 587 であることを確認します。

h. [Advanced] を選択します。

i. スマートホスト名で、使用する Amazon SES エンドポイントを入力します (たとえば、email-smtp.us-west-2.amazonaws.com)。Amazon SES AWS リージョン が利用可能なのエンドポイント URLs「」の「Amazon Simple Email Service (Amazon SES)」を参照してくださいAWS 全般のリファレンス。

j. Server Manager のダッシュボードに戻ります。

k. Server Manager のダッシュボードで、[SMTP Virtual Server #1] を右クリックし、新しい設定が選択されるようにサービスを再起動します。

l. このサーバーを介して E メールを送信します。メッセージヘッダーを調べると、そのメッセージが Amazon SES を介して配信されたことを確認できます。

コマンドラインを使用して、Amazon SES SMTP インターフェイスへの接続をテストする

コマンドラインからこのセクションで説明されている方法を使用して、Amazon SES SMTP エンドポイントへの接続をテストし、SMTP 認証情報を検証し、接続の問題をトラブルシューティングできます。以下の手順では、代表的なオペレーティングシステムに搭載されているツールやライブラリを使用します。

SMTP 接続問題のトラブルシューティングの詳細については、「Amazon SES SMTP の問題」を参照してください。

前提条件

Amazon SES SMTP インターフェイスに接続するときは、SMTP 認証情報のセットを指定する必要があります。これらの SMTP 認証情報は、標準の AWS 認証情報とは異なります。2 つのタイプの認証情報は、一方をもう一方の代わりに使用することはできません。SMTP 認証情報を取得する方法の詳細については、「the section called “SMTP 認証情報の取得”」を参照してください。

Amazon SES SMTP インターフェイスへの接続のテスト

コマンドラインを使用すると、認証情報の入力やメッセージの送信を行うことなく、Amazon SES SMTP インターフェイスへの接続をテストできます。この手順は、基本的な接続問題のトラブルシューティングに役立ちます。テスト接続に失敗した場合は、「SMTP に関する問題」を参照してください。

このセクションでは、OpenSSL (ほとんどの Linux、macOS、Unix ディストリビューションに搭載され、Windows でも使用可能) と PowerShell の`Test-NetConnection`コマンドレット (ほとんどの最新バージョンの Windows に搭載) の両方を使用して接続をテストする手順を示します。

Linux, macOS, or Unix

OpenSSL を使用して Amazon SES SMTP インターフェイスに接続するには、ポート 587 経由で明示的な SSL を使用する方法と、ポート 465 経由で暗黙的な SSL を使用する方法の 2 つがあります。

明示的な SSL を使用して SMTP インターフェイスに接続するには

- コマンドラインで、次のコマンドを入力して Amazon SES SMTP サーバーに接続します。

```
openssl s_client -crlf -quiet -starttls smtp -connect email-smtp.us-
west-2.amazonaws.com:587
```

前述のコマンドで、*email-smtp.us-west-2.amazonaws.com* を AWS リージョンの Amazon SES SMTP エンドポイントの URL に置き換えます。詳細については、「the section called “リージョン”」を参照してください。

正常に接続されると次のような出力が表示されます。

```
depth=2 C = US, O = Amazon, CN = Amazon Root CA 1
verify return:1
depth=1 C = US, O = Amazon, OU = Server CA 1B, CN = Amazon
verify return:1
depth=0 CN = email-smtp.us-west-2.amazonaws.com
verify return:1
250 Ok
```

約 10 秒間何もしないと、接続は自動的に終了します。

または、暗黙的な SSL を使用して、ポート 465 経由で SMTP インターフェイスに接続することもできます。

暗黙的な SSL を使用して SMTP インターフェイスに接続するには

- コマンドラインで、次のコマンドを入力して Amazon SES SMTP サーバーに接続します。

```
openssl s_client -crlf -quiet -connect email-smtp.us-west-2.amazonaws.com:465
```

前述のコマンドで、*email-smtp.us-west-2.amazonaws.com* を AWS リージョンの Amazon SES SMTP エンドポイントの URL に置き換えます。詳細については、「the section called “リージョン”」を参照してください。

正常に接続されると次のような出力が表示されます。

```
depth=2 C = US, O = Amazon, CN = Amazon Root CA 1
verify return:1
depth=1 C = US, O = Amazon, OU = Server CA 1B, CN = Amazon
verify return:1
depth=0 CN = email-smtp.us-west-2.amazonaws.com
```

```
verify return:1
220 email-smtp.amazonaws.com ESMTP SimpleEmailService-d-VCSHDP1YZ
 A1b2C3d4E5f6G7h8I9j0
```

約 10 秒間何もしないと、接続は自動的に終了します。

PowerShell

PowerShell の Test-NetConnection コマンドレットを使用して Amazon SES SMTP サーバーに接続します。

> **Note**
>
> Test-NetConnection コマンドレットでは、コンピュータが Amazon SES SMTP エンドポイントに接続できるかどうかを判断できます。ただし、コンピュータが SMTP エンドポイントに暗黙的な SSL 接続や明示的な SSL 接続を確立できるかどうかはテストしません。SSL 接続をテストするには、OpenSSL for Windows をインストールするか、テスト E メールを送信できます。

Test-NetConnection コマンドレットを使用して SMTP インターフェイスに接続するには

- Amazon SES で、次のコマンドを入力して Amazon SES SMTP サーバーに接続します。

  ```
  Test-NetConnection -Port 587 -ComputerName email-smtp.us-west-2.amazonaws.com
  ```

  前述のコマンドで、*email-smtp.us-west-2.amazonaws.com* を AWS リージョンの Amazon SES SMTP エンドポイントの URL に置き換え、*587* をポート番号に置き換えます。Amazon SES の地域のエンドポイントの詳細については、の「the section called “リージョン”」を参照してください。

  接続に成功すると、次の例のような出力が表示されます。

  ```
  ComputerName     : email-smtp.us-west-2.amazonaws.com
  RemoteAddress    : 198.51.100.126
  RemotePort       : 587
  InterfaceAlias   : Ethernet
  SourceAddress    : 203.0.113.46
  TcpTestSucceeded : True
  ```

コマンドラインを使用し、Amazon SES SMTP インターフェイスを介して E メールを送信する

コマンドラインを使用して、Amazon SES SMTP インターフェイスを介してメッセージを送信することもできます。この手順は、SMTP 認証情報をテストする場合や、Amazon SES を使用して送信するメッセージを特定の受信者が受信できるかどうかをテストする場合に役立ちます。

Linux, macOS, or Unix

E メール送信者が SMTP サーバーに接続すると、クライアントはスタンダードなリクエストのセットを発行し、サーバーは各リクエストにスタンダードのレスポンスで応答します。この一連のリクエストとレスポンスは SMTP 対話と呼ばれます。OpenSSL を使用して Amazon SES SMTP サーバーに接続すると、サーバーは SMTP 対話が発生することを予想します。

OpenSSL を使用して SMTP インターフェイスに接続する場合は、base64 エンコードを使用して SMTP 認証情報をエンコードする必要があります。このセクションでは、base64 を使用して認証情報をエンコードする手順を示します。

SMTP インターフェイスを使用してコマンドラインから E メールを送信するには

1. コマンドラインで以下を入力します。*email-smtp.us-west-2.amazonaws.com* は、使用する AWS リージョンの Amazon SES SMTP エンドポイントの URL に置き換えます。詳細については、「」を参照してくださいthe section called “リージョン”。

```
#!/bin/bash

# Prompt user to provide following information
read -p "Configuration set: " CONFIGSET
read -p "Enter SMTP username: " SMTPUsername
read -p "Enter SMTP password: " SMTPPassword
read -p "Sender email address: " MAILFROM
read -p "Receiver email address: " RCPT
read -p "Email subject: " SUBJECT
read -p "Message to send: " DATA

echo

# Encode SMTP username and password using base64
EncodedSMTPUsername=$(echo -n "$SMTPUsername" | openssl enc -base64)
EncodedSMTPPassword=$(echo -n "$SMTPPassword" | openssl enc -base64)

# Construct the email
```

```
Email="EHLO example.com
AUTH LOGIN
$EncodedSMTPUsername
$EncodedSMTPPassword
MAIL FROM: $MAILFROM
RCPT TO: $RCPT
DATA
X-SES-CONFIGURATION-SET: $CONFIGSET
From: $MAILFROM
To: $RCPT
Subject: $SUBJECT

$DATA
.
QUIT"

echo "$Email" | openssl s_client -crlf -quiet -starttls smtp -connect email-smtp.us-west-2.amazonaws.com:587
```

2. 各変数のプロンプトで、値を入力します。
3. • ポート 465 経由で暗黙的な SSL を使用して送信するには、以下を使用します。

```
openssl s_client -crlf -quiet -connect email-smtp.us-west-2.amazonaws.com:465
```

メッセージが Amazon SES によって受け入れられた場合は、次の例のような出力が表示されます。

```
250 Ok 01010160d7de98d8-21e57d9a-JZho-416c-bbe1-8ebaAexample-000000
```

250 Ok に続く数字の文字列とテキストは、E メールのメッセージ ID です。

> **Note**
>
> 約 10 秒間何もしないと、接続は自動的に終了します。

PowerShell

Net.Mail.SmtpClient クラスを使用すると、明示的な SSL を通じてポート 587 経由で E メールを送信できます。

> **Note**
>
> Net.Mail.SmtpClientクラスは正式に廃止されており、Microsoft はサードパーティー製のライブラリを使用することを推奨しています。このコードはテスト専用であり、本番環境用のワークロードには使用しないでください。

明示的な SSL を使用して PowerShell 経由で E メールを送信するには

1. テキストエディタで新規ファイルを作成します。ファイルに次のコードを貼り付けます。

```
function SendEmail($Server, $Port, $Sender, $Recipient, $Subject, $Body) {
    $Credentials = [Net.NetworkCredential](Get-Credential)

    $SMTPClient = New-Object Net.Mail.SmtpClient($Server, $Port)
    $SMTPClient.EnableSsl = $true
    $SMTPClient.Credentials = New-Object
 System.Net.NetworkCredential($Credentials.Username, $Credentials.Password);

    try {
        Write-Output "Sending message..."
        $SMTPClient.Send($Sender, $Recipient, $Subject, $Body)
        Write-Output "Message successfully sent to $($Recipient)"
    } catch [System.Exception] {
        Write-Output "An error occurred:"
        Write-Error $_
    }
}

function SendTestEmail(){
    $Server = "email-smtp.us-west-2.amazonaws.com"
    $Port = 587

    $Subject = "Test email sent from Amazon SES"
    $Body = "This message was sent from Amazon SES using PowerShell (explicit
 SSL, port 587)."

    $Sender = "sender@example.com"
    $Recipient = "recipient@example.com"

    SendEmail $Server $Port $Sender $Recipient $Subject $Body
}
```

```
SendTestEmail
```

終了したら、SendEmail.ps1としてファイルを保存します。

2. 前のステップで作成したファイルを次のように変更します。

   - *sender@example.com* を、メッセージの送信元の E メールアドレスに置き換えます。
   - *recipient@example.com*をメッセージの送信先アドレスに置き換えます。
   - *email-smtp.us-west-2.amazonaws.com*を AWS 地域の Amazon SES SMTP エンドポイントの URL に置き換えます。詳細については、「リージョンと Amazon SES」を参照してください。

3. PowerShell を使用して、次のコマンドを入力します。

   ```
   .\path\to\SendEmail.ps1
   ```

   上のコマンドで、*path\to\SendEmail.ps1* をステップ 1 で作成したファイルへのパスに置き換えます。

4. プロンプトが表示されたら、SMTP ユーザー名とパスワードを入力します。

または、System.Web.Mail.SmtpMail クラスを使用して、ポート 465 経由で暗黙的な SSL を通じて E メールを送信することもできます。

> **Note**
>
> System.Web.Mail.SmtpMail クラスは正式に廃止されており、Microsoft はサードパーティー製のライブラリを使用することを推奨しています。このコードはテスト専用であり、本番環境用のワークロードには使用しないでください。

暗黙的な SSL を使用して PowerShell 経由で E メールを送信するには

1. テキストエディタで新規ファイルを作成します。ファイルに次のコードを貼り付けます。

   ```
   [System.Reflection.Assembly]::LoadWithPartialName("System.Web") > $null

   function SendEmail($Server, $Port, $Sender, $Recipient, $Subject, $Body) {
       $Credentials = [Net.NetworkCredential](Get-Credential)
   ```

```
    $mail = New-Object System.Web.Mail.MailMessage
    $mail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/
smtpserver", $Server)
    $mail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/
smtpserverport", $Port)
    $mail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/
smtpusessl", $true)
    $mail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/
sendusername", $Credentials.UserName)
    $mail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/
sendpassword", $Credentials.Password)
    $mail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/
smtpconnectiontimeout", $timeout / 1000)
    $mail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/sendusing",
 2)
    $mail.Fields.Add("http://schemas.microsoft.com/cdo/configuration/
smtpauthenticate", 1)

    $mail.From = $Sender
    $mail.To = $Recipient
    $mail.Subject = $Subject
    $mail.Body = $Body

    try {
        Write-Output "Sending message..."
        [System.Web.Mail.SmtpMail]::Send($mail)
        Write-Output "Message successfully sent to $($Recipient)"
    } catch [System.Exception] {
        Write-Output "An error occurred:"
        Write-Error $_
    }
}

function SendTestEmail(){
    $Server = "email-smtp.us-west-2.amazonaws.com"
    $Port = 465

    $Subject = "Test email sent from Amazon SES"
    $Body = "This message was sent from Amazon SES using PowerShell (implicit
 SSL, port 465)."

    $Sender = "sender@example.com"
    $Recipient = "recipient@example.com"
```

```
    SendEmail $Server $Port $Sender $Recipient $Subject $Body
}

SendTestEmail
```

終了したら、SendEmail.ps1としてファイルを保存します。

2. 前のステップで作成したファイルを次のように変更します。

   - *sender@example.com* を、メッセージの送信元の E メールアドレスに置き換えます。
   - *recipient@example.com*をメッセージの送信先アドレスに置き換えます。
   - *email-smtp.us-west-2.amazonaws.com*を AWS 地域の Amazon SES SMTP エンドポイントの URL に置き換えます。詳細については、「リージョンと Amazon SES」を参照してください。

3. PowerShell を使用して、次のコマンドを入力します。

```
.\path\to\SendEmail.ps1
```

上のコマンドで、*path\to\SendEmail.ps1* をステップ 1 で作成したファイルへのパスに置き換えます。

4. プロンプトが表示されたら、SMTP ユーザー名とパスワードを入力します。

Amazon SES API を使用して E メールを送信する

Amazon SES を介して本稼働 E メールを送信する場合には、Simple Mail Transfer Protocol (SMTP) インターフェイスまたは Amazon SES API を使用できます。 SMTP インターフェイスの詳細については、「Amazon SES SMTP インターフェイスを使用して E メールを送信」を参照してください。このセクションでは、API を使用して E メールを送信する方法について説明します。

Amazon SES の API を使用して E メールを送信する場合は、メッセージの内容を指定し、Amazon SES は MIME E メールを組み立てます。また、メッセージの内容を完全に制御できるように、E メールを自分で構成することもできます。API の詳細については、Amazon Simple Email Service API リファレンスを参照してください。Amazon SES AWS リージョン が利用可能な のエンドポイント URLs「」の「Amazon Simple Email Service エンドポイントとクォータ」を参照してくださいAWS 全般のリファレンス。

以下の方法で API を呼び出すことができます。

- 直接 HTTPS リクエストを作成する - これは、リクエストの認証と署名を手動で処理し、リクエストを手動で作成する必要があるため、最も高度な方法です。Amazon SES API の詳細については、API v2 リファレンスページのWelcomeを参照してください。
- AWS SDK を使用する —AWS SDKsを使用すると、Amazon SES を含む複数の AWS サービスの APIs に簡単にアクセスできます。 Amazon SES SDK を使用すると、認証、リクエスト署名、再試行ロジック、エラー処理などの低レベルの機能が自動的に実行されるため、充実したアプリケーションの構築に専念できます。
- command line インターフェイスの使用 – AWS Command Line Interface は、Amazon SES 用のコマンドラインツールです。さらに、PowerShell 環境でスクリプトを作成できるように、AWS Tools for PowerShell も提供しています。

Amazon SES API に直接アクセスするか、 SDK、、 AWS Command Line Interface または AWS Tools for PowerShell を介して AWS 間接的にアクセスするかにかかわらず、Amazon SES API には、E メールメッセージの構成に対する制御の程度に応じて、2 つの異なる方法で E メールを送信できます。

- フォーマット済み - Amazon SES は、正しくフォーマットされた E メールメッセージを構成して送信します。ユーザーは、送信元と宛先のアドレス、件名、メッセージ本文を入力するだけです。Amazon SES は残りのすべてを処理します。詳細については、「Amazon SES API を使用してフォーマット済み E メールを送信する」を参照してください。
- Raw - E メールメッセージを手動で構成して送信します。自分で E メールヘッダーおよび MIME の種類を指定します。E メールのフォーマット作業を自分で行った経験があれば、未処理のインターフェイスにより、メッセージの構成をより詳細に制御できます。詳細については、「Amazon SES API v2 を使用した raw E メールの送信」を参照してください。

内容

Amazon SES API を使用してフォーマット済み E メールを送信する

フォーマットされた E メールを送信するには、 を使用する AWS Management Console か、アプリケーションから直接 Amazon SES API を呼び出すか、 AWS SDK、、 AWS Command Line Interfaceまたは を介して間接的に呼び出します AWS Tools for Windows PowerShell。

Amazon SES API は、 `SendEmail`アクションを提供し、フォーマット済み E メールを構成して送信できるようにします。 `SendEmail`には、送信元アドレス、宛先アドレス、メッセージの件名、およびメッセージの本文 (テキスト、 HTML、またはその両方) が必要です。詳細については、SendEmail(API リファレンス) またはSendEmail (API v2 リファレンス) を参照してください。

> **Note**
>
> E メールアドレスは、7 ビット ASCII 文字列になっている必要があります。送信先または送信元の E メールアドレス内で、ドメインの部分に Unicode 文字が含まれる場合は、Punycode を使用してドメインをエンコードする必要があります。詳細については、RFC 3492 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フォーマット済みメッセージを構成する方法の例は、コードの例を参照してください。

`SendEmail`に複数の呼び出しを作成する場合の E メール送信速度向上方法については、「Amazon SES のスループットを上げる」を参照してください。

Amazon SES API v2 を使用した raw E メールの送信

コンテンツタイプを `raw` として指定して Amazon SES API v2 `SendEmail` オペレーションを使用すると、raw E メール形式でカスタマイズしたメッセージを受信者に送信できます。

E メールヘッダーフィールドについて

Simple Mail Transfer Protocol (SMTP) は、E メールエンベロープとそのパラメータのいくつかを定義することにより、E メールメッセージの送信方法を定義しますが、メッセージの内容については何も定義しません。一方、Internet Message Format (RFC 5322) は、メッセージの構成方法を定義します。

Internet Message Format の仕様に従って、すべてのメッセージ E メールはヘッダーと本文から構成されます。ヘッダーはメッセージのメタデータで構成され、本文にメッセージそのものが含まれま

す。E メールヘッダーと本文の詳細については、「Amazon SESのE メール形式」を参照してください。

MIME の使用

SMTP プロトコルはもともと 7 ビット ASCII 文字のみを含む E メールメッセージを送信するように設計されていました。この仕様により、ASCII 以外のテキストエンコード (Unicode など)、バイナリコンテンツ、または添付ファイルでは SMTP が不十分になります。多目的インターネットメール拡張 (MIME) 標準は、SMTP を使用して、他の多くの種類のコンテンツを送信できるようにするために開発されたものです。

MIME 標準には、メッセージ本文を複数のパートに分割し、パートごとに、どのような操作を行うかを指定する機能があります。たとえば、E メールメッセージ本文の、あるパートはプレーンテキスト、別のパートは HTML という場合があります。さらに、MIME では、E メールメッセージに 1 つ以上の添付ファイルを含めることができます。メッセージの受取人は、E メールクライアント内から添付ファイルを見たり、保存したりできます。

メッセージヘッダーとコンテンツとは空白行で分離されます。E メールの各パートは、boundary で分離されます。boundary は、各パートの開始と終了を示す文字列です。

次の例のマルチパートメッセージには、テキストと HTML パート、および添付ファイルが含まれています。添付ファイルは添付ファイルヘッダーのすぐ下に配置する必要があります。ほとんどの場合、この例のように base64 でエンコードされます。

```
From: "Sender Name" <sender@example.com>
To: recipient@example.com
Subject: Customer service contact info
Content-Type: multipart/mixed;
    boundary="a3f166a86b56ff6c37755292d690675717ea3cd9de81228ec2b76ed4a15d6d1a"

--a3f166a86b56ff6c37755292d690675717ea3cd9de81228ec2b76ed4a15d6d1a
Content-Type: multipart/alternative;
    boundary="sub_a3f166a86b56ff6c37755292d690675717ea3cd9de81228ec2b76ed4a15d6d1a"

--sub_a3f166a86b56ff6c37755292d690675717ea3cd9de81228ec2b76ed4a15d6d1a
Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: quoted-printable

Please see the attached file for a list of customers to contact.

--sub_a3f166a86b56ff6c37755292d690675717ea3cd9de81228ec2b76ed4a15d6d1a
```

```
Content-Type: text/html; charset=iso-8859-1
Content-Transfer-Encoding: quoted-printable

<html>
<head></head>
<body>
<h1>Hello!</h1>
<p>Please see the attached file for a list of customers to contact.</p>
</body>
</html>

--sub_a3f166a86b56ff6c37755292d690675717ea3cd9de81228ec2b76ed4a15d6d1a--

--a3f166a86b56ff6c37755292d690675717ea3cd9de81228ec2b76ed4a15d6d1a
Content-Type: text/plain; name="customers.txt"
Content-Description: customers.txt
Content-Disposition: attachment;filename="customers.txt";
    creation-date="Sat, 05 Aug 2017 19:35:36 GMT";
Content-Transfer-Encoding: base64

SUQsRmlyc3ROYW1lLExhc3ROYW1lLENvdW50cnkKMzQ4LEpvaG4sU3RpbGVzLENhbmFkYQo5MjM4
OSxKaWUsTGl1LENoaW5hCjczNCxTaGlybGV5LFJvZHJpZ3VleixVbml0ZWQgU3RhdGVzCjI4OTMs
QW5heWEsSXllbmdhcixJbmRpYQ==

--a3f166a86b56ff6c37755292d690675717ea3cd9de81228ec2b76ed4a15d6d1a--
```

メッセージのコンテンツタイプが multipart/mixed であることから、メッセージに多数のパートがあり (この例では、本文および添付ファイル)、受信するクライアントは各パートを別々に扱う必要があることがわかります。

本文セクション内に入れ子になっているのは、multipart/alternative コンテンツタイプを使用する 2 番目のパートです。このコンテンツタイプは、各パートに同じコンテンツの代替バージョンが含まれる (この場合はテキストバージョンおよび HTML バージョン) ことを示します。受取人の E メールクライアントで HTML コンテンツを表示できる場合は、メッセージ本文の HTML バージョンが表示されます。受取人の E メールクライアントで HTML コンテンツを表示できない場合は、メッセージ本文のプレーンテキストバージョンが表示されます。

メッセージの両方のバージョンには添付ファイルも含まれます (この場合、一部の顧客名を含むショートテキストファイル)。

この例のように MIME パートを他のパートに入れ子にすると、入れ子になったパートは、親のパートの boundary パラメータとは異なる boundary パラメータを使用する必要があります。これらの

境界は固有の文字列でなければなりません。MIME パーツ間の境界を定義するには、2 つのハイフンを (--) タイプし、その後に境界文字列が続きます。MIME パーツの最後に、境界文字列の先頭および末尾の両方に 2 つのハイフンを置きます。

> **Note**
>
> メッセージに 500 MIME を超えるパーツを含めることはできません。

MIME エンコード

古いシステムとの互換性を維持するために、Amazon SES は RFC 2821 で定義されている SMTP の 7 ビット ASCII 制限を優先します。非 ASCII 文字を含むコンテンツを送信する場合は、これらの文字を 7 ビット ASCII 文字を使用する形式にエンコードする必要があります。

E メールアドレス

E メールアドレスは、7 ビット ASCII 文字列になっている必要があります。送信先または送信元の E メールアドレス内で、ドメインの部分に Unicode 文字が含まれる場合は、Punycode を使用してドメインをエンコードする必要があります。Punycode は E メールアドレスのローカル部分 (@ 記号の前の部分) では許可されていません。また、「差出人」名にも許可されていません。「フレンドリ元」名に Unicode 文字を使用する場合は、このセクションで説明するように、MIME エンコードされた単語構文を使用して「フレンドリ元」名をエンコードする必要があります。Punycode の詳細については、RFC 3492 を参照してください。

> **Note**
>
> このルールは、メッセージヘッダーではなく、メッセージエンベロープで指定する E メールアドレスにのみ適用されます。Amazon SES API v2 `SendEmail` オペレーションを使用する場合、`Source` パラメータと `Destinations` パラメータで指定するアドレスが、それぞれエンベロープの送信者と受信者を定義することになります。

E メールヘッダー

メッセージヘッダーをエンコードするには、MIME encoded-word 構文を使用します。MIME encoded word 構文では、次の形式が使用されます。

```
=?charset?encoding?encoded-text?=
```

*encoding*の値は Qまたは Bとなります。エンコードの値が Q の場合、値 *encoded-text* には Q エンコードを使用する必要があります。エンコードの値が B の場合、*encoded-text* の値には base64 エンコードを使用する必要があります。

たとえば、「Як ти поживаєш?」を使用する場合 E メールの件名に次のエンコードのいずれかを使用することができます。

- Q エンコード

```
=?utf-8?Q?
=D0=AF=D0=BA_=D1=82=D0=B8_=D0=BF=D0=BE=D0=B6=D0=B8=D0=B2=D0=B0=D1=94=D1=88=3F?=
```

- Base64 エンコード

```
=?utf-8?B?0K/QuiDRgtC4INC/0L7QttC40LLQsNGU0Yg/?=
```

Q エンコードの詳細については、RFC 2047 を参照してください。base64 エンコードの詳細については、RFC 2045 を参照してください。

メッセージ本文

メッセージの本文をエンコードするには、quoted-printable エンコードまたは base64 エンコードを使用できます。次に、Content-Transfer-Encoding ヘッダーを使用して、使用するエンコードスキームを指定します。

たとえば、メッセージの本文に次のテキストが含まれているとします。

१९७२ मे रे टॉमलंसिन ने पहला ई-मेल संदश भेजा | रे टॉमलंसिन ने ही स्रव्परथम @ चन्हि का चयन कयिा और इनही को ईमल का आवष्किारक माना जाता है

base64 エンコードを使用してこのテキストをエンコードする場合、最初に以下のヘッダーを指定します。

```
Content-Transfer-Encoding: base64
```

次に、E メールの本文セクションに、base64 でエンコードされたテキストを含めます。

```
4KWn4KWv4KWt4KWoIOCkruClhyDgpLDgpYcg4KSf4KWJ4KSu4KSy4KS/4KSC4KS44KSoIOCkqOCl
hyDgpKrgpLngpLLgpL4g4KSILeCkruClh+CksiDgpLjgpILgpKbgpYfgpLYg4KSt4KWH4KSc4KS+
```

```
IHwg4KSw4KWHIOCkn+ClieCkruCksuCkv+CkguCkuOCkqCDgpKjgpYcg4KS54KWAIOCkuOCksOCl
jeCkteCkquCljeCksOCkpeCkriBAIOCkmuCkv+CkqOCljeCkuSDgpJXgpL4g4KSa4KSv4KSoIOCk
leCkv+Ckr+CkviDgpJTgpLAg4KSH4KSo4KWN4KS54KWAIOCkleCliyDgpIjgpK7gpYfgpLIg4KSV
4KS+IOCkhuCkteCkv+Ckt+CljeCkleCkvuCksOCklSDgpK7gpL7gpKjgpL4g4KSc4KS+4KSk4KS+
IOCkueCliEAo=
```

> **Note**
>
> 場合によっては、Amazon SES を使用して送信するメッセージに 8 ビットのContent-Transfer-Encodingを使用できます。ただし、Amazon SES がメッセージを変更する必要がある場合 (たとえば、オープンとクリックの追跡を使用した場合)、8 ビットでエンコードされたコンテンツは、受取人の受信トレイに届いたときに正しく表示されないことがあります。このため、7 ビットの ASCII 以外のコンテンツは常にエンコードする必要があります。

添付ファイル

E メールにファイルをアタッチするには、base64 エンコードを使用して添付ファイルをエンコードする必要があります。添付ファイルは通常、次のヘッダーを含む専用の MIME メッセージ部分に配置されています。

- Content-Type – 添付ファイルの種類。一般的な MIME Content-Type 宣言の例を以下に示します。
  - プレーンテキストファイル – `Content-Type: text/plain; name="sample.txt"`
  - Microsoft Word ドキュメント – `Content-Type: application/msword; name="document.docx"`
  - JPG イメージ – `Content-Type: image/jpeg; name="photo.jpeg"`
- Content-Disposition – 受取人の E メールクライアントがコンテンツをどのように処理するかを指定します。添付ファイルの場合、この値は `Content-Disposition: attachment` です。
- Content-Transfer-Encoding – 添付ファイルのエンコードに使用されるスキーム。添付ファイルでは、ほとんどの場合この値は `base64` です。
- エンコードされた添付ファイル – 例に示すように、実際の添付ファイルをエンコードして、添付ファイルヘッダーの下の本文に含める必要があります。

Amazon SES は最も一般的なファイルタイプに対応しています。Amazon SES が対応していないファイルの種類の一覧については、「Amazon SES でサポートされていない添付ファイルの種類」を参照してください。

Amazon SES API v2 を使用した raw E メールの送信

Amazon SES API v2 は SendEmail アクションを提供します。これにより、コンテンツタイプを simple、raw、または templated のいずれかを指定し、このように指定した形式の E メールメッセージを作成して送信できるようになります。詳細については、「SendEmail」を参照してください。次の例では、コンテンツ タイプを raw として指定して、raw E メール形式を使用してメッセージを送信します。

> **Note**
>
> SendEmail に複数の呼び出しを作成する場合の E メール送信速度向上方法については、「Amazon SES のスループットを上げる」を参照してください。

メッセージ本文には、正しくフォーマットされ、適切なヘッダーフィールドとメッセージ本文が エンコードされた raw E メールメッセージを含める必要があります。アプリケーション内で raw メッセージを手動で構成することはできますが、既存のメールライブラリを使用して構成するほうが、はるかに簡単です。

Java

次のコード例は、JavaMailライブラリとAWS SDK for Javaを使用して raw E メールを作成および送信する方法を示しています。

```
package com.amazonaws.samples;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.PrintStream;
import java.nio.ByteBuffer;
import java.util.Properties;

// JavaMail libraries. Download the JavaMail API
// from https://javaee.github.io/javamail/
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.activation.FileDataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
```

```
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;

// AWS SDK libraries. Download the AWS SDK for Java // from https://aws.amazon.com/
sdk-for-java
import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleemail.AmazonSimpleEmailService;
import com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClientBuilder;
import com.amazonaws.services.simpleemail.model.RawMessage;
import com.amazonaws.services.simpleemail.model.SendRawEmailRequest;

public class AmazonSESSample {

 // Replace sender@example.com with your "From" address.
 // This address must be verified with Amazon SES.
 private static String SENDER = "Sender Name <sender@example.com>";

 // Replace recipient@example.com with a "To" address. If your account
 // is still in the sandbox, this address must be verified.
 private static String RECIPIENT = "recipient@example.com";

 // Specify a configuration set. If you do not want to use a configuration
 // set, comment the following variable, and the
 // ConfigurationSetName=CONFIGURATION_SET argument below.
 private static String CONFIGURATION_SET = "ConfigSet";

 // The subject line for the email.
 private static String SUBJECT = "Customer service contact info";

 // The full path to the file that will be attached to the email.
 // If you're using Windows, escape backslashes as shown in this variable.
 private static String ATTACHMENT = "C:\\Users\\sender\\customers-to-contact.xlsx";

 // The email body for recipients with non-HTML email clients.
 private static String BODY_TEXT = "Hello,\r\n"
                                   + "Please see the attached file for a list "
                                   + "of customers to contact.";

 // The HTML body of the email.
 private static String BODY_HTML = "<html>"
                                   + "<head></head>"
                                   + "<body>"
```

```
                                        + "<h1>Hello!</h1>"
                                        + "<p>Please see the attached file for a "
                                        + "list of customers to contact.</p>"
                                        + "</body>"
                                        + "</html>";

   public static void main(String[] args) throws AddressException,
MessagingException, IOException {

    Session session = Session.getDefaultInstance(new Properties());

       // Create a new MimeMessage object.
       MimeMessage message = new MimeMessage(session);

       // Add subject, from and to lines.
       message.setSubject(SUBJECT, "UTF-8");
       message.setFrom(new InternetAddress(SENDER));
       message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(RECIPIENT));

       // Create a multipart/alternative child container.
       MimeMultipart msg_body = new MimeMultipart("alternative");

       // Create a wrapper for the HTML and text parts.
       MimeBodyPart wrap = new MimeBodyPart();

       // Define the text part.
       MimeBodyPart textPart = new MimeBodyPart();
       textPart.setContent(BODY_TEXT, "text/plain; charset=UTF-8");

       // Define the HTML part.
       MimeBodyPart htmlPart = new MimeBodyPart();
       htmlPart.setContent(BODY_HTML,"text/html; charset=UTF-8");

       // Add the text and HTML parts to the child container.
       msg_body.addBodyPart(textPart);
       msg_body.addBodyPart(htmlPart);

       // Add the child container to the wrapper object.
       wrap.setContent(msg_body);

       // Create a multipart/mixed parent container.
       MimeMultipart msg = new MimeMultipart("mixed");
```

```
        // Add the parent container to the message.
        message.setContent(msg);

        // Add the multipart/alternative part to the message.
        msg.addBodyPart(wrap);

        // Define the attachment
        MimeBodyPart att = new MimeBodyPart();
        DataSource fds = new FileDataSource(ATTACHMENT);
        att.setDataHandler(new DataHandler(fds));
        att.setFileName(fds.getName());

        // Add the attachment to the message.
        msg.addBodyPart(att);

        // Try to send the email.
        try {
            System.out.println("Attempting to send an email through Amazon SES "
                               +"using the AWS SDK for Java...");

            // Instantiate an Amazon SES client, which will make the service
            // call with the supplied AWS credentials.
            AmazonSimpleEmailService client =
                    AmazonSimpleEmailServiceClientBuilder.standard()
                    // Replace US_WEST_2 with the AWS Region you're using for
                    // Amazon SES.
                    .withRegion(Regions.US_WEST_2).build();

            // Print the raw email content on the console
            PrintStream out = System.out;
            message.writeTo(out);

            // Send the email.
            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            message.writeTo(outputStream);
            RawMessage rawMessage =
              new RawMessage(ByteBuffer.wrap(outputStream.toByteArray()));

            SendRawEmailRequest rawEmailRequest =
              new SendRawEmailRequest(rawMessage)
                  .withConfigurationSetName(CONFIGURATION_SET);

            client.sendRawEmail(rawEmailRequest);
            System.out.println("Email sent!");
```

```
        // Display an error if something goes wrong.
        } catch (Exception ex) {
          System.out.println("Email Failed");
            System.err.println("Error message: " + ex.getMessage());
            ex.printStackTrace();
        }
    }
}
```

Python

次のコード例は、Python email.mime パッケージと AWS SDK for Python (Boto) を使用して raw E メールを作成および送信する方法を示しています。

```
import json
import boto3
from botocore.exceptions import ClientError
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.application import MIMEApplication
import os

def boto3_rawemailv2():
    SENDER = "Sender <sender@example.com>"
    RECIPIENT = "recipient@example.com"
    CONFIGURATION_SET = "ConfigSet"
    AWS_REGION = "us-east-1"
    SUBJECT = "Customer service contact info"
    ATTACHMENT = "path/to/customers-to-contact.xlsx"
    BODY_TEXT = "Hello,\r\nPlease see the attached file for a list of customers to
 contact."

    # The HTML body of the email.
    BODY_HTML = """\
    <html>
    <head/>
    <body>
    <h1>Hello!</h1>
    <p>Please see the attached file for a list of customers to contact.</p>
    </body>
    </html>
    """
```

```
    # The character encoding for the email.
    CHARSET = "utf-8"
    msg = MIMEMultipart('mixed')
    # Add subject, from and to lines.
    msg['Subject'] = SUBJECT
    msg['From'] = SENDER
    msg['To'] = RECIPIENT

    # Create a multipart/alternative child container.
    msg_body = MIMEMultipart('alternative')

    # Encode the text and HTML content and set the character encoding. This step is
    # necessary if you're sending a message with characters outside the ASCII range.
    textpart = MIMEText(BODY_TEXT.encode(CHARSET), 'plain', CHARSET)
    htmlpart = MIMEText(BODY_HTML.encode(CHARSET), 'html', CHARSET)

    # Add the text and HTML parts to the child container.
    msg_body.attach(textpart)
    msg_body.attach(htmlpart)

    # Define the attachment part and encode it using MIMEApplication.
    att = MIMEApplication(open(ATTACHMENT, 'rb').read())

    # Add a header to tell the email client to treat this part as an attachment,
    # and to give the attachment a name.
    att.add_header('Content-
Disposition','attachment',filename=os.path.basename(ATTACHMENT))

    # Attach the multipart/alternative child container to the multipart/mixed
    # parent container.
    msg.attach(msg_body)
    msg.attach(att)

    #changes start from here
    strmsg = str(msg)
    body = bytes (strmsg, 'utf-8')




    client = boto3.client('sesv2')
    response = client.send_email(
    FromEmailAddress=SENDER,
    Destination={
```

```
        'ToAddresses': [RECIPIENT]
    },
    Content={
        'Raw': {
            'Data': body
        }
    }
    )
    print(response)
boto3_rawemailv2 ()
```

テンプレートを使用して、Amazon SES API でパーソナライズされた E メールを送信する

Amazon SES では、保存されたテンプレートを使用するか、インラインテンプレートを使用して、テンプレート化された E メールを送信できます。

- 保存済みテンプレート – Amazon SES v2 API の `CreateEmailTemplate`オペレーションを使用して SES で作成および保存される`Template`リソースを指します。 Amazon SES テンプレートには、書き込まれたコンテンツとインラインで変数 (プレースホルダー) を含む E メールの件名と本文が含まれています。ストアドテンプレートの名前とテンプレート内のプレースホルダー変数への動的データは、 `SendEmail`または v2 API `SendBulkEmail` オペレーションを呼び出すときに提供されます。

  保存されたテンプレートは簡単に再利用でき、同様の種類の E メールを送信する際の時間と労力を節約できます。各 E メールをゼロから作成する代わりに、基本構造と設計を一度作成するだけで、テンプレート内の動的コンテンツを更新できます。
- インラインテンプレート – `Template`リソースは使用されませんが、変数 (プレースホルダー) を含む E メールの件名と本文は、これらのプレースホルダー変数の値とともに、 `SendEmail`または v2 API `SendBulkEmail` オペレーションを呼び出すときにインラインで提供されます。

  インラインテンプレートは、SES アカウントでテンプレートリソースを管理する必要がなくなり、アプリケーションロジック内にテンプレートコンテンツを直接含めることができるため、統合プロセスを簡素化することで、一括 E メールを送信するプロセスを合理化します。あたりの 20,000 テンプレートの制限にはカウントされません AWS リージョン。

ストアドテンプレートを使用する場合、次の制限が適用されます。

- それぞれに最大 20,000 個の E メールテンプレートを作成できます AWS リージョン。
- 各テンプレートは、テキストと HTML パートの両方を含めて、最大 500 KB のサイズまで可能です。

インラインテンプレートを使用する場合、次の制限が適用されます。

- 各入力 JSON ファイルは、テキスト部分と HTML 部分の両方を含めて、最大 1 MB のサイズにすることができます。

以下は、保存済みテンプレートとインラインテンプレートの両方に適用されます。

- 使用できる置換変数の数に制限はありません。
- `SendBulkEmail` オペレーションの呼び出しごとに、最大 50 個の送信先オブジェクトに E メールを送信できます。`Destination` オブジェクトには、ToAddresses、CcAddresses、および BccAddresses で定義された複数の受信者を含めることができます。v2 API への 1 回の呼び出しで問い合わせることができる送信先の数は、アカウントの最大送信レートによって制限される場合があります。詳細については、「Amazon SES 送信制限の管理」を参照してください。

この章では、ストアドテンプレートとインラインテンプレートの両方を使用する例を含む手順について説明します。

> **Note**
>
> これらの手順では、 AWS CLIがすでにインストールされ、設定されていることを前提としています。のインストールと設定の詳細については AWS CLI、「 AWS Command Line Interface ユーザーガイド」を参照してください。

(オプション) パート 1: レンダリング失敗イベント通知を設定する

無効なパーソナライズコンテンツを含む E メールを送信する場合、Amazon SES はメッセージを受け入れることもありますが、配信することはできません。このため、パーソナライズされた E メールを送信する場合は、Amazon SNS を介してレンダリング失敗イベント通知を送信するように SES を設定する必要があります。レンダリング失敗イベント通知を受信した場合、無効なコンテンツが含まれていたメッセージを確認し、問題を修正して、もう一度メッセージを送信できます。

このセクションの手順はオプションですが、強くお勧めします。

レンダリング失敗イベント通知を設定するには

1. Amazon SNS トピックを作成する。手続きについては、Amazon Simple Notification Service デベロッパーガイドのトピックの作成を参照してください。
2. Amazon SNS トピックを購読します。たとえば、E メールでレンダリング失敗通知を受信する場合は、そのトピックを E メールエンドポイント (E メールアドレス) で購読する必要があります。

   手順については、Amazon Simple Notification Service デベロッパーガイドのトピックを購読するを参照してください。
3. 「the section called “Amazon SNS 送信先のセットアップ”」の手順を完了して、設定セットをセットアップし、レンダリング失敗イベントを Amazon SNS トピックに発行します。

(オプション) パート 2: E メールテンプレートを作成する

保存されたテンプレートを使用する場合は、このセクションでは、SES `CreateEmailTemplate` v2 API オペレーションを使用してテンプレートを作成する方法を示します。インラインテンプレートを使用する場合は、このステップをスキップできます。

この手順は、 AWS CLIがすでにインストールされ、設定されていることを前提としています。のインストールと設定の詳細については AWS CLI、「 AWS Command Line Interface ユーザーガイド」を参照してください。

テンプレートを作成するには

1. テキストエディタで、新しいファイルを作成し、必要に応じてカスタマイズする次のコードを貼り付けます。

```
{
    "TemplateName": "MyTemplate",
    "TemplateContent": {
        "Subject": "Greetings, {{name}}!",
        "Text": "Dear {{name}},\r\nYour favorite animal is {{favoriteanimal}}.",
        "Html": "<h1>Hello {{name}},</h1><p>Your favorite animal is
 {{favoriteanimal}}.</p>"
    }
}
```

   このコードには次のプロパティが含まれています。

- TemplateName – Templateリソースの名前。E メールを送信する場合に、この名前を参照します。
- TemplateContent – 次の属性のコンテナ。
  - SubjectPart – E メールの件名。このプロパティには、置換タグが含まれる場合があります。これらのタグは以下のフォーマットを使用します: {{tagname}}。E メールを送信するとき、各宛先の tagname に対する値を指定できます。
  - HtmlPart – E メールの HTML 本文。このプロパティには、置換タグが含まれる場合があります。前述の例には、{{name}} および {{favoriteanimal}} の 2 つのタグが含まれます。
  - TextPart – E メールのテキスト本文。E メールクライアントに HTML コンテンツが表示されない受信者には、このバージョンの E メールが表示されます。このプロパティには、代替タグが含まれる場合もあります。

2. 上記の例をニーズに合わせてカスタマイズし、ファイルを *mytemplate.json* として保存します。
3. コマンドラインで、次のコマンドを入力して v2 API CreateEmailTemplate オペレーションを使用して新しいテンプレートを作成します。

```
aws sesv2 create-email-template --cli-input-json file://mytemplate.json
```

パート 3: パーソナライズされた E メールを送信する

次の 2 つの SES v2 API オペレーションを使用して、保存されたテンプレートまたはインラインテンプレートを使用して E メールを送信できます。

- SendEmail オペレーションは、カスタマイズされた E メールを 1 つの送信先オブジェクトに送信するのに役立ちます。v2 API Destination オブジェクトには、ToAddresses、CcAddresses、および BccAddresses プロパティを含めることができます。これらは任意の組み合わせで使用でき、同じ E メールを受信する 1 つ以上の E メールアドレスを含めることができます。
- SendBulkEmail オペレーションは、v2 API への 1 回の呼び出しで複数の送信先オブジェクトに一意の E メールを送信する場合に便利です。

このセクションでは、 を使用して AWS CLI 、これら両方の送信オペレーションを使用してテンプレート化された E メールを送信する方法の例を示します。

テンプレート化された E メールを 1 つの送信先オブジェクトに送信する

SendEmail オペレーションを使用して、1 つの送信先オブジェクトで定義された 1 人以上の受信者に E メールを送信できます。Destination オブジェクトにあるすべての受信者に、同じ E メールが届きます。

テンプレート化された E メールを 1 つの送信先オブジェクトに送信するには

1. 保存済みテンプレートを使用するかインラインテンプレートを使用するかに応じて、テキストエディタに貼り付けるコード例を選択し、必要に応じてカスタマイズします。

   Stored template code example

   前のステップで作成したテンプレート MyTemplate が TemplateNameパラメータの値として参照されていることに注意してください。

```
{
    "FromEmailAddress": "Mary Major <mary.major@example.com>",
    "Destination": {
        "ToAddresses": [
            "alejandro.rosalez@example.com", "jimmy.jet@example.com"
        ]
    },
    "Content": {
        "Template": {
            "TemplateName": "MyTemplate",
            "TemplateData": "{ \"name\":\"Alejandro\", \"favoriteanimal\":
 \"alligator\" }"
        }
    },
    "ConfigurationSetName": "ConfigSet"
}
```

   このコードには次のプロパティが含まれています。

   - FromEmailAddress – 送信者の E メールアドレス。
   - 送信先 – ToAddresses、CcAddresses、および BccAddresses プロパティで定義された E メール受信者を含むオブジェクト。これらは任意の組み合わせで使用でき、同じ E メールを受信する 1 つ以上の E メールアドレスを含めることができます。
   - TemplateName – E メールに適用するTemplateリソースの名前。

- TemplateData – キーと値のペアを含むエスケープされた JSON 文字列。キーは、 など、保存されたテンプレートのTemplateContentプロパティで定義された変数に対応します{{name}}。値は、変数を置き換えるコンテンツを表します。
- ConfigurationSetName – E メールを送信するときに使用する設定セットの名前。

> **Note**
>
> レンダリング失敗イベントを Amazon SNS に発行するように設定された設定セットを使用することをお勧めします。詳細については、「the section called “ (オプション) パート 1: 通知を設定する”」を参照してください。

Inline template code example

TemplateContent プロパティ (通常、保存されたテンプレートで定義されます) は、これをインラインテンプレートにする TemplateDataプロパティとともにインラインで定義されていることに注意してください。

```
{
    "FromEmailAddress": "Mary Major <mary.major@example.com>",
    "Destination": {
        "ToAddresses": [
            "alejandro.rosalez@example.com", "jimmy.jet@example.com"
        ]
    },
    "Content": {
        "Template": {
            "TemplateContent": {
                "Subject": "Greetings, {{name}}!",
                "Text": "Dear {{name}},\r\nYour favorite animal is
 {{favoriteanimal}}.",
                "Html": "<h1>Hello {{name}},</h1><p>Your favorite animal is
 {{favoriteanimal}}.</p>"
            },
            "TemplateData": "{ \"name\":\"Alejandro\", \"favoriteanimal\":
 \"alligator\" }"
        }
    },
    "ConfigurationSetName": "ConfigSet"
}
```

このコードには次のプロパティが含まれています。

- FromEmailAddress – 送信者の E メールアドレス。
- 送信先 – ToAddresses、CcAddresses、および BccAddresses プロパティで定義された E メール受信者を含むオブジェクト。これらは任意の組み合わせで使用でき、同じ E メールを受信する 1 つ以上の E メールアドレスを含めることができます。
- TemplateContent – 次の属性のコンテナ。
  - SubjectPart – E メールの件名。このプロパティには、置換タグが含まれる場合があります。これらのタグは以下のフォーマットを使用します: {{tagname}}。E メールを送信するとき、各宛先の tagname に対する値を指定できます。
  - HtmlPart – E メールの HTML 本文。このプロパティには、置換タグが含まれる場合があります。前述の例には、{{name}} および {{favoriteanimal}} の 2 つのタグが含まれます。
  - TextPart – E メールのテキスト本文。E メールクライアントに HTML コンテンツが表示されない受信者には、このバージョンの E メールが表示されます。このプロパティには、代替タグが含まれる場合もあります。
- TemplateData – キーと値のペアを含むエスケープされた JSON 文字列。キーは、など、このファイルのTemplateContentプロパティで定義された変数に対応します{{name}}。値は、変数を置き換えるコンテンツを表します。
- ConfigurationSetName – E メールを送信するときに使用する設定セットの名前。

> **Note**
>
> レンダリング失敗イベントを Amazon SNS に発行するように設定された設定セットを使用することをお勧めします。詳細については、「the section called “ (オプション) パート 1: 通知を設定する”」を参照してください。

2. 上記の例をニーズに合わせてカスタマイズし、ファイルを *myemail.json* として保存します。
3. コマンドラインで、次の v2 API コマンドを入力して E メールを送信します。

```
aws sesv2 send-email --cli-input-json file://myemail.json
```

テンプレート化された E メールを複数の送信先オブジェクトに送信する

SendBulkEmail オペレーションを使用して、SES v2 API への 1 回の呼び出しで複数の送信先オブジェクトに E メールを送信できます。SES は、各Destinationオブジェクトの受信者に一意の E メールを送信します。

テンプレート化された E メールを複数の送信先オブジェクトに送信するには

1. 保存済みテンプレートを使用するかインラインテンプレートを使用するかに応じて、テキストエディタに貼り付けるコード例を選択し、必要に応じてカスタマイズします。

   Stored template code example

   前のステップで作成したテンプレート MyTemplate が TemplateNameパラメータの値として参照されていることに注意してください。

```
{
    "FromEmailAddress": "Mary Major <mary.major@example.com>",
    "DefaultContent": {
        "Template": {
            "TemplateName": "MyTemplate",
            "TemplateData": "{ \"name\":\"friend\", \"favoriteanimal\":\"unknown
\" }"
        }
    },
    "BulkEmailEntries": [
        {
            "Destination": {
                "ToAddresses": [
                    "anaya.iyengar@example.com"
                ]
            },
            "ReplacementEmailContent": {
                "ReplacementTemplate": {
                    "ReplacementTemplateData": "{ \"name\":\"Anaya\",
 \"favoriteanimal\":\"angelfish\" }"
                }
            }
        },
        {
            "Destination": {
                "ToAddresses": [
                    "liu.jie@example.com"
```

```
                ]
            },
            "ReplacementEmailContent": {
                "ReplacementTemplate": {
                    "ReplacementTemplateData": "{ \"name\":\"Liu\",
 \"favoriteanimal\":\"lion\" }"
                }
            }
        },
        {
            "Destination": {
                "ToAddresses": [
                    "shirley.rodriguez@example.com"
                ]
            },
            "ReplacementEmailContent": {
                "ReplacementTemplate": {
                    "ReplacementTemplateData": "{ \"name\":\"Shirley\",
 \"favoriteanimal\":\"shark\" }"
                }
            }
        },
        {
            "Destination": {
                "ToAddresses": [
                    "richard.roe@example.com"
                ]
            },
            "ReplacementEmailContent": {
                "ReplacementTemplate": {
                    "ReplacementTemplateData": "{}"
                }
            }
        }
    ],
    "ConfigurationSetName": "ConfigSet"
}
```

このコードには次のプロパティが含まれています。

- FromEmailAddress – 送信者の E メールアドレス。
- DefaultContent – オブジェクトTemplateNameと オブジェクトを含む JSON TemplateData オブジェクト。

- TemplateName – E メールに適用するTemplateリソースの名前。
- TemplateData – オブジェクトに空の JSON ReplacementEmailContent オブジェクトが ReplacementTemplateDataプロパティ{}に含まれている場合に使用されるキーと値のペアが含まれます。
- BulkEmailEntries – 1 つ以上のDestinationオブジェクトを含む配列。
- 送信先 – ToAddresses、CcAddresses、および BccAddresses プロパティで定義された E メール受信者を含むオブジェクト。これらは任意の組み合わせで使用でき、同じ E メールを受信する 1 つ以上の E メールアドレスを含めることができます。
- ReplacementTemplateData – キーと値のペアを含むエスケープされた JSON 文字列。キーは、 などのテンプレート内の変数に対応します{{name}}。値は E メール内の変数を置き換える内容を表します。(ここでの JSON 文字列が空で、 で示されている場合{}、DefaultContent オブジェクト内の TemplateDataプロパティで定義されたキーと値のペアが使用されます)。
- ConfigurationSetName – E メールを送信するときに使用する設定セットの名前。

> **Note**
>
> レンダリング失敗イベントを Amazon SNS に発行するように設定された設定セットを使用することをお勧めします。詳細については、「the section called “ (オプション) パート 1: 通知を設定する”」を参照してください。

Inline template code example

TemplateContent プロパティ (通常、保存されたテンプレートで定義されます) は、これをインラインテンプレートにする TemplateDataプロパティとともにインラインで定義されていることに注意してください。

```
{
    "FromEmailAddress": "Mary Major <mary.major@example.com>",
    "DefaultContent": {
        "Template": {
            "TemplateContent": {
                "Subject": "Greetings, {{name}}!",
                "Text": "Dear {{name}},\r\nYour favorite animal is
 {{favoriteanimal}}.",
                "Html": "<h1>Hello {{name}},</h1><p>Your favorite animal is
 {{favoriteanimal}}.</p>"
```

```
            },
            "TemplateData": "{ \"name\":\"friend\", \"favoriteanimal\":\"unknown
\" }"
        }
    },
    "BulkEmailEntries": [
        {
            "Destination": {
                "ToAddresses": [
                    "anaya.iyengar@example.com"
                ]
            },
            "ReplacementEmailContent": {
                "ReplacementTemplate": {
                    "ReplacementTemplateData": "{ \"name\":\"Anaya\",
 \"favoriteanimal\":\"angelfish\" }"
                }
            }
        },
        {
            "Destination": {
                "ToAddresses": [
                    "liu.jie@example.com"
                ]
            },
            "ReplacementEmailContent": {
                "ReplacementTemplate": {
                    "ReplacementTemplateData": "{ \"name\":\"Liu\",
 \"favoriteanimal\":\"lion\" }"
                }
            }
        },
        {
            "Destination": {
                "ToAddresses": [
                    "shirley.rodriguez@example.com"
                ]
            },
            "ReplacementEmailContent": {
                "ReplacementTemplate": {
                    "ReplacementTemplateData": "{ \"name\":\"Shirley\",
 \"favoriteanimal\":\"shark\" }"
                }
            }
```

```
        },
        {
            "Destination": {
                "ToAddresses": [
                    "richard.roe@example.com"
                ]
            },
            "ReplacementEmailContent": {
                "ReplacementTemplate": {
                    "ReplacementTemplateData": "{}"
                }
            }
        }
    ],
    "ConfigurationSetName": "ConfigSet"
}
```

このコードには次のプロパティが含まれています。

- FromEmailAddress – 送信者の E メールアドレス。
- DefaultContent – オブジェクトTemplateContentと オブジェクトを含む JSON TemplateData オブジェクト。
- TemplateContent – 次の属性のコンテナ。
  - SubjectPart – E メールの件名。このプロパティには、置換タグが含まれる場合があります。これらのタグは以下のフォーマットを使用します: {{tagname}}。E メールを送信するとき、各宛先の tagname に対する値を指定できます。
  - HtmlPart – E メールの HTML 本文。このプロパティには、置換タグが含まれる場合があります。前述の例には、{{name}} および {{favoriteanimal}} の 2 つのタグが含まれます。
  - TextPart – E メールのテキスト本文。E メールクライアントに HTML コンテンツが表示されない受信者には、このバージョンの E メールが表示されます。このプロパティには、代替タグが含まれる場合もあります。
- TemplateData – オブジェクトに空の JSON ReplacementEmailContent オブジェクトが ReplacementTemplateDataプロパティ{}に含まれている場合に使用されるキーと値のペアが含まれます。
- BulkEmailEntries – 1 つ以上のDestinationオブジェクトを含む配列。

- 送信先 – ToAddresses、CcAddresses、および BccAddresses プロパティで定義された E メール受信者を含むオブジェクト。これらは任意の組み合わせで使用でき、同じ E メールを受信する 1 つ以上の E メールアドレスを含めることができます。
- ReplacementTemplateData – キーと値のペアを含むエスケープされた JSON 文字列。キーは、 など、このファイルのTemplateContentプロパティで定義された変数に対応します{{name}}。値は E メール内の変数を置き換える内容を表します。(ここでの JSON 文字列が空で、 で示されている場合は{}、 DefaultContent オブジェクト内の TemplateDataプロパティで定義されたキーと値のペアが使用されます)。
- ConfigurationSetName – E メールを送信するときに使用する設定セットの名前。

> **Note**
>
> レンダリング失敗イベントを Amazon SNS に発行するように設定された設定セットを使用することをお勧めします。詳細については、「the section called “ (オプション) パート 1: 通知を設定する”」を参照してください。

2. 前のステップのコードの値をニーズに合わせて変更し、ファイルを *mybulkemail.json* として保存します。
3. コマンドラインで、次の v2 API コマンドを入力して一括 E メールを送信します。

```
aws sesv2 send-bulk-email --cli-input-json file://mybulkemail.json
```

E メールの高度なパーソナライズ

ストアドテンプレートを使用している場合、つまり SES v2 API で オペレーションを使用して Amazon SES にTemplateリソースを作成した場合、Handlebars システムを利用して、ネストされた属性、配列の反復、基本的な条件ステートメント、インラインパーシャルの作成などの高度な機能を含むテンプレートを作成できます。 CreateEmailTemplateこのセクションでは、これらの機能の例を示します。

このセクションに記載されている機能の他にも、Handlebars にはさまざまな機能を提供します。詳細については、handlebarsjs.com の「Built-In Helpers」を参照してください。

> **Note**
>
> SES は、メッセージの HTML テンプレートをレンダリングするときに HTML コンテンツをエスケープしません。つまり、コンタクトフォームなどのユーザー入力データを含める場合は、クライアント側でエスケープする必要があります。

トピック

- 入れ子の属性の解析
- リストに対する反復処理
- 基本的な条件ステートメントの使用
- インラインパーシャルの作成

入れ子の属性の解析

Handlebars には入れ子パスのサポートが含まれています。入れ子パスを使うことで、複雑な顧客データを簡単に整理し、E メールテンプレート内でそのデータを参照できるようになります。

たとえば、受信者のデータをいくつかの一般的なカテゴリで整理し、各カテゴリに詳細な情報を含めることができます。以下のコード例は、このような構造を 1 人の受信者で実現しています。

```
{
  "meta":{
    "userId":"51806220607"
  },
  "contact":{
    "firstName":"Anaya",
    "lastName":"Iyengar",
    "city":"Bengaluru",
    "country":"India",
    "postalCode":"560052"
  },
  "subscription":[
    {
      "interest":"Sports"
    },
    {
      "interest":"Travel"
    },
    {
```

```
      "interest":"Cooking"
    }
  ]
}
```

E メールテンプレートで入れ子の属性を参照するには、親属性の名前、ピリオド (.)、値を含める属性の名前の順に指定します。たとえば前述の例で示したデータ構造を使用し、それぞれの受信者のファーストネームを E メールテンプレートに含める場合は、次のテキストを E メールテンプレートに含めます。`Hello {{contact.firstName}}!`

Handlebars は、複数の階層にわたって入れ子になったパスを解析できます。つまり、テンプレートのデータ構造を柔軟に設定することができます。

リストに対する反復処理

eachヘルパー関数は、配列内の項目を反復処理します。以下のコード例の E メールテンプレートでは、eachヘルパー関数を使用して、それぞれの受信者の関心を項目別に記載したリストを作成しています。

```
{
  "Template": {
    "TemplateName": "Preferences",
    "SubjectPart": "Subscription Preferences for {{contact.firstName}}
 {{contact.lastName}}",
    "HtmlPart": "<h1>Your Preferences</h1>
                <p>You have indicated that you are interested in receiving
                  information about the following subjects:</p>
                <ul>
                  {{#each subscription}}
                    <li>{{interest}}</li>
                  {{/each}}
                </ul>
                <p>You can change these settings at any time by visiting
                   the <a href=https://www.example.com/prefererences/i.aspx?
id={{meta.userId}}>
                   Preference Center</a>.</p>",
    "TextPart": "Your Preferences\n\nYou have indicated that you are interested in
                receiving information about the following subjects:\n
                {{#each subscription}}
                  - {{interest}}\n
                {{/each}}
                \nYou can change these settings at any time by
                visiting the Preference Center at
```

```
                https://www.example.com/prefererences/i.aspx?id={{meta.userId}}"
  }
}
```

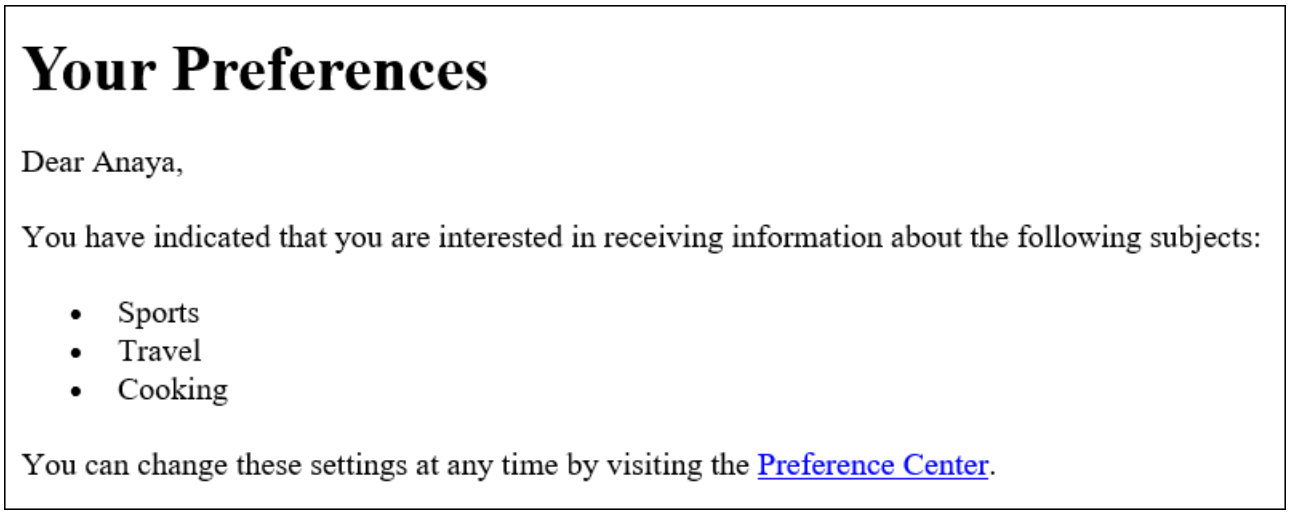
> **Important**
>
> 上記のコード例の HtmlPartおよびTextPart属性には、読みやすくするための改行が含まれています。テンプレートの JSON ファイルでは、これらの値に改行を含めることはできません。この例を独自の JSON ファイルにコピーして貼り付ける場合は、HtmlPart およびTextPartセクションの改行と不要なスペースを削除してから次に進んでください。

テンプレートを作成したら、SendEmailまたはSendBulkEmailオペレーションを使用し、このテンプレートを使って受信者に E メールを送信します。Interestsオブジェクトに値が少なくとも 1 つあれば、それぞれの受信者は、項目別の関心リストを含む E メールを受信します。以下の例は、上記のテンプレートを使用した複数の受信者への E メール送信に使用できる JSON ファイルを示しています。

```
{
  "Source":"Sender Name <sender@example.com>",
  "Template":"Preferences",
  "Destinations":[
    {
      "Destination":{
        "ToAddresses":[
          "anaya.iyengar@example.com"
        ]
      },
      "ReplacementTemplateData":"{\"meta\":{\"userId\":\"51806220607\"},\"contact\":
{\"firstName\":\"Anaya\",\"lastName\":\"Iyengar\"},\"subscription\":[{\"interest\":
\"Sports\"},{\"interest\":\"Travel\"},{\"interest\":\"Cooking\"}]}"
      },
    {
      "Destination":{
        "ToAddresses":[
          "shirley.rodriguez@example.com"
        ]
      },
      "ReplacementTemplateData":"{\"meta\":{\"userId\":\"1981624758263\"},\"contact\":
{\"firstName\":\"Shirley\",\"lastName\":\"Rodriguez\"},\"subscription\":[{\"interest\":
\"Technology\"},{\"interest\":\"Politics\"}]}"
```

```
    }
  ],
  "DefaultTemplateData":"{\"meta\":{\"userId\":\"\"},\"contact\":{\"firstName\":
\"Friend\",\"lastName\":\"\"},\"subscription\":[]}"
}
```

上記の例でリストされた受信者にSendBulkEmail オペレーションを使用して E メールを送信すると、受信者には以下の画像のようなメッセージが届きます。



Your Preferences

Dear Anaya,

You have indicated that you are interested in receiving information about the following subjects:

- Sports
- Travel
- Cooking

You can change these settings at any time by visiting the Preference Center.

基本的な条件ステートメントの使用

このセクションの内容は、前のセクションで説明した例に基づいています。前のセクションの例では、eachヘルパーを使用して関心のリストを反復処理しています。ただし、関心が指定されていない受信者には、空白のリストが記載された E メールが届くことになります。{{if}}ヘルパーを使用すると、テンプレート内のデータに特定の属性が存在する場合に、E メールに異なるフォーマットを適用することができます。以下のコードでは、{{if}} ヘルパーを使用して、Subscription 配列に何らかの値が含まれている場合に、前のセクションの箇条書きリストを表示しています。配列が空の場合、別のテキストブロックが表示されます。

```
{
  "Template": {
    "TemplateName": "Preferences2",
    "SubjectPart": "Subscription Preferences for {{contact.firstName}}
 {{contact.lastName}}",
    "HtmlPart": "<h1>Your Preferences</h1>
                <p>Dear {{contact.firstName}},</p>
                {{#if subscription}}
                  <p>You have indicated that you are interested in receiving
                    information about the following subjects:</p>
```

```
                    <ul>
                    {{#each subscription}}
                      <li>{{interest}}</li>
                    {{/each}}
                    </ul>
                    <p>You can change these settings at any time by visiting
                      the <a href=https://www.example.com/prefererences/i.aspx?
id={{meta.userId}}>
                      Preference Center</a>.</p>
                {{else}}
                  <p>Please update your subscription preferences by visiting
                    the <a href=https://www.example.com/prefererences/i.aspx?
id={{meta.userId}}>
                    Preference Center</a>.
                {{/if}}",
    "TextPart": "Your Preferences\n\nDear {{contact.firstName}},\n\n
                {{#if subscription}}
                  You have indicated that you are interested in receiving
                  information about the following subjects:\n
                  {{#each subscription}}
                    - {{interest}}\n
                  {{/each}}
                  \nYou can change these settings at any time by visiting the
                  Preference Center at https://www.example.com/prefererences/i.aspx?
id={{meta.userId}}.
                {{else}}
                  Please update your subscription preferences by visiting the
                  Preference Center at https://www.example.com/prefererences/i.aspx?
id={{meta.userId}}.
                {{/if}}"
  }
}
```

Important

上記のコード例のHtmlPartおよびTextPart属性には、読みやすくするための改行が含まれています。テンプレートの JSON ファイルでは、これらの値に改行を含めることはできません。この例を独自の JSON ファイルにコピーして貼り付ける場合は、HtmlPartおよびTextPartセクションの改行と不要なスペースを削除してから次に進んでください。

以下の例は、上記のテンプレートを使用した複数の受信者への E メール送信に使用できる JSON ファイルを示しています。

```
{
  "Source":"Sender Name <sender@example.com>",
  "Template":"Preferences2",
  "Destinations":[
    {
      "Destination":{
        "ToAddresses":[
          "anaya.iyengar@example.com"
        ]
      },
      "ReplacementTemplateData":"{\"meta\":{\"userId\":\"51806220607\"},\"contact\":
{\"firstName\":\"Anaya\",\"lastName\":\"Iyengar\"},\"subscription\":[{\"interest\":
\"Sports\"},{\"interest\":\"Cooking\"}]}"
      },
    {
      "Destination":{
        "ToAddresses":[
          "shirley.rodriguez@example.com"
        ]
      },
      "ReplacementTemplateData":"{\"meta\":{\"userId\":\"1981624758263\"},\"contact\":
{\"firstName\":\"Shirley\",\"lastName\":\"Rodriguez\"}}"
    }
  ],
  "DefaultTemplateData":"{\"meta\":{\"userId\":\"\"},\"contact\":{\"firstName\":
\"Friend\",\"lastName\":\"\"},\"subscription\":[]}"
}
```

この例の場合、テンプレートのデータに関心リストが含まれている受信者は、前のセクションで示した例と同じ E メールを受け取ります。一方、テンプレートデータに関心が含まれていない受信者には、以下の画像のような E メールが届きます。

インラインパーシャルの作成

インラインパーシャルを使用して、文字列の繰り返しを含むテンプレートを簡素化することができます。たとえば、テンプレートの先頭に以下のコードを追加することで、受信者のファーストネームと (存在する場合は) ラストネームを含むインラインパーシャルを作成できます。

```
{{#* inline \"fullName\"}}{{firstName}}{{#if lastName}} {{lastName}}{{/if}}{{/
inline}}\n
```

> **Note**
>
> テンプレートの内容と\n ブロックを分離するためには、改行文字 ({{inline}}) が必要です。最終的な出力には改行は表示されません。

fullNameパーシャルを作成したら、以下の例のように、大なり記号 (>)、スペース、パーシャルの名前を指定することで ({{> fullName}})、このパーシャルをテンプレートの任意の場所に含めることができます。インラインパーシャルは、メールの特定部分をまたがって適用されません。たとえば E メールの HTML バージョンとテキストバージョンに同じインラインパーシャルを使用する場合、HtmlPartおよびTextPartセクションの両方に、このインラインパーシャルを定義する必要があります。

インラインパーシャルを、配列を反復処理する際に使用することもできます。以下のコードを使用して、fullNameインラインパーシャルを使用するテンプレートを作成できます。この例では、受信者の名前とその他の名前の配列の両方に、インラインパーシャルが適用されています。

```
{
  "Template": {
    "TemplateName": "Preferences3",
    "SubjectPart": "{{firstName}}'s Subscription Preferences",
    "HtmlPart": "{{#* inline \"fullName\"}}
                  {{firstName}}{{#if lastName}} {{lastName}}{{/if}}
                {{/inline~}}\n
                <h1>Hello {{> fullName}}!</h1>
                <p>You have listed the following people as your friends:</p>
                <ul>
                {{#each friends}}
                  <li>{{> fullName}}</li>
                {{/each}}</ul>",
    "TextPart": "{{#* inline \"fullName\"}}
```

```
                  {{firstName}}{{#if lastName}} {{lastName}}{{/if}}
                {{/inline~}}\n
                Hello {{> fullName}}! You have listed the following people
                as your friends:\n
                {{#each friends}}
                  - {{> fullName}}\n
                {{/each}}"
  }
}
```

> **Important**
>
> 上記のコード例のHtmlPartおよびTextPart属性には、読みやすくするための改行が含まれています。テンプレートの JSON ファイルでは、これらの値に改行を含めることはできません。この例を独自の JSON ファイルにコピーして貼り付ける場合は、これらのセクションの改行と不要なスペースを削除してください。

E メールテンプレートの管理

E メールテンプレートの作成に加えて、Amazon SES v2 API を使用して既存のテンプレートを更新または削除したり、既存のテンプレートをすべて一覧表示したり、テンプレートの内容を表示したりすることもできます。

このセクションでは、 AWS CLI を使用して SES テンプレートに関連するタスクを実行する手順について説明します。

> **Note**
>
> これらの手順では、 AWS CLIがすでにインストールされ、設定されていることを前提としています。のインストールと設定の詳細については AWS CLI、「AWS Command Line Interface ユーザーガイド」を参照してください。

E メールテンプレートの一覧表示

ListEmailTemplate SES v2 API オペレーションを使用して、既存のすべての E メールテンプレートのリストを表示できます。

E メールテンプレートの一覧を表示するには

- コマンドラインで以下のコマンドを入力します。

```
aws sesv2 list-email-templates
```

現在のリージョンの SES アカウントに既存の E メールテンプレートがある場合、このコマンドは次の例のようなレスポンスを返します。

```
{
    "TemplatesMetadata": [
        {
            "Name": "SpecialOffers",
            "CreatedTimestamp": "2020-08-05T16:04:12.640Z"
        },
        {
            "Name": "NewsAndUpdates",
            "CreatedTimestamp": "2019-10-03T20:03:34.574Z"
        }
    ]
}
```

テンプレートをまだ作成していない場合、このコマンドはメンバーのいない`TemplatesMetadata`オブジェクトを返します。

特定の E メール テンプレートの内容の表示

SES v2 API `GetEmailTemplate` オペレーションを使用して、特定の E メールテンプレートの内容を表示できます。

E メールテンプレートの内容を表示するには

- コマンドラインで以下のコマンドを入力します。

```
aws sesv2 get-email-template --template-name MyTemplate
```

上記のコマンドで、*`MyTemplate`*を、表示したいテンプレートの名前に置き換えます。

指定したテンプレート名が SES アカウントに存在するテンプレートと一致する場合、このコマンドは次の例のようなレスポンスを返します。

```
{
    "Template": {
        "TemplateName": "TestMessage",
        "SubjectPart": "Amazon SES Test Message",
        "TextPart": "Hello! This is the text part of the message.",
        "HtmlPart": "<html>\n<body>\n<h2>Hello!</h2>\n<p>This is the HTML part of
 the message.</p></body>\n</html>"
    }
}
```

指定したテンプレート名が SES アカウントに存在するテンプレートと一致しない場合、コマンドはNotFoundExceptionエラーを返します。

E メールテンプレートの削除

SES v2 API DeleteEmailTemplate オペレーションを使用して、特定の E メールテンプレートを削除できます。

E メールテンプレートを削除する

- コマンドラインで以下のコマンドを入力します。

```
aws sesv2 delete-email-template --template-name MyTemplate
```

前述のコマンドで、*MyTemplate* を削除するテンプレート名で置き換えます。

このコマンドは出力を提供しません。GetTemplate オペレーションを使用して、テンプレートが削除されたことを確認できます。

E メールテンプレートの更新

SES v2 API UpdateEmailTemplate オペレーションを使用して、既存の E メールテンプレートを更新できます。たとえば、このオペレーションは、E メールテンプレートの件名を変更する場合や、メッセージの本文を変更する必要がある場合に役立ちます。

E メールテンプレートを更新する

1. GetEmailTemplate コマンドを使用し、コマンドラインに次のコマンドを入力して、既存のテンプレートを取得できます。

```
aws sesv2 get-email-template --template-name MyTemplate
```

上記のコマンドで、*MyTemplate*を、テンプレートの更新名に置き換えます。

指定したテンプレート名が SES アカウントに存在するテンプレートと一致する場合、このコマンドは次の例のようなレスポンスを返します。

```
{
    "Template": {
        "TemplateName": "TestMessage",
        "SubjectPart": "Amazon SES Test Message",
        "TextPart": "Hello! This is the text part of the message.",
        "HtmlPart": "<html>\n<body>\n<h2>Hello!</h2>\n<p>This is the HTML part of
 the message.</p></body>\n</html>"
    }
}
```

2. テキストエディタで新規ファイルを作成します。前のコマンドの出力をファイルに貼り付けます。
3. 必要に応じてテンプレートを変更します。省略した行は、テンプレートから削除されます。たとえば、テンプレートの SubjectPart を変更したい場合のみ、TextPart および HtmlPart のプロパティを含める必要があります。

   終了したら、update_template.jsonとしてファイルを保存します。
4. コマンドラインで以下のコマンドを入力します。

```
aws sesv2 update-email-template --cli-input-json file://path/to/
update_template.json
```

前述の上記のコマンドで、*path/to/update_template.json* を、前のステップで作成した update_template.json ファイルのパスに置き換えます。

テンプレートが正常に更新された場合、このコマンドは出力を提供しません。GetEmailTemplate オペレーションを使用して、テンプレートが更新されたことを確認できます。

指定したテンプレートが存在しない場合、このコマンドは TemplateDoesNotExist エラーを返します。テンプレートに TextPart または HtmlPart のプロパティ (またはその両方) が含まれていない場合、このコマンドは InvalidParameterValue エラーを返します。

AWS SDK を使用して Amazon SES 経由で E メールを送信する

AWS SDK を使用して Amazon SES 経由で E メールを送信できます。 AWS SDKsは複数のプログラミング言語で使用できます。詳細については、Tools for Amazon Web Services を参照してください。

前提条件

次のセクションのコードサンプルを完了するには、次の前提条件を完了する必要があります。

- まだ行っていない場合は、「Amazon Simple Email Service を設定する」の作業を完了してください。
- Amazon SES で E メールアドレスを検証する- Amazon SES で E メールを送信するには、送信者の E メールアドレスを所有していることを検証する必要があります。アカウントが Amazon SES サンドボックスにまだある場合は、受信者の E メールアドレスも検証する必要があります。E メールアドレスを検証するには、Amazon SES コンソールを使用することをお勧めします。詳細については、「Eメールアドレス ID の作成」を参照してください。
- AWS 認証情報の取得 — SDK を使用して Amazon SES にアクセスするには、 AWS アクセスキー ID と AWS シークレットアクセスキーが必要です。認証情報を取得するするには、 AWS Management Consoleの「セキュリティの認証情報」のページを参照してください。認証情報の詳細については、「Amazon SES 認証情報の種類」を参照してください。
- 共有認証情報ファイルの作成 — このセクションのサンプルコードが正常に機能するためには、共有認証情報ファイルを作成する必要があります。詳細については、「AWS SDK を使用して Amazon SES 経由で E メールを送信するときに使用する共有認証情報ファイルの作成」を参照してください。

コードの例

> **Important**
>
> 次のチュートリアルでは、受信を確認できるように自分宛に E メールを送信します。さらに詳しい実験や負荷テストには、Amazon SES メールボックスシミュレーターを使用してくだ

さい。メールボックスシミュレーターに送信される E メールは、送信クォータに加算されず、バウンス率や苦情率の計算にも含まれません。詳細については、手動でメールボックスシミュレーターを使用するを参照ください。

.NET

以下の手順は、Visual Studioと AWS SDK for .NETを使用して、Amazon SES 経由で E メールを送信する方法を示しています。

このソリューションは次のコンポーネントを使用してテスト済みです。

- Microsoft Visual Studio コミュニティ 2017、バージョン 15.4.0。
- Microsoft .NET Framework バージョン 4.6.1.
- NuGet を使用してインストールされた AWSSDK.Core パッケージ (バージョン 3.3.19)。
- NuGet を使用してインストールされた AWSSDK.SimpleEmail パッケージ (バージョン 3.3.6.1)。

開始する前に、次のタスクを実行します。

- Visual Studio のインストール - Visual Studio は https://www.visualstudio.com/から入手可能です。

を使用して E メールを送信するには AWS SDK for .NET

1. 以下のステップを実行して、新しいプロジェクトを作成します。
   a. Visual Studio を起動します。
   b. [ファイル] メニューで [New]、[Project] の順に選択します。
   c. [New Project] ウィンドウの左側のパネルで、[Installed]、[Visual C#] の順に展開します。
   d. 右側のパネルで、[Console App (.NET Framework)] を選択します。
   e. 名前に **`AmazonSESSample`** と入力し、OK を選択します。
2. 次のステップを実行して、NuGet を使用して Amazon SES パッケージをソリューションに含めます。

a. ソリューションエクスプローラーペインで、プロジェクトを右クリックして、コンテキストメニューの [NuGet パッケージの管理] を選択します。

b. [NuGet: AmazonSESSample] タブで、[参照] を選択します。

c. 検索ボックスに [**AWSSDK.SimpleEmail**] と入力します

d. [AWSSDK.SimpleEmail] パッケージを選択し、[インストール] を選択します。

e. 変更のプレビューウィンドウで、OK を選択します。

3. [Program.cs] タブで、次のコードを貼り付けます。

```
using Amazon;
using System;
using System.Collections.Generic;
using Amazon.SimpleEmail;
using Amazon.SimpleEmail.Model;

namespace AmazonSESSample
{
    class Program
    {
        // Replace sender@example.com with your "From" address.
        // This address must be verified with Amazon SES.
        static readonly string senderAddress = "sender@example.com";

        // Replace recipient@example.com with a "To" address. If your account
        // is still in the sandbox, this address must be verified.
        static readonly string receiverAddress = "recipient@example.com";

        // The configuration set to use for this email. If you do not want to
 use a
        // configuration set, comment out the following property and the
        // ConfigurationSetName = configSet argument below.
        static readonly string configSet = "ConfigSet";

        // The subject line for the email.
        static readonly string subject = "Amazon SES test (AWS SDK for .NET)";

        // The email body for recipients with non-HTML email clients.
        static readonly string textBody = "Amazon SES Test (.NET)\r\n"
                                        + "This email was sent through Amazon
 SES "
                                        + "using the AWS SDK for .NET.";
```

```
        // The HTML body of the email.
        static readonly string htmlBody = @"<html>
<head></head>
<body>
  <h1>Amazon SES Test (AWS SDK for .NET)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/ses/'>Amazon SES</a> using the
    <a href='https://aws.amazon.com/sdk-for-net/'> AWS SDK for .NET</a>.</p>
</body>
</html>";

        static void Main(string[] args)
        {
            // Replace USWest2 with the AWS Region you're using for Amazon SES.
            // Acceptable values are EUWest1, USEast1, and USWest2.
            using (var client = new
 AmazonSimpleEmailServiceClient(RegionEndpoint.USWest2))
            {
                var sendRequest = new SendEmailRequest
                {
                    Source = senderAddress,
                    Destination = new Destination
                    {
                        ToAddresses =
                        new List<string> { receiverAddress }
                    },
                    Message = new Message
                    {
                        Subject = new Content(subject),
                        Body = new Body
                        {
                            Html = new Content
                            {
                                Charset = "UTF-8",
                                Data = htmlBody
                            },
                            Text = new Content
                            {
                                Charset = "UTF-8",
                                Data = textBody
                            }
                        }
                    },
```

```
                    // If you are not using a configuration set, comment
                    // or remove the following line
                    ConfigurationSetName = configSet
                };
                try
                {
                    Console.WriteLine("Sending email using Amazon SES...");
                    var response = client.SendEmail(sendRequest);
                    Console.WriteLine("The email was sent successfully.");
                }
                catch (Exception ex)
                {
                    Console.WriteLine("The email was not sent.");
                    Console.WriteLine("Error message: " + ex.Message);

                }
            }

            Console.Write("Press any key to continue...");
            Console.ReadKey();
        }
    }
}
```

4. コードエディタで、以下の作業を行います。

   - *sender@example.com* を "差出人:" の E メールアドレスに置き換えます。このアドレスは確認する必要があります。詳細については、「検証済みID」を参照してください。
   - *recipient@example.com* を "宛先:" のアドレスに置き換えます。アカウントがサンドボックスにまだある場合は、このアドレスも確認する必要があります。
   - *ConfigSet*を、この E メールを送信するときに使用する設定セットの名前に置き換えます。
   - *USWest2* を、Amazon SES を使用して E メールを送信するために使用する AWS リージョン エンドポイントの名前に置き換えます。Amazon SES を使用できるリージョンのリストについては、「AWS 全般のリファレンス」の「Amazon Simple Email Service (Amazon SES)」を参照してください。

   終了したら、Program.csを保存します。

5. 次の手順に従ってアプリケーションをビルドおよび実行します。

a. [Build] メニューの [Build Solution] を選択します。
b. [Debug] メニューの [Start Debugging] を選択します。コンソールウィンドウが表示されます。

6. コンソールの出力を確認します。E メールが正常に送信されると、コンソールに "The email was sent successfully." と表示されます
7. E メールが正常に送信されたら、受信者アドレスの E メールクライアントにサインインします。送信した E メールメッセージを確認します。

Java

次の手順では、Eclipse IDE for Java EE Developers と を使用して AWS SDK プロジェクトAWS Toolkit for Eclipseを作成し、Java コードを変更して Amazon SES 経由で E メールを送信する方法を示します。

開始する前に、次のタスクを実行します。

- Eclipse のインストール - Eclipse はhttps://www.eclipse.org/downloadsからダウンロードできます。このチュートリアルのコードは、バージョン 1.8 の Java Runtime Environment を実行する Eclipse Neon.3 (バージョン 4.6.3) でテスト済みです。
- のインストール AWS Toolkit for Eclipse— Eclipse のインストール AWS Toolkit for Eclipse に を追加する手順については、https://aws.amazon.com/eclipse を参照してください。このチュートリアルのコードはバージョン 2.3.1 の AWS Toolkit for Eclipseでテスト済みです。

を使用して E メールを送信するには AWS SDK for Java

1. 次の手順を実行して、Eclipse AWS で Java プロジェクトを作成します。

   a. Eclipse を起動します。
   b. [File] メニューで [New]、[Other] の順に選択します。[New] ウィンドウで、AWSフォルダを展開し、[AWS Java Project] を選択します。
   c. 新しい Java AWS プロジェクトダイアログボックスで、次の操作を行います。

      i. [Project name] に、プロジェクト名を入力します。
      ii. AWS SDK for Java のサンプルで、Amazon Simple Email Service JavaMail のサンプルを選択します。
      iii. [Finish] を選択します。

2. Eclipse の [Package Explorer] ペインで、プロジェクトを展開します。
3. プロジェクトの src/main/java フォルダ、com.amazon.aws.samples フォルダの順に展開し、AmazonSESSample.java をダブルクリックします。
4. AmazonSESSample.java の内容全体を次のコードに置き換えます。

```
package com.amazonaws.samples;

import java.io.IOException;

import com.amazonaws.regions.Regions;
import com.amazonaws.services.simpleemail.AmazonSimpleEmailService;
import com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClientBuilder;
import com.amazonaws.services.simpleemail.model.Body;
import com.amazonaws.services.simpleemail.model.Content;
import com.amazonaws.services.simpleemail.model.Destination;
import com.amazonaws.services.simpleemail.model.Message;
import com.amazonaws.services.simpleemail.model.SendEmailRequest;

public class AmazonSESSample {

  // Replace sender@example.com with your "From" address.
  // This address must be verified with Amazon SES.
  static final String FROM = "sender@example.com";

  // Replace recipient@example.com with a "To" address. If your account
  // is still in the sandbox, this address must be verified.
  static final String TO = "recipient@example.com";

  // The configuration set to use for this email. If you do not want to use a
  // configuration set, comment the following variable and the
  // .withConfigurationSetName(CONFIGSET); argument below.
  static final String CONFIGSET = "ConfigSet";

  // The subject line for the email.
  static final String SUBJECT = "Amazon SES test (AWS SDK for Java)";

  // The HTML body for the email.
  static final String HTMLBODY = "<h1>Amazon SES test (AWS SDK for Java)</h1>"
      + "<p>This email was sent with <a href='https://aws.amazon.com/ses/'>"
      + "Amazon SES</a> using the <a href='https://aws.amazon.com/sdk-for-
java/'>"
      + "AWS SDK for Java</a>";
```

```
  // The email body for recipients with non-HTML email clients.
  static final String TEXTBODY = "This email was sent through Amazon SES "
      + "using the AWS SDK for Java.";

  public static void main(String[] args) throws IOException {

    try {
      AmazonSimpleEmailService client =
          AmazonSimpleEmailServiceClientBuilder.standard()
          // Replace US_WEST_2 with the AWS Region you're using for
          // Amazon SES.
            .withRegion(Regions.US_WEST_2).build();
      SendEmailRequest request = new SendEmailRequest()
          .withDestination(
              new Destination().withToAddresses(TO))
          .withMessage(new Message()
              .withBody(new Body()
                  .withHtml(new Content()
                      .withCharset("UTF-8").withData(HTMLBODY))
                  .withText(new Content()
                      .withCharset("UTF-8").withData(TEXTBODY)))
              .withSubject(new Content()
                  .withCharset("UTF-8").withData(SUBJECT)))
          .withSource(FROM)
          // Comment or remove the next line if you are not using a
          // configuration set
          .withConfigurationSetName(CONFIGSET);
      client.sendEmail(request);
      System.out.println("Email sent!");
    } catch (Exception ex) {
      System.out.println("The email was not sent. Error message: "
          + ex.getMessage());
    }
  }
}
```

5. AmazonSESSample.javaで、以下を実際の値に置き換えます。

> **Important**
>
> E メールアドレスでは、大文字と小文字は区別されます。検証したアドレスと完全に一致することを確認してください。

- `SENDER@EXAMPLE.COM` - 「From」E メールアドレスに置き換えます。このアドレスを確認してから、プログラムを実行してください。詳細については、「Amazon SES の検証済みID」を参照してください。
- `RECIPIENT@EXAMPLE.COM` - 「To」E メールアドレスに置き換えます。アカウントがサンドボックスにまだある場合は、このアドレスを使用前に確認する必要があります。詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。
- (オプション)**`us-west-2`** — 米国西部 (オレゴン) 以外の地域で Amazon SES を使用する場合は、これを使用する地域に置き換えます。Amazon SES を使用できるリージョンのリストについては、「AWS 全般のリファレンス」の「Amazon Simple Email Service (Amazon SES)」を参照してください。

6. `AmazonSESSample.java` を保存します。
7. プロジェクトを構築します。[Project]、[Build Project] の順に選択します。

> **Note**
>
> このオプションが無効の場合、自動構築が有効になっている可能性があります。その場合は、このステップをスキップします。

8. プログラムを開始して E メールを送信します。[Run] を選択した後、もう一度 [Run] を選択します。
9. Eclipse でコンソールペインの出力を確認します。E メールが正常に送信されると、コンソールに `"Email sent!"` が表示されます。送信に失敗すると、エラーメッセージが表示されます。
10. E メールが正常に送信されたら、受信者アドレスの E メールクライアントにサインインします。送信した E メールメッセージを確認します。

PHP

このトピックでは、AWS SDK for PHP を使用して Amazon SES 経由で E メールを送信する方法を示します。

開始する前に、次のタスクを実行します。

- PHP をインストールする — PHP は、http://php.net/downloads.php から入手できます。このチュートリアルでは、バージョン 5.5 以上の PHP が必要です。PHP をインストールした後、コマンドプロンプトから PHP を実行できるように環境変数に PHP のパスを追加します。このチュートリアルのコードは PHP 7.2.7 でテスト済みです。
- AWS SDK for PHP バージョン 3 のインストール — ダウンロードとインストールの手順については、 AWS SDK for PHP ドキュメントを参照してください。このチュートリアルのコードは SDK バージョン 3.64.13 でテスト済みです。

を使用して Amazon SES 経由で E メールを送信するには AWS SDK for PHP

1. テキストエディタで amazon-ses-sample.php という名前のファイルを作成します。次のコードを貼り付けます。

```
<?php

// If necessary, modify the path in the require statement below to refer to the
// location of your Composer autoload.php file.
require 'vendor/autoload.php';

use Aws\Ses\SesClient;
use Aws\Exception\AwsException;

// Create an SesClient. Change the value of the region parameter if you're
// using an AWS Region other than US West (Oregon). Change the value of the
// profile parameter if you want to use a profile in your credentials file
// other than the default.
$SesClient = new SesClient([
    'profile' => 'default',
    'version' => '2010-12-01',
    'region'  => 'us-west-2'
]);

// Replace sender@example.com with your "From" address.
// This address must be verified with Amazon SES.
```

```
$sender_email = 'sender@example.com';

// Replace these sample addresses with the addresses of your recipients. If
// your account is still in the sandbox, these addresses must be verified.
$recipient_emails = ['recipient1@example.com','recipient2@example.com'];

// Specify a configuration set. If you do not want to use a configuration
// set, comment the following variable, and the
// 'ConfigurationSetName' => $configuration_set argument below.
$configuration_set = 'ConfigSet';

$subject = 'Amazon SES test (AWS SDK for PHP)';
$plaintext_body = 'This email was sent with Amazon SES using the AWS SDK for
 PHP.' ;
$html_body =  '<h1>AWS Amazon Simple Email Service Test Email</h1>'.
              '<p>This email was sent with <a href="https://aws.amazon.com/
ses/">'.
              'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-
php/">'.
              'AWS SDK for PHP</a>.</p>';
$char_set = 'UTF-8';

try {
    $result = $SesClient->sendEmail([
        'Destination' => [
            'ToAddresses' => $recipient_emails,
        ],
        'ReplyToAddresses' => [$sender_email],
        'Source' => $sender_email,
        'Message' => [
          'Body' => [
              'Html' => [
                  'Charset' => $char_set,
                  'Data' => $html_body,
              ],
              'Text' => [
                  'Charset' => $char_set,
                  'Data' => $plaintext_body,
              ],
          ],
          'Subject' => [
              'Charset' => $char_set,
              'Data' => $subject,
          ],
```

```
        ],
        // If you aren't using a configuration set, comment or delete the
        // following line
        'ConfigurationSetName' => $configuration_set,
    ]);
    $messageId = $result['MessageId'];
    echo("Email sent! Message ID: $messageId"."\n");
} catch (AwsException $e) {
    // output error message if fails
    echo $e->getMessage();
    echo("The email was not sent. Error message: ".$e-
>getAwsErrorMessage()."\n");
    echo "\n";
}
```

2. amazon-ses-sample.phpで、以下を独自の値に置き換えます。

   - **path_to_sdk_inclusion**— をプログラム AWS SDK for PHP に を含めるために必要なパスに置き換えます。詳細については、AWS SDK for PHP ドキュメントを参照してください。
   - **sender@example.com** — Amazon SES で検証した E メールアドレスに置き換えます。詳細については、「検証済みID」を参照してください。Amazon SES では、E メールアドレスの大文字と小文字が区別されます。検証したアドレスと完全に一致するアドレスを入力してください。
   - **recipient1@example.com**、**recipient2@example.com** - 受信者のアドレスに置き換えます。アカウントがサンドボックスにまだある場合は、受取人のアドレスも確認済みである必要があります。詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。検証したアドレスと完全に一致するアドレスを入力してください。
   - (オプション)**ConfigSet** — この E メールを送信する際に設定セットを使用する場合、この値を設定セットの名前で置き換えます。設定セットの詳細については、「Amazon SES の設定セットの使用」を参照してください。
   - (オプション)**us-west-2** — 米国西部 (オレゴン) 以外の地域で Amazon SES を使用する場合は、これを使用する地域に置き換えます。Amazon SES を使用できるリージョンのリストについては、「AWS 全般のリファレンス」の「Amazon Simple Email Service (Amazon SES)」を参照してください。

3. amazon-ses-sample.phpを保存します。

4. プログラムを実行するには、amazon-ses-sample.php と同じディレクトリでコマンドプロンプトを開き、次のコマンドを入力します。

```
$ php amazon-ses-sample.php
```

5. 出力を確認します。E メールが正常に送信されると、コンソールに "Email sent!" が表示されます。送信に失敗すると、エラーメッセージが表示されます。

> **Note**
>
> プログラムの実行時に "cURL error 60: SSL certificate problem" エラーが発生した場合は、AWS SDK for PHP のドキュメントに従って、最新の CA バンドルをダウンロードしてください。次に、amazon-ses-sample.php で、SesClient::factory 配列に以下の行を追加し、ダウンロードした CA バンドルのパスで path_of_certsを置き換えて、プログラムを再実行します。
>
> ```
> 'http' => [
>    'verify' => 'path_of_certs\ca-bundle.crt'
> ]
> ```

6. 受信者のアドレスの E メールクライアントにサインインします。送信した メッセージを確認します。

Ruby

このトピックでは、AWS SDK for Ruby を使用して Amazon SES 経由で E メールを送信する方法を示します。

開始する前に、次のタスクを実行します。

- Ruby のインストール - Ruby は https://www.ruby-lang.org/en/downloads/からダウンロードできます。このチュートリアルのコードは Ruby 1.9.3 でテスト済みです。Ruby をインストールした後、コマンドプロンプトから Ruby を実行できるように環境変数に Ruby のパスを追加します。
- のインストール AWS SDK for Ruby — ダウンロードとインストールの手順については、 AWS SDK for Ruby デベロッパーガイドの「 AWS SDK for Rubyのインストール」を参照してください。このチュートリアルのサンプルコードは AWS SDK for Rubyバージョン 2.9.36 でテスト済みです。

- 共有認証情報ファイルの作成 — このセクションのサンプルコードが正常に機能するためには、共有認証情報ファイルを作成する必要があります。詳細については、「AWS SDK を使用して Amazon SES 経由で E メールを送信するときに使用する共有認証情報ファイルの作成」を参照してください。

を使用して Amazon SES 経由で E メールを送信するには AWS SDK for Ruby

1. テキストエディタで amazon-ses-sample.rb という名前のファイルを作成します。ファイルに次のコードを貼り付けます。

```
require 'aws-sdk'

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = "sender@example.com"

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = "recipient@example.com"

# Specify a configuration set. If you do not want to use a configuration
# set, comment the following variable and the
# configuration_set_name: configsetname argument below.
configsetname = "ConfigSet"

# Replace us-west-2 with the AWS Region you're using for Amazon SES.
awsregion = "us-west-2"

# The subject line for the email.
subject = "Amazon SES test (AWS SDK for Ruby)"

# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (AWS SDK for Ruby)</h1>'\
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  'AWS SDK for Ruby</a>.'

# The email body for recipients with non-HTML email clients.
textbody = "This email was sent with Amazon SES using the AWS SDK for Ruby."

# Specify the text encoding scheme.
```

```
encoding = "UTF-8"

# Create a new SES resource and specify a region
ses = Aws::SES::Client.new(region: awsregion)

# Try to send the email.
begin

  # Provide the contents of the email.
  resp = ses.send_email({
    destination: {
      to_addresses: [
        recipient,
      ],
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody,
        },
        text: {
          charset: encoding,
          data: textbody,
        },
      },
      subject: {
        charset: encoding,
        data: subject,
      },
    },
  source: sender,
  # Comment or remove the following line if you are not using
  # a configuration set
  configuration_set_name: configsetname,
  })
  puts "Email sent!"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts "Email not sent. Error message: #{error}"

end
```

2. amazon-ses-sample.rbで、以下を独自の値に置き換えます。

   - **sender@example.com**— Amazon SES で検証した E メールアドレスに置き換えます。詳細については、「検証済みID」を参照してください。Amazon SES では、E メールアドレスの大文字と小文字が区別されます。検証したアドレスと完全に一致するアドレスを入力してください。
   - **recipient@example.com**— 受信者のアドレスに置き換えます。アカウントがサンドボックスにまだある場合は、このアドレスを使用前に確認する必要があります。詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。検証したアドレスと完全に一致するアドレスを入力してください。
   - (オプション)**us-west-2** — 米国西部 (オレゴン) 以外の地域で Amazon SES を使用する場合は、これを使用する地域に置き換えます。Amazon SES を使用できるリージョンのリストについては、「AWS 全般のリファレンス」の「Amazon Simple Email Service (Amazon SES)」を参照してください。
3. amazon-ses-sample.rbを保存します。
4. プログラムを実行するには、amazon-ses-sample.rb と同じディレクトリでコマンドプロンプトを開き、ruby amazon-ses-sample.rb と入力します
5. 出力を確認します。E メールが正常に送信されると、コンソールに "Email sent!" が表示されます。送信に失敗すると、エラーメッセージが表示されます。
6. 受信者のアドレスの E メールクライアントにサインインします。送信した E メールメッセージを確認します。

Python

このトピックでは、AWS SDK for Python (Boto)を使用して Amazon SES 経由で E メールを送信する方法を示します。

開始する前に、次のタスクを実行します。

- Amazon SES で E メールアドレスを検証する - Amazon SES で E メールを送信するには、送信者の E メールアドレスを所有していることを検証する必要があります。アカウントが Amazon SES サンドボックスにまだある場合は、受信者の E メールアドレスも検証する必要があります。E メールアドレスを検証するには、Amazon SES コンソールを使用することをお勧めします。詳細については、「Eメールアドレス ID の作成」を参照してください。
- 認証情報の取得 AWS — SDK を使用して Amazon SES にアクセスするには、 AWS アクセスキー ID と AWS シークレットアクセスキーが必要です。認証情報を取得するには、 AWS

Management Consoleの「セキュリティの認証情報」のページを参照してください。認証情報の詳細については、「Amazon SES 認証情報の種類」を参照してください。

- Python のインストール - Python はhttps://www.python.org/downloads/からダウンロードできます。このチュートリアルのコードは Python 2.7.6 および Python 3.6.1 でテスト済みです。Python をインストールした後、コマンドプロンプトから Python を実行できるように環境変数に Python のパスを追加します。
- のインストール AWS SDK for Python (Boto) — ダウンロードとインストールの手順については、 AWS SDK for Python (Boto) ドキュメントを参照してください。このチュートリアルのサンプルコードは SDK for Python のバージョン 1.4.4 でテスト済みです。

SDK for Python を使用して Amazon SES 経由で E メールを送信するには

1. テキストエディタでamazon-ses-sample.pyという名前のファイルを作成します。ファイルに次のコードを貼り付けます。

```
import boto3
from botocore.exceptions import ClientError

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
SENDER = "Sender Name <sender@example.com>"

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
RECIPIENT = "recipient@example.com"

# Specify a configuration set. If you do not want to use a configuration
# set, comment the following variable, and the
# ConfigurationSetName=CONFIGURATION_SET argument below.
CONFIGURATION_SET = "ConfigSet"

# If necessary, replace us-west-2 with the AWS Region you're using for Amazon
 SES.
AWS_REGION = "us-west-2"

# The subject line for the email.
SUBJECT = "Amazon SES Test (SDK for Python)"

# The email body for recipients with non-HTML email clients.
BODY_TEXT = ("Amazon SES Test (Python)\r\n"
```

```
             "This email was sent with Amazon SES using the "
             "AWS SDK for Python (Boto)."
            )

# The HTML body of the email.
BODY_HTML = """<html>
<head></head>
<body>
  <h1>Amazon SES Test (SDK for Python)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/ses/'>Amazon SES</a> using the
    <a href='https://aws.amazon.com/sdk-for-python/'> AWS SDK for Python
 (Boto)</a>.</p>
</body>
</html>
            """

# The character encoding for the email.
CHARSET = "UTF-8"

# Create a new SES resource and specify a region.
client = boto3.client('ses',region_name=AWS_REGION)

# Try to send the email.
try:
    #Provide the contents of the email.
    response = client.send_email(
        Destination={
            'ToAddresses': [
                RECIPIENT,
            ],
        },
        Message={
            'Body': {
                'Html': {
                    'Charset': CHARSET,
                    'Data': BODY_HTML,
                },
                'Text': {
                    'Charset': CHARSET,
                    'Data': BODY_TEXT,
                },
            },
            'Subject': {
```

```
                'Charset': CHARSET,
                'Data': SUBJECT,
            },
        },
        Source=SENDER,
        # If you are not using a configuration set, comment or delete the
        # following line
        ConfigurationSetName=CONFIGURATION_SET,
    )
# Display an error if something goes wrong.
except ClientError as e:
    print(e.response['Error']['Message'])
else:
    print("Email sent! Message ID:"),
    print(response['MessageId'])
```

2. amazon-ses-sample.pyで、以下を独自の値に置き換えます。

   - **sender@example.com**— Amazon SES で検証した E メールアドレスに置き換えます。詳細については、「検証済みID」を参照してください。Amazon SES では、E メールアドレスの大文字と小文字が区別されます。検証したアドレスと完全に一致するアドレスを入力してください。
   - **recipient@example.com**— 受信者のアドレスに置き換えます。アカウントがサンドボックスにまだある場合は、このアドレスを使用前に確認する必要があります。詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。検証したアドレスと完全に一致するアドレスを入力してください。
   - (オプション)**us-west-2** — 米国西部 (オレゴン) 以外の地域で Amazon SES を使用する場合は、これを使用する地域に置き換えます。Amazon SES を使用できるリージョンのリストについては、「AWS 全般のリファレンス」の「Amazon Simple Email Service (Amazon SES)」を参照してください。

3. amazon-ses-sample.py を保存します。
4. プログラムを実行するには、amazon-ses-sample.py と同じディレクトリでコマンドプロンプトを開き、python amazon-ses-sample.py と入力します。
5. 出力を確認します。E メールが正常に送信されると、コンソールに "Email sent!" が表示されます。送信に失敗すると、エラーメッセージが表示されます。
6. 受信者のアドレスの E メールクライアントにサインインします。送信した E メールメッセージを確認します。

AWS SDK を使用して Amazon SES 経由で E メールを送信するときに使用する共有認証情報ファイルの作成

次の手順は、ホームディレクトリに認証情報の共有ファイルを作成する方法を示しています。SDK サンプルコードが正常に動作するには、このファイルを作成する必要があります。

1. テキストエディタで新規ファイルを作成します。ファイルに次のコードを貼り付けます。

```
[default]
aws_access_key_id = YOUR_AWS_ACCESS_KEY_ID
aws_secret_access_key = YOUR_AWS_SECRET_ACCESS_KEY
```

2. 先ほど作成したテキストファイルで、 を一意の AWS アクセスキー ID YOUR_AWS_ACCESS_KEY に置き換え、 を一意の AWS シークレットアクセスキーYOUR_AWS_SECRET_ACCESS_KEYに置き換えます。
3. ファイルを保存します。次の表は、オペレーティングシステム別の正しい場所とファイル名を示しています。

使用しているクライアント	ファイルの保存名
Windows	C:\Users\<yourUserName>\.aws\credentials
Linux、macOS、または Unix	~/.aws/credentials

> **Important**
> 認証情報ファイルを保存するときは、ファイル拡張子を含めないでください。

Amazon SES でサポートされているコンテンツのエンコーディング

以下は、参考のために提供されています。

Amazon SES は、次のコンテンツエンコーディングをサポートしています。

- deflate
- gzip

- `identity`

Amazon SES は、RFC 7231 仕様に従って、次の Accept-Encoding ヘッダー形式もサポートしています。

- `Accept-Encoding:deflate,gzip`
- `Accept-Encoding:`
- `Accept-Encoding:*`
- `Accept-Encoding:deflate;q=0.5,gzip;q=1.0`
- `Accept-Encoding:gzip;q=1.0,identity;q=0.5,*;q=0`

Amazon SES およびセキュリティプロトコル

このトピックでは、Amazon SES に接続する際、そして Amazon SES が受信者に E メールを送信する際に使用できるセキュリティプロトコルについて説明します。

E メール送信者から Amazon SES へ

Amazon SES に接続するために使われるセキュリティプロトコルは、以下に説明されるように、Amazon SES API あるいは Amazon SES SMTP インターフェイスの使用によって異なります。

HTTPS

Amazon SES API を使用している場合 (直接または AWS SDK を介して)、すべての通信は Amazon SES HTTPS エンドポイントを介して TLS によって暗号化されます。Amazon SES の HTTPS エンドポイントは、TLS 1.2 および TLS 1.3 をサポートします。

SMTP インターフェイス

SMTP インターフェイスを介して Amazon SES にアクセスする場合、Transport Layer Security (TLS) を使って接続を暗号化する必要があります。TLS は、以前のプロトコルの名前である「Secure Sockets Layer (SSL)」と呼ばれることが多いことに注意してください。

Amazon SES は、TLS で暗号化された接続を確立するために、STARTTLS および TLS ラッパーという 2 つのメカニズムをサポートしています。

- STARTTLS – STARTTLS とは、暗号化されていない接続を暗号化された接続にアップグレードする方法です。STARTTLS には、様々なプロトコルに対応したバージョンがあります。SMTP バー

ジョンは、「RFC 3207」に定義されています。STARTTLS 接続の場合、Amazon SES は TLS 1.2 と TLS 1.3 をサポートします。

- TLS Wrapper – TLS Wrapper (SMTPS またはハンドシェイクプロトコルとも呼ばれる) は、最初に暗号化されていない接続を確立するのではなく、最初から暗号化された接続を開始する方法です。TLS ラッパーを使用する場合、Amazon SES SMTP エンドポイントは TLS ネゴシエーションを実行しません。TLS を使用してエンドポイントに接続し、通信全体で TLS の使用を継続するのはクライアントの役割です。TLS ラッパーは古いプロトコルですが、数多くのクライアントが今もサポートしています。TLS ラッパー接続の場合、Amazon SES は TLS 1.2 および TLS 1.3 をサポートします。

これらの方法を使用した、Amazon SES SMTP インターフェイスへの接続について詳しくは、「Amazon SES SMTP エンドポイントへの接続」を参照してください。

Amazon SES から受信者へ

TLS 1.3 はデフォルトの配信方法であるとはいえ、SES は以前のバージョンの TLS を使用してメールサーバーに E メールを配信できます。

デフォルトで、Amazon SES は便宜的 TLSを使用します。つまり、Amazon SES は常に受信メールサーバーへの安全な接続を確立しようとします。安全な接続を確立できない場合、Amazon SES は平文メッセージを送信します。

この動作は、設定セットを使用することで変更できます。PutConfigurationSetDeliveryOptions API オペレーションを使用して、設定セットの `TlsPolicy` プロパティを `Require` に設定します。この変更には、AWS CLI を使用できます。

設定セットに TLS 接続を要求するよう Amazon SES を設定するには

- コマンドラインで以下のコマンドを入力します。

```
aws sesv2 put-configuration-set-delivery-options --configuration-set-
name MyConfigurationSet --tls-policy REQUIRE
```

前述の例では、*MyConfigurationSet* をお客様の設定セットの名前に置き換えます。

この設定セットを使用して E メールを送信する場合、Amazon SES は、安全な接続を確立できる受信 E メールサーバーに対してのみメッセージを送信します。受信 E メールサーバーへの安全な接続が確立できない場合、Amazon SES はメッセージを破棄します。

エンドツーエンドの暗号化

Amazon SES を使用して、S/MIME または PGP を使用して暗号化されたメッセージを送信できます。これらのプロトコルを使用するメッセージは、送信者によって暗号化されます。これらのコンテンツは、メッセージを復号化するために必要なプライベートキーを有する受信者のみが表示できます。

Amazon SES では以下の MIME タイプがサポートされ、S/MIME で暗号化された E メールを送信するために使用できます。

- `application/pkcs7-mime`
- `application/pkcs7-signature`
- `application/x-pkcs7-mime`
- `application/x-pkcs7-signature`

Amazon SES では以下の MIME タイプ もサポートされており、PGP で暗号化された E メールを送信するために使用できます。

- `application/pgp-encrypted`
- `application/pgp-keys`
- `application/pgp-signature`

Amazon SES ヘッダーフィールド

Amazon SES は、RFC 822で規定された形式に準拠するあらゆる E メールヘッダーに対応できます。

以下のフィールドをメッセージのヘッダーセクションに複数回含めることはできません。

- `Accept-Language`
- `acceptLanguage`
- `Archived-At`
- `Auto-Submitted`
- `Bounces-to`
- `Comments`

- Content-Alternative
- Content-Base
- Content-Class
- Content-Description
- Content-Disposition
- Content-Duration
- Content-ID
- Content-Language
- Content-Length
- Content-Location
- Content-MD5
- Content-Transfer-Encoding
- Content-Type
- Date
- Delivered-To
- Disposition-Notification-Options
- Disposition-Notification-To
- DKIM-Signature
- DomainKey-Signature
- Errors-To
- From
- Importance
- In-Reply-To
- Keywords
- List-Archive
- List-Help
- List-Id
- List-Owner
- List-Post

- List-Subscribe
- List-Unsubscribe
- List-Unsubscribe-Post
- Message-Context
- Message-ID
- MIME-Version
- Organization
- Original-From
- Original-Message-ID
- Original-Recipient
- Original-Subject
- Precedence
- Priority
- References
- Reply-To
- Return-Path
- Return-Receipt-To
- Sender
- Solicitation
- Sensitivity
- Subject
- Thread-Index
- Thread-Topic
- User-Agent
- VBR-Info

考慮事項

- acceptLanguage フィールドは標準ではありません。可能であれば、代わりに Accept-Languageヘッダーを使用してください。

- Dateヘッダーを指定した場合、Amazon SES は、Amazon SES がメッセージを受け取ったときに、UTC タイムゾーンの日付と時刻に対応するタイムスタンプでヘッダーを上書きします。
- Message-IDヘッダーを指定すると、Amazon SES はヘッダーを独自の値で上書きします。
- Return-Pathヘッダーを指定した場合、Amazon SES は指定されたアドレスにバウンスと苦情の通知を送信します。ただし、受取人が受け取るメッセージのReturn-Pathヘッダーには別の値が含まれています。
- Simple または Templated のコンテンツで Amazon SES API v2 SendEmail オペレーションを使用する場合、または SendBulkEmail オペレーションを使用する場合、SES が設定したヘッダーのカスタムヘッダーコンテンツはユーザーは設定できません。したがって、次のヘッダーはカスタムヘッダーとして許可されません。
  - BCC, CC, Content-Disposition, Content-Type, Date, From, Message-ID, MIME-Version, Reply-To, Return-Path, Subject, To

Amazon SES でサポートされていない添付ファイルの種類

多目的インターネットメール拡張 (MIME) スタンダードを使用して、Amazon SES 経由で送信メッセージにファイルを添付できます。Amazon SES は、すべての添付ファイルのタイプを受け入れます。しかし次のリストのファイル拡張子を持つ添付ファイルは例外です。

.ade	.hta	.mau	.mst	.psc1
.adp	.inf	.mav	.ops	.psc2
.app	.ins	.maw	.pcd	.tmp
.asp	.isp	.mda	.pif	.url
.bas	.its	.mdb	.plg	.vb
.bat	.js	.mde	.prf	.vbe
.cer	.jse	.mdt	.prg	.vbs
.chm	.ksh	.mdw	.reg	.vps
.cmd	.lib	.mdz	.scf	.vsmacros
.com	.lnk	.msc	.scr	.vss

.cpl	.mad	.msh	.sct	.vst
.crt	.maf	.msh1	.shb	.vsw
.csh	.mag	.msh2	.shs	.vxd
.der	.mam	.mshxml	.sys	.ws
.exe	.maq	.msh1xml	.ps1	.wsc
.fxp	.mar	.msh2xml	.ps1xml	.wsf
.gadget	.mas	.msi	.ps2	.wsh
.hlp	.mat	.msp	.ps2xml	.xnk

ISP によっては他にも制限があります (添付ファイルのアーカイブに関する制限など)。実際に本稼働する前に大手 ISP を使って E メール送信をテストするようお勧めします。

Amazon SES を使用した E メールの受信

Amazon SES を使用して E メール送信を管理するだけでなく、1 つ以上のドメインに代わって E メールを受信するように SES を設定することもできます。E メール受信者としての SES は、基本的なメール受信操作を処理します。例えば、他のメールサーバーとの通信、スパムやウイルスのスキャン、信頼できないソース (Spamhaus または SES のブロックリストのアドレス) からのメールの拒否、ドメイン内の受取人宛のメールの承認などを行います。

受信メールの処理範囲は、指定したカスタム手順によって決まります。これらの手順には、次の 2 つの形式があります。

- 受信ルール（受信者ベースの制御）は、受信メールを細かく制御できます。受信ルールを使用すると、受信メールを Amazon S3 バケットへの配信、Amazon SNS トピックへの公開、Amazon WorkMail への送信、メッセージが特定の E アドレス宛ての場合、返送、メッセージの自動送信などの高度な処理を実行できます。
- IP アドレスフィルタ（IP ベースの制御）は、幅広いレベルの制御を提供し、設定が簡単です。これらのフィルタを使用すると、特定の IP アドレスまたは IP アドレス範囲からのすべてのメッセージを明示的にブロックまたは許可できます。

E メールの受信、設定、および受信ルールまたはIPアドレスフィルターを使用した実装について学習するには、まず、Eメール受信の概念とユースケースを理解して、その機能とさまざまな使用方法の概要を確認してください。次に、Eメール受信の前提条件の設定について記載したE メール受信のセットアップをご覧ください。次に、受信ルールおよびIP アドレスフィルタ設定で使用されるウィザードについて、E メール受信コンソールウォークスルーを参照してください。

> **Note**
>
> E メール受信は、SES が E メールの受信をサポートしている AWS リージョンにアカウントがある場合にのみ、使用できます。「AWS 全般のリファレンス」 の「E メール受信エンドポイント」表には、SES が E メール受信をサポートするすべての AWS リージョン が一覧表示されています。

このセクションのトピック:

- Amazon SES E メール受信のコンソールウォークスルー
- Amazon SES によるメール受信のメトリクスの表示

Amazon SESのEメール受信の概念とユースケース

メールの受信者としてAmazon SESを使用する場合は、メールの処理方法をサービスに指示します。主な方法である受信ルールを使用すると、受信者ベースの制御を使用して、受信者に基づいて実行する一連のアクションを指定し、Eメール受信を細かく制御することができます。もう1つの方法であるIPアドレスフィルターは、発信元のIPアドレスまたはアドレスの範囲に基づいてメールをブロックまたは許可するための幅広いレベルのIPベースの制御を提供します。

このセクションでは、これらの両方の方法と、Amazon SES が受信した E メールを処理する方法の概要と、ルールと加工を設定するときに E メールを受信し、加工し、処理する方法を検討するのに役立つユースケースについて説明します。

このセクションのトピック:

- 受信ルールを使用した受信者ベースの制御
- IP アドレスフィルタを使用した IP ベースの制御
- E メール受信プロセス
- Amazon SES による E メール受信に関するユースケースと制限
- E メール受信認証とマルウェアスキャン

受信ルールを使用した受信者ベースの制御

受信メールを制御する主な方法は、自分が所有している検証済み ID (ドメイン、サブドメイン、または E メールアドレス) のアクションの順序付きリストを使用して、メールの処理方法を指定することです。E メールアドレスは、自分が所有している検証済みドメイン ID のいずれかに属している必要があります。これらのアクションはルールセット 内で作成する受信ルール で定義および指定されます。

オプションとして、受信メールの宛先となる受信者が条件で指定された受信者IDと一致する場合にのみアクションが実行されるように指定する方法として、受信者条件を追加することもできます。たとえば、example.com というドメインを所有している場合、user@example.com 宛てのメールはバウンスさせ、example.com およびそのサブドメイン宛てのそれ以外のすべてのメールを配信するように指定できます。

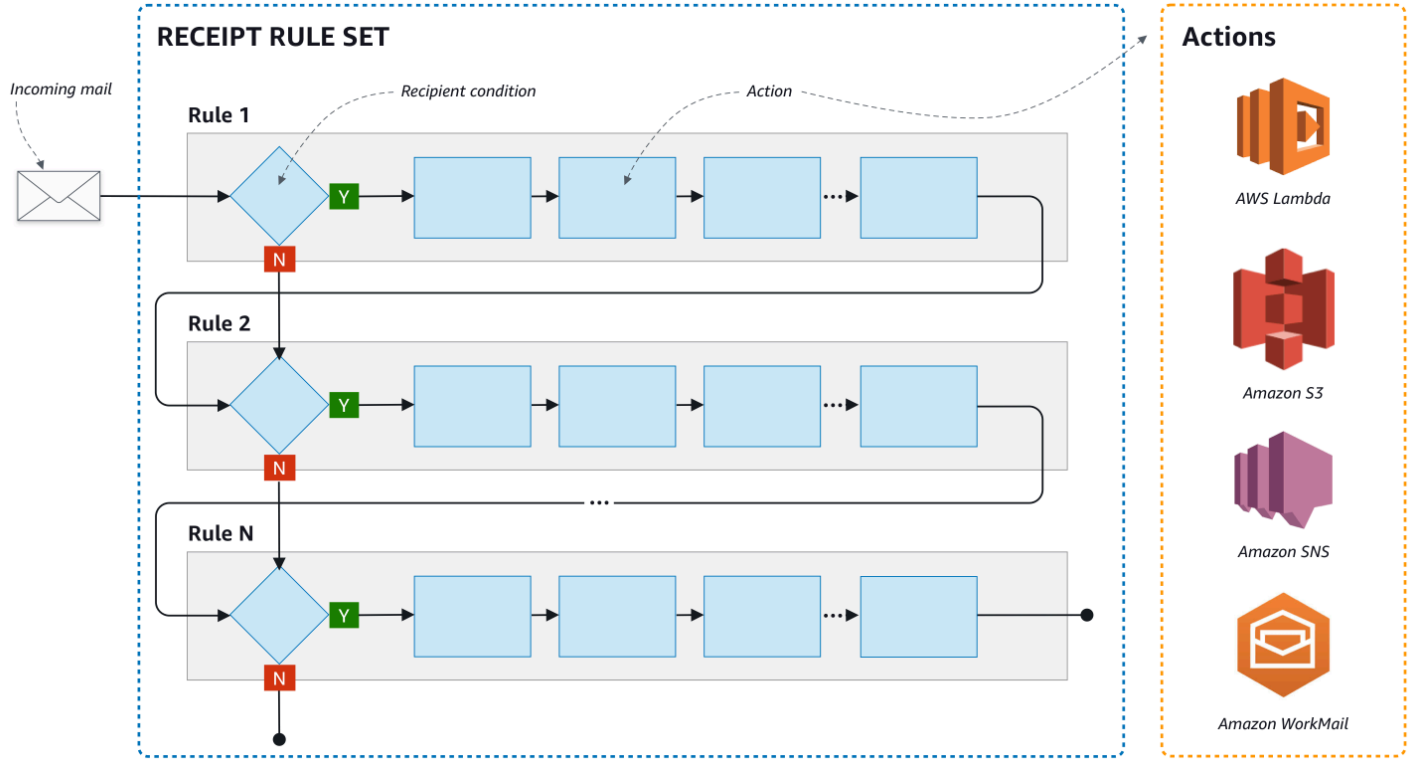
受信者の条件を追加しなければ、検証済みドメインに属するすべてのメールアドレス、ドメイン、およびサブドメインにアクションが適用されます。次のアクションは受信ルールに適用できます。

- ヘッダー追加アクション - 受信 E メールにヘッダーを追加します。このアクションは通常、他のアクションとの組み合わせでのみ使用します。
- バウンス応答の返信アクション - 送信者にバウンス応答を返すことによって E メールをブロックし、オプションで Amazon SNS を通じて通知します。
- AWS Lambda 関数の呼び出しアクション - Lambda 関数を通じてコードを呼び出し、オプションで Amazon SNS を通じて通知します。
- S3バケットへの配信アクション - メールを Amazon S3 バケットに配信し、オプションで Amazon SNS を通じて通知します。
- Amazon SNS トピックへの発行アクション - E メール全体を Amazon SNS トピックに発行します。

> **Note**
>
> この SNS アクションでは、メールのコンテンツの完全なコピーが Amazon SNS 通知に含まれます。ここで言及されるそれ以外の Amazon SNS 通知オプションは、単純にメールの配信を通知するだけであり、メールのコンテンツそのものではなくメールに関する情報が含まれます。

- ルールセットの停止アクション - 受信ルールセットの評価を終了し、オプションで、Amazon SNS を通じて通知します。
- Amazon WorkMail との統合アクション - メールを Amazon WorkMail で処理します。Amazon WorkMail がセットアップを処理するため、このアクションを直接使用することは通常はありません。

複数の受信ルールがグループ化されたものがルールセット です。既存のルールセットがない場合は、受信ルールの作成を開始する前にルールセットを作成する必要があります。ご使用のAWS アカウントで複数の受信ルールを定義することはできますが、有効にできる受信ルールセットは常に 1 つだけです。次の図は、受信ルール、受信ルールセット、アクションが相互にどのように関連するのかを示しています。



IP アドレスフィルタを使用した IP ベースの制御

IP アドレスフィルタ を設定することで、メールの流れを制御できます。IP アドレスフィルタはオプションであり、特定の IP アドレスまたは IP アドレス範囲から送信されるメールを受け入れるかブロックするかを指定できます。IP アドレスフィルタには、ブロックリスト (そこから受信するメールをブロックする IP アドレス) と 許可リスト (常にメールを受け入れる IP アドレス) を含めることができます。

IP アドレスフィルタは、スパムをブロックするのに便利です。Amazon SES は、Spamhaus に記載されているものを含め、スパムの送信元として知られる IP アドレスのブロックリストを独自に保持しています。ただし、IP アドレスを許可リストに追加することで、該当する IP アドレスからのメールを受信するよう選択できます。どの IP アドレスをブロックしているかはログに示されないため、該当する送信者はブロックされていることを送信先に通知する必要があります。これは、送信者の IP アドレスがブロックリスト (Spamhaus など) に含まれているかどうかを確認し、リストからの除外をリクエストするよう送信者に勧める良い機会ともなります。これにより、受信者は IP アドレスのフィルタを維持する必要がなくなり、送信者は E メールの配信率を改善できるという点で、受信者と送信者の両方にとって有益です。

> **Note**
>
> - IP アドレスフィルターの設定に関係なく、Amazon EC2 は許可リストに登録されていない限り、ポート 25 でのアウトバウンドトラフィック (メール送信) をブロックします。詳細については、こちらの AWSre:POST の記事を参照してください。
> - 既知の IP アドレスで構成される有限のリストからのメールのみを受信する場合は、`0.0.0.0/0` を含むブロックリストをセットアップし、信頼する IP アドレスを含む許可リストをセットアップします。この設定では、デフォルトではすべての IP アドレスがブロックされ、明示的に指定する IP アドレスからのメールのみが許可されます。

E メール受信プロセス

ご使用のドメイン宛ての E メールを Amazon SES が受信するときは、以下のイベントが発生します。

1. Amazon SES は最初に送信者の IP アドレスを検索します。Amazon SES は、次の場合を除き、メールがこの段階を通過することを許可します。
   - IP アドレスがブロックリストに含まれている。
   - IP アドレスが Amazon SES のブロックリストに含まれており、指定した許可リストには含まれていません。
2. Amazon SES は、設定された有効な受信ルールセットを調査して、受信者の条件と一致する条件が受信ルールに含まれているかどうかを判定します。
   - 受信者の条件があり、受信した E メールの受信者のいずれかに一致する場合、Amazon SES は E メールを受け入れます。一致がない場合、Amazon SES は当該メールをブロックします。
   - 受信ルールに受信者の条件が含まれていない場合、Amazon SES はメールを受け入れます。ルールのすべてのアクションは、お客様が所有している検証済みのすべての ID に適用されます。
3. Amazon SES は E メールを認証し、スパムとマルウェアについてコンテンツをスキャンします。
   - Amazon SES に E メールを配信したリモートホストの IP アドレスは、SMTP トランザクションで使用される MAIL FROM のドメインで指定された SPF ポリシーと照合されます。
   - メールのヘッダーセクションにある DKIM 署名がチェックされます。
   - コンテンツスキャンが有効になっている場合、スパムとマルウェアについて E メールコンテンツをスキャンします。

- E メール認証とコンテンツスキャンの結果は、受信ルールの評価中に利用可能になります。

  詳細については、「メール認証とマルウェア検出」を参照してください。

4. Amazon SES が受け入れる E メールの場合、有効なルールセット内のすべての受信ルールが、定義した順序で適用されます。各受信ルール内では、アクションはユーザーが定義した順序で実行されます。

Amazon SES による E メール受信に関するユースケースと制限

Amazon SES による E メール受信に関して、一般的な考慮事項とユースケースをいくつか説明します。質問と回答形式で提示され、Amazon SES を使用して、所有している 1 つ以上の検証済みドメインに代わって E メールを受信および管理することが有益かどうかを判断するのに役立つ、よくある質問と事実です。

リージョナルな可用性

ご利用のリージョンで Amazon SES は E メール受信をサポートしていますか。

Amazon SES は、特定の AWS リージョンでのみ E メール受信をサポートしています。E メールの受信がサポートされるリージョンの完全なリストについては、「AWS 全般のリファレンス」の「Amazon Simple Email Service のエンドポイントとクォータ」を参照してください。

POP または IMAP ベースのEメールクライアント

受信メールの受信に Microsoft Outlook を使用することはできますか？

Amazon SES には、E メールを受信するための POP サーバーや IMAP サーバーは含まれていません。つまり、Microsoft Outlook などの E メールクライアントを使用して E メールを受信することはできません。E メールクライアントを使用して送信と受信の両方が可能なソリューションが必要な場合は、Amazon WorkMail の使用を検討してください。

他の AWS サービスの使用

適切なアクセス権限をセットアップしましたか。

メールをS3 バケットに配信する場合、自身が所有していない Amazon SNS トピックに公開する場合、 Lambda 関数をトリガーする場合、または顧客が管理するキーを使用する場合は、それらのリソースへのアクセス許可を Amazon SES に付与する必要があります。Amazon SES にアクセス権限

を付与するには、それらの AWS のサービスに対応するコンソールまたは API から、リソースに対するポリシーを作成します。詳細については、「アクセス許可の付与」を参照してください。

E メールのコンテンツ

どのような方法で Amazon SES から E メールのコンテンツを受信しますか。

Amazon SES では、E メールのコンテンツを 2 通りの方法で提供できます。指定した S3 バケットに E メールを保存する方法、または E メールのコピーを含む Amazon SNS 通知を送信する方法です。Amazon SES では、変更を加えていない E メールを多目的インターネットメール拡張 (MIME) 形式で配信します。MIME 形式の詳細については、RFC 2045 を参照してください。

受信するメールの大きさはどれくらいですか。

S3 バケットに E メールを保存する場合、E メールの最大サイズ (ヘッダーを含む) は 40 MB です。Amazon SNS を通じて E メールを受信する場合、E メールの最大サイズ (ヘッダーを含む) は 150 KB です。

E メールの処理をどのようにトリガーしますか。

メールが配信された後で、独自のコードで処理する場合があります。たとえば、アプリケーションを使用して Base64 エンコードの E メールを表示可能な形式に変換してから、エンドユーザーが E メールクライアントで使用できるようにする場合が考えられます。このプロセスを開始する方法はいくつかあります。

- E メールが Amazon S3 に配信される場合、アプリケーションは、S3 アクションによって生成される Amazon SNS 通知をリッスンし、通知から E メールのメッセージ ID を抽出して、そのメッセージ ID を使用して Amazon S3 から E メールを取得します。

  または、Lambda 関数を記述することで、受信ルールに E メール処理を組み込むこともできます。この場合、受信ルールでは、まず E メールを Amazon S3 に書き込んでから Lambda 関数をトリガーする必要があります。Lambda 関数が他のアクションの実行方法に影響する結果を返す必要があるかどうかに応じて、受信ルールから複数の Lambda アクションを同期または非同期で実行することができます。ユースケースで同期実行が絶対に必要とされるのでない限りは、非同期実行を使用することをお勧めします。AWS Lambda の詳細については、AWS Lambda 開発者ガイドを参照してください。
- SNS アクションを使用して E メールを Amazon SNS 通知を介して配信する場合は、アプリケーションは、Amazon SNS 通知をリッスンし、通知から E メールメッセージを抽出することができます。

E メールを暗号化しますか。

Amazon SES はAWS Key Management Service (AWS KMS) と統合され、オプションでS3 バケットに書き込むメールを暗号化します。Amazon SES は、クライアント側の暗号化を使用して、Amazon S3 に書き込む前にメールを暗号化します。つまり、Amazon S3 からメールを取得した後にご自身でコンテンツを復号する必要があります。- AWS SDK for Java および AWS SDK for Ruby は、復号化を処理できるクライアントを提供します。Amazon SES では、E メールをS3 バケットに配信するよう選択した場合にのみEメールを暗号化できます。

不要なメール

メール受信プロセスのどの時点で、不要なメールをブロックしますか。

送信者が受信者に E メールを送信しようとすると、送信者の E メールサーバーは一連のコマンドを受信者のサーバーと交換します。 このシーケンスは SMTP 対話と呼ばれます。

E メール受信プロセスの 2 つの時点 (SMTP 対話中と SMTP 対話後) で、受信 E メールをブロックできます。SMTP 対話中にメッセージをブロックするには IP アドレスフィルターを使用し、SMTP 対話後に E メールをブロックするには受信ルールを使用します。

特定の IP アドレスから送信される E メールをブロックするには、IP アドレスフィルターを使用できます。IP アドレスフィルターを使用して不要メールをブロックする利点は、SMTP 対話中にブロックされたメッセージに対しては料金が発生しないことです。 IP アドレスフィルターを使用することの欠点は、実際のメッセージの内容を分析することなく、指定した IP アドレスからの E メールをブロックしてしまうことです。IP アドレスフィルターの詳細については、「IP アドレスフィルタのコンソールウォークスルーの作成」を参照してください。

受信ルールを使用して、メッセージの送信先アドレス (ドメイン、またはサブドメイン) に基づいて、バウンス通知を E メールの送信者に送信できます。 受信ルールを使用する利点は、送信者にバウンス通知を送信する前に、受信メッセージに対して追加の分析を実行できることです。たとえば、メッセージが DKIM 認証に失敗した場合やスパムとして識別された場合にのみバウンス通知を送信するために AWS Lambda を使用できます。 受信ルールを使用することの欠点は、受信ルールが SMTP 対話の後に処理されるため、受信したメッセージごとに請求されることです。 Lambda を使用して着信メッセージの内容を分析した場合も、料金が発生する可能性があります。 受信ルールの詳細については、「受信ルールのコンソールウォークスルーの作成」を参照してください。Lambda を使用した受信 E メール分析の詳細については、「Lambda 関数の例」を参照してください。

メールのストリーム

メールのストリームをどのように分割しますか。

ご使用のドメインでは、さまざまなクラスのメールを受信する可能性があります。たとえば、user@example.com 宛てのメールのように、ドメインのメールの一部は個人の受信箱に配信されます。それ以外に、unsubscribe@example.com 宛てのメールのように、自動化されたシステムに送信されるべきものもあります。それぞれ異なる処理を行えるように、受信ルールを使用して受信メールを分割することができます。受信ルールをセットアップする方法については、「受信ルールの作成」を参照してください。

E メール受信認証とマルウェアスキャン

Amazon SES は受信した各 E メールを認証し、オプションでスパムやマルウェアについて E メールのコンテンツをスキャンします。SES は、E メール認証またはコンテンツスキャンの結果に基づいて、受信した E メールに対してアクションを実行しません。しかし、これらの操作の結果は、Amazon SNS の通知などの SES 受信ルールアクションで使用できる属性として、または Amazon S3 に配信されるメッセージのヘッダーとして、お客様に提供されます。

E メールの認証

Amazon SES は、SPF、DKIM、DMARC を使用して受信した各 E メールを認証します。各認証メカニズムの結果は、アクティブな受信ルールセットのルールの評価の一環として SES がディスパッチする Amazon SNS 通知に表示されます。さらに、Amazon S3 でメールのコピーを受信することを選択した場合、E メール認証の結果は SES が E メールのヘッダーセクションに追加する `Authentication-Results` ヘッダーでキャプチャされます :

```
Authentication-Results: example.com;
spf=pass (spfCheck: 10.0.0.1 is permitted by domain of example.com) client-ip=10.0.0.1;
 envelope-from=example@example.com; helo=10.0.0.1;
dkim=pass header.i=example.com;
dkim=permerror header.i=some-example.com;
dmarc=pass header.from=example@example.com;
```

`Authentication-Results` ヘッダーについては、「RFC 8601」を参照してください。

スパムおよびマルウェア検出のための E メールコンテンツのスキャン

Amazon SES は、E メールに一致する受信ルールのスキャン有効 (API) やスパムとウイルススキャン (コンソール) 属性の値に応じて、受信した E メールコンテンツをマルウェアについてスキャンします。デフォルトでは、SES は受信した E メールコンテンツをマルウェアについてスキャンします。特定の受信ルールに一致する受信メールのコンテンツスキャンを無効にするには、API を使用する場合は、受信ルールの [ScanEnabled] (スキャン有効) フラグを false に設定し、コンソールを

使用する場合は、[Spam and virus scanning] (スパムとウイルススキャン) チェックボックスをオフにします。E メールに一致する受信ルールでスキャンが有効になっている場合、コンテンツスキャンの結果は、アクティブな受信ルールセットのルール評価の一環として SES がディスパッチする Amazon SNS 通知に表示されます。さらに、Amazon S3 でメールのコピーを受信することを選択した場合、コンテンツスキャンの結果はSES がメールのヘッダーセクションに追加するヘッダー `X-SES-Spam-Verdict` および `X-SES-Virus-Verdict` にキャプチャされます。

```
X-SES-Spam-Verdict: PASS
X-SES-Virus-Verdict: FAIL
```

上記のヘッダーに使用できる値は、次のとおりです :

- スパム
- ウイルス

これでE メール受信の概念、そのしくみ、ユースケースを理解されたので、E メール受信のセットアップ から始めてみましょう。

Amazon SES E メール受信の設定

このセクションでは、メールを受信できるように Amazon SES を設定する前に、行うべき前提条件について説明します。Eメール受信の概念とユースケース Amazon SESの動作方法の概念を理解し、E メール受信、フィルタリング、処理をどのように実行するかについての概念を理解するを読むことが重要です。

ルールセット、受信ルール、およびIP アドレスフィルタリングを作成し、Eメールの受信を設定する前に、まず次の設定の前提条件を完了する必要があります。

- DNS レコードを発行して Amazon SES でドメインを検証し、それを所有していることを証明します。
- MX レコードを発行して Amazon SES でドメインの E メールを受信できるようにします。
- Amazon SES に、他の AWS リソースへのアクセスを付与して、受信ルールのアクションを実行します。

ドメイン ID を作成して検証する場合、DNS 設定にレコードを発行して検証プロセスを完了しますが、これだけではEメールの受信を使用できません。メール受信に特有で、カスタムメール送信元ドメインを指定するために MX レコードを発行する必要もあります。このレコードは、SES がドメイ

ンのEメールを受信できるようにするために、ドメインの DNS 設定で使用されます。受信ルールで選択したアクションは、Amazon SES でこれらのアクションに求められるそれぞれの AWS サービスを使用する許可がない限り動作しないため、許可が必要です。

Eメール受信を使用するために必要なこれら3つの前提条件については、これ以降のトピックで説明します。

- Amazon SES による E メール受信のためのドメインの検証
- Amazon E SESメール受信用の MX レコードの発行
- E メール受信SESのために Amazon にアクセス許可を付与する

Amazon SES による E メール受信のためのドメインの検証

Amazon SES での E メールの送受信に使用するドメインに関しては、まず、ご自身がそのドメインを所有していることを証明する必要があります。検証の手順には、SES でのドメイン検証の開始と、DNS レコード (ご使用の検証方法によって CNAME または TXT のどちらか) を DNS プロバイダーに発行することが含まれます。

コンソールを使用して、ドメインを Easy DKIM (簡易 DKIM) または Bring Your Own DKIM (BYODKIM) (自分の DKIM を使用する) のいずれかで確認し、簡単に DNS レコードをコピーして DNS プロバイダーに発行できます。これを行う方法については、「ドメイン ID の作成」で説明しています。必要に応じて、SES の `VerifyDomainDkim` または `VerifyDomainIdentity` API を使用できます。

ドメインまたは E メールアドレスが検証されていることは、SES コンソールの [Verified identities] (検証済み ID) テーブルでステータスを確認するか、SES の `GetIdentityVerificationAttributes` または `GetEmailIdentity` API を使用することで簡単に確認できます。

Amazon E SESメール受信用の MX レコードの発行

メールエクスチェンジャレコード (MX レコード#) は、ドメインに送信された E メールを受け入れることができるメールサーバーを指定する設定です。

Amazon が受信 E メールSESを管理するには、ドメインDNSの設定に MX レコードを追加する必要があります。作成する MX レコードは、Amazon を使用する AWS リージョンの E メールを受信するエンドポイントを指しますSES。例えば、米国西部 (オレゴン) リージョンのエンドポイントは inbound-smtp.us-west-2.amazonaws.com です。エンドポイントの詳細なリストについては、「SES のリージョンとエンドポイント」を参照してください。

> **Note**
>
> Amazon で E メールを受信するエンドポイントSESは、 IMAPまたは E POP3メールサーバーではありません。E メールクライアントの受信メールサーバーURLsとして使用することはできません。E メールクライアントを使用して E メールを送受信できるソリューションが必要な場合は、Amazon WorkMail の使用を検討してください。

次の手順には、MX レコードを作成するための一般的なステップが含まれています。MX レコードを作成する具体的な手順は、 DNSまたはホスティングプロバイダーによって異なります。ドメインDNSの設定に MX レコードを追加する方法については、プロバイダーのドキュメントを参照してください。

> **Note**

MX レコードをドメインDNSの設定に追加するには

1. (前提条件) これらの手順を完了するには、ドメインのDNSレコードを変更する必要があります。DNS レコードにアクセスできない場合、またはアクセスしにくい場合は、システム管理者にお問い合わせください。
2. DNS プロバイダーの マネジメントコンソールにサインインします。
3. 新しい MX レコードを作成します。
4. MX レコードの [Name] (名前) にドメインを入力します。例えば、ドメイン example.com に送信される E メールを Amazon でSES管理する場合は、次のように入力します。

   ```
   example.com.
   ```

   > **Note**
   >
   > DNS プロバイダーに応じて: 1) ドメイン拡張の . 末尾の は必須ではない場合があります。2) Name フィールドは、ホスト、ドメイン、またはメールドメインと呼ばれる場合があります。

5. [Type (タイプ)] で [MX] を選択します。

> **Note**
>
> 一部のDNSプロバイダーは、タイプフィールドをレコードタイプまたは同様の名前と呼んでいます。

6. [値] に以下の値を入力します。

```
10 inbound-smtp.region.amazonaws.com
```

前の例では、 を、Amazon で使用する AWS リージョンの E メールを受信するエンドポイントの*region*アドレスに置き換えますSES。たとえば、米国東部 (バージニア北部) リージョンを使用している場合は、 *region*を に置き換えますus-east-1。メール受信用のエンドポイントの完全なリストについては、「SES のリージョンとエンドポイント」を参照してください。

> **Note**
>
> 一部のDNSプロバイダーのマネジメントコンソールには、レコード Value とレコード Priority の個別のフィールドが含まれています。DNS プロバイダーに当てはまる場合は、 Priority 値10に を入力し、URLValue に受信メールエンドポイントを入力します。

> **Important**
>
> MX レコードを作成する具体的な手順は、 DNSまたはホスティングプロバイダーによって異なります。ドメインDNSの設定に MX レコードを追加する方法については、プロバイダーのドキュメントを参照するか、プロバイダーにお問い合わせください。

さまざまなプロバイダーの MX レコードを作成するための手順

ドメインの MX レコードを作成する手順は、使用するDNSプロバイダーによって異なります。このセクションには、いくつかの一般的なDNSプロバイダーのドキュメントへのリンクが含まれています。これは、プロバイダーの完全なリストではありません。プロバイダーが以下にリストされていない場合でも、おそらく Amazon で使用できますSES。このリストに含まれているからといって、他社の製品やサービスを支持または推奨するものではありません。

DNS/ホスティングプロバイダー名	ドキュメントのリンク
Amazon Route 53	Amazon Route 53 コンソールを使用して、レコードを作成する
GoDaddy	MX レコードを追加する (外部リンク)
DreamHost	MX レコードを変更するにはどうすればよいですか? (外部リンク)
Cloudflare	E メールレコードのセットアップ (外部リンク)
HostGator	MX レコードの変更 - Windows (外部リンク)
Namecheap	メールサービスに必要な MX レコードを設定するには、どうすればよいですか? (外部リンク)
Names.co.uk	ドメインDNSの設定の変更 (外部リンク)
Wix	Wix アカウントの MX レコードの追加または更新 (外部リンク)

E メール受信SESのために Amazon にアクセス許可を付与する

Amazon Simple Storage Service (Amazon S3) バケットへの E メールの送信や AWS Lambda 関数の呼び出しなどSES、 で E メールを受信したときに実行できるタスクには、特別なアクセス許可が必要です。このセクションには、いくつかの一般的なユースケースのサンプルポリシーを含みます。

このセクションのトピック:

- S3 バケットへの配信アクションのIAMロールアクセス許可の設定
- S3 バケットに書き込むアクセスSES許可を付与する
- AWS KMS キーを使用するアクセスSES許可を付与する
- AWS Lambda 関数を呼び出すアクセスSES許可を付与する
- 別の AWS アカウントに属する Amazon SNSトピックに発行するSESアクセス許可を付与する

S3 バケットへの配信アクションのIAMロールアクセス許可の設定

このIAMロールには、次の点が適用されます。

- これは、S3 バケットアクションへの配信 用にのみ使用可能です。
- SES the section called “E メールの受信” が利用できないリージョンに存在する S3 バケットに書き込む場合は、このバケットを使用する必要があります。

S3 バケットに書き込む場合は、 の関連リソースにアクセスするためのアクセス許可を IAMロールに付与できますS3 バケットアクションへの配信。また、次のセクションで説明するように、IAM信頼ポリシーを使用してアクションを実行するために、そのロールを引き受けるSESアクセス許可を付与する必要があります。

このアクセス許可ポリシーは、IAMロールのインラインポリシーエディタに貼り付ける必要があります。「」を参照S3 バケットアクションへの配信し、IAMロール項目に記載されているステップに従ってください。(次の例には、SNSトピック通知を使用する場合のオプションのアクセス許可、または S3 アクションのカスタマーマネージドキーも含まれます)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        // Required: allows SES to write in the bucket
        {
            "Sid": "S3Access",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::my-bucket/*"
        },
        // Optional: use if an SNS topic is used in the S3 action
        {
            "Sid": "SNSAccess",
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:region:111122223333:my-topic"
        },
        // Optional: use if a customer managed key is used in the S3 action
        {
            "Sid": "KMSAccess",
            "Effect": "Allow",
            "Action": "kms:GenerateDataKey*",
            "Resource": "arn:aws:kms:region::111122223333:key/key-id"
```

```
        }
    ]
}
```

上のポリシー例に、以下の変更を加えます。

- を、書き込み先の S3 バケットの名前*my-bucket*に置き換えます。
- を、受信ルールを作成した AWS リージョン *region*に置き換えます。
- *111122223333* を AWS アカウント ID に置き換えます。
- を、通知を発行するSNSトピックの名前*my-topic*に置き換えます。
- をKMSキーの ID *key-id*に置き換えます。

S3 アクションIAMロールの信頼ポリシー

次の信頼ポリシーをIAMロールの信頼関係に追加して、 がそのロールを引き受けSESることを許可する必要があります。

> **Note**
>
> この信頼ポリシーを手動で追加する必要があるのは、S3 バケットアクションへの配信ワークフローのIAMロール項目で指定されたステップを使用してSESコンソールからIAMロールを作成しなかった場合のみです。コンソールからIAMロールを作成すると、この信頼ポリシーが自動的に生成され、ロールに適用され、このステップが不要になります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSESAssume",
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                  "AWS:SourceAccount":"111122223333",
```

```
                    "AWS:SourceArn": "arn:aws:ses:region:111122223333:receipt-rule-
set/rule_set_name:receipt-rule/receipt_rule_name"
                }
            }
        }
    ]
}
```

上のポリシー例に、以下の変更を加えます。

- を、受信ルールを作成した AWS リージョン *region*に置き換えます。
- *111122223333* を AWS アカウント ID に置き換えます。
- を、Amazon S3 バケットへの配信アクションを含む受信ルールを含むルールセットの名前*rule_set_name*に置き換えます。
- を、Amazon S3 バケットへの配信アクションを含む受信ルールの名前*receipt_rule_name*に置き換えます。

S3 バケットに書き込むアクセスSES許可を付与する

次のポリシーを S3 バケットに適用すると、E SESメール受信が利用可能なリージョンに存在する限り、そのバケットに書き込むSESアクセス許可が に付与されます。E メール受信リージョン外のバケットに書き込む場合は、「」を参照してくださいS3 バケットへの配信アクションのIAMロールアクセス許可の設定。入力メールを Amazon S3 に転送する受信ルールを作成することに関する詳細については、「S3 バケットアクションへの配信」を参照してください。

S3 バケットの添付ポリシーの詳細については、Amazon Simple Storage Service ユーザーガイド の「バケットポリシーとユーザーポリシーの使用 」を参照してください。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AllowSESPuts",
      "Effect":"Allow",
      "Principal":{
        "Service":"ses.amazonaws.com"
      },
      "Action":"s3:PutObject",
      "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*",
      "Condition":{
```

```
        "StringEquals":{
          "AWS:SourceAccount":"111122223333",
          "AWS:SourceArn": "arn:aws:ses:region:111122223333:receipt-rule-
set/rule_set_name:receipt-rule/receipt_rule_name"
        }
      }
    }
  ]
}
```

上のポリシー例に、以下の変更を加えます。

- を、書き込み先の S3 バケットの名前*amzn-s3-demo-bucket*に置き換えます。
- を、受信ルールを作成した AWS リージョン*region*に置き換えます。
- *111122223333* を AWS アカウント ID に置き換えます。
- を、Amazon S3 バケットへの配信アクションを含む受信ルールを含むルールセットの名前*rule_set_name*に置き換えます。
- を、Amazon S3 バケットへの配信アクションを含む受信ルールの名前*receipt_rule_name*に置き換えます。

AWS KMS キーを使用するアクセスSES許可を付与する

SES が E メールを暗号化するには、受信ルールの設定時に指定した AWS KMS キーを使用するためのアクセス許可が必要です。アカウントでデフォルトKMSキー (aws/ses) を使用するか、作成したカスタマーマネージドキーを使用できます。デフォルトKMSキーを使用する場合は、追加のステップを実行して、そのキーを使用するためのSESアクセス許可を付与する必要はありません。カスタマーマネージドキーを使用する場合は、キーのポリシーにステートメントを追加して、そのキーを使用するためのSESアクセス許可を付与する必要があります。

次のポリシーステートメントをキーポリシーとして使用して、 がドメインで E メールを受信したときにカスタマーマネージドキーSESを使用できるようにします。

```
{
  "Sid": "AllowSESToEncryptMessagesBelongingToThisAccount",
  "Effect": "Allow",
  "Principal": {
    "Service":"ses.amazonaws.com"
  },
  "Action": [
```

```
    "kms:GenerateDataKey*"
  ],
  "Resource": "*",
  "Condition":{
        "StringEquals":{
          "AWS:SourceAccount":"111122223333",
          "AWS:SourceArn": "arn:aws:ses:region:111122223333:receipt-rule-
set/rule_set_name:receipt-rule/receipt_rule_name"
        }
      }
}
```

上のポリシー例に、以下の変更を加えます。

- を、受信ルールを作成した AWS リージョン*region*に置き換えます。
- *111122223333* を AWS アカウント ID に置き換えます。
- を、E メール受信に関連付けられた受信ルールを含むルールセットの名前*rule_set_name*に置き換えます。
- を、E メール受信に関連付けられた受信ルールの名前*receipt_rule_name*に置き換えます。

を使用してサーバー側の暗号化が有効になっている S3 バケット AWS KMS に暗号化されたメッセージを送信する場合は、ポリシーアクション を追加する必要があります"kms:Decrypt"。前の例を使用してこのアクションをポリシーに追加すると、次のように表示されます。

```
{
  "Sid": "AllowSESToEncryptMessagesBelongingToThisAccount",
  "Effect": "Allow",
  "Principal": {
    "Service":"ses.amazonaws.com"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey*"
  ],
  "Resource": "*",
  "Condition":{
        "StringEquals":{
          "AWS:SourceAccount":"111122223333",
          "AWS:SourceArn": "arn:aws:ses:region:111122223333:receipt-rule-
set/rule_set_name:receipt-rule/receipt_rule_name"
        }
```

```
      }
}
```

AWS KMS キーへのポリシーのアタッチの詳細については、 AWS Key Management Service デベロッパーガイドの「 でのキーポリシーの使用 AWS KMS」を参照してください。

AWS Lambda 関数を呼び出すアクセスSES許可を付与する

SES で AWS Lambda 関数を呼び出すには、SESコンソールで受信ルールを作成するときに関数を選択できます。これを行うと、 は必要なアクセス許可を関数SESに自動的に追加します。

または、 の AddPermission AWS Lambda オペレーションを使用して、 関数APIにポリシーをアタッチすることもできます。への次の呼び出しAddPermissionAPIは、Lambda 関数を呼び出すアクセスSES許可を付与します。Lambda 関数へのポリシーの添付に関する詳細については、AWS Lambda デベロッパーガイドの「AWS Lambda のアクセス許可」を参照ください。

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "ses.amazonaws.com",
  "SourceAccount": "111122223333",
  "SourceArn": "arn:aws:ses:region:111122223333:receipt-rule-set/rule_set_name:receipt-
rule/receipt_rule_name"
  "StatementId": "GiveSESPermissionToInvokeFunction"
}
```

上のポリシー例に、以下の変更を加えます。

- を、受信ルールを作成した AWS リージョン*region*に置き換えます。
- *111122223333* を AWS アカウント ID に置き換えます。
- を、Lambda 関数を作成した受信ルールを含むルールセットの名前*rule_set_name*に置き換えます。
- を Lambda 関数を含む受信ルールの名前*receipt_rule_name*に置き換えます。

別の AWS アカウントに属する Amazon SNSトピックに発行するSESアクセス許可を付与する

別の AWS アカウントのトピックに通知を発行するには、Amazon SNSトピックにポリシーをアタッチする必要があります。SNS トピックは、ドメインおよび受信ルールセットと同じリージョンに存在する必要があります。

次のポリシーは、別の AWS アカウントの Amazon SNSトピックに発行するSESアクセス許可を付与します。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "Service":"ses.amazonaws.com"
      },
      "Action":"SNS:Publish",
      "Resource":"arn:aws:sns:topic_region:sns_topic_account_id:topic_name",
      "Condition":{
        "StringEquals":{
          "AWS:SourceAccount":"aws_account_id",
          "AWS:SourceArn": "arn:aws:ses:receipt_region:aws_account_id:receipt-rule-
set/rule_set_name:receipt-rule/receipt_rule_name"
        }
      }
    }
  ]
}
```

上のポリシー例に、以下の変更を加えます。

- を、Amazon SNSトピック AWS リージョン が作成された *topic_region*に置き換えます。
- を、Amazon SNSトピックを所有する AWS アカウントの ID *sns_topic_account_id*に置き換えます。
- を、通知を発行する Amazon SNSトピックの名前*topic_name*に置き換えます。
- を、E メールを受信するように設定されたアカウントの ID *aws_account_id*に置き換えます AWS 。
- を、受信ルールを作成した AWS リージョン *receipt_region*に置き換えます。
- を、Amazon SNSトピックへの発行アクションを作成した受信ルールを含むルールセットの名前*rule_set_name*に置き換えます。
- を Amazon SNSトピックへの発行アクションを含む受信ルールの名前*receipt_rule_name*に置き換えます。

Amazon SNSトピックでサーバー側の暗号化 AWS KMS に を使用する場合は、 AWS KMS キーポリシーにアクセス許可を追加する必要があります。アクセス許可を追加するには、次のポリシーを AWS KMS キーポリシーにアタッチします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSESToUseKMSKey",
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon SES E メール受信のコンソールウォークスルー

このセクションでは、Eメール受信コンソールウィザードについて説明します。このウィザードでは、受信ルール およびIPアドレスフィルタ を設定して、メールの受信を管理できます。コンソールウィザードを使用する前に、Eメール受信の概念とユースケース および E メール受信のセットアップ を読み、Eメール受信の仕組みの概念とセットアップの前提条件を満たしていることを確認してください。

受信ルールと IP アドレスフィルタを設定するためのコンソールウィザードについては、以下で説明します。

- 受信ルールのコンソールウォークスルーの作成
- IP アドレスフィルタのコンソールウォークスルーの作成

受信ルールのコンソールウォークスルーの作成

このセクションでは、Amazon SESコンソールを使用して受信ルールを作成および定義する方法について説明します。受信ルールがどのように機能するのか理解するための重要なポイントは、次のとおりです。

- ルールセット には、受信ルールの順序セットが含まれており、受信ルール には、順序付けられた一連のアクションが含まれます。
- 受信ルールは、指定したアクションの順序付きリストを実行して受信メールを処理するSES方法をAmazon に指示します。
- この順序付けられたアクションのリストは、オプションで受信者条件の初回マッチングに応じて作成できます。指定しない場合、アクションは検証済みドメインに属するすべての ID に適用されます。
- 受信ルールは、ルールセットと呼ばれるコンテナで作成および定義されます。複数のルールセットを作成できますが、一度にアクティブにできるのは 1 つだけです。
- アクティブなルールセット内の受信ルールは、指定した順序で実行されます。
- 受信ルールを作成する前に、まず含めるべきルールセット を作成する必要があります。

オプションで、「Amazon Simple Email Service APIリファレンス」で説明されているように、`CreateReceiptRuleSet` API を使用して空の受信ルールセットを作成できます。次に、Amazon SESコンソールまたは `CreateReceiptRule` API を使用して受信ルールを追加できます。

ウォークスルーに進む前に、受信者ベースのEメール受信を使用するために必要なすべての前提条件を満たしていることを確認してください。また、

前提条件

受信ルールを使用して受信者ベースのEメール制御を設定するには、次の前提条件を満たす必要があります。

1. エンドポイントが Amazon が E メール受信SESをサポート AWS リージョン している にあることを確認します。の「E メール受信エンドポイント」表には、 AWS リージョン が E メール受信SESをサポートしているすべての の E メール受信エンドポイントが AWS 全般のリファレンス 一覧表示されています。
2. まず、Amazon でドメイン ID を作成して検証する必要がありますSES。

3. 次に、MX レコードをドメインの設定に発行することで、ドメインのメールを受け入れるメールサーバーを指定する必要があります。 DNS(MX レコードは、Amazon を使用する AWS リージョンのメールを受信する Amazon SESエンドポイントを参照する必要があります) SES。
4. 最後に、受信ルールアクションを実行するには、Amazon に他の AWS リソースへのアクセスSES許可を付与する必要があります。

ルールセットと受信ルールの作成

このウォークスルーは、まずルールを含むルールセットを作成することから始まり、その後、ルールの作成 ウィザードを使用して、受信ルールを作成、定義、順序付けします。このウィザードには、ルール設定の定義、受信者条件の追加、アクションの追加、およびすべての設定の確認を行うための 4 つの画面があります。

コンソールを使用してルールセットと受信ルールを作成するには

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの[Configuration]で[Eメール受信]を選択します。

   > **Note**
   >
   > アカウントが で E メール受信をサポートしていない場合、E AWS リージョン SES メール受信はSESコンソールの左側のナビゲーションペインに表示されません。「the section called “前提条件”」に記載されている最初の項目を参照してください。

3. [Email receiving] (メール受信) ペインの [Receipt rule sets] (受信ルールセット) タブで、[Create rule set] (ルールセットを作成) を選択します。
4. ルールセットの一意の名前を入力し、ルールセットの作成 を選択します。
5. ルールの作成 を選択すると、ルールの作成 ウィザードが開きます。
6. 受信ルール詳細 のルール設定の定義 ページで、ルール名 を入力します。
7. Status では、作成後に Amazon がこのルールを実行しないようにする場合にのみ、Enabled チェックボックスをオフにSESします。それ以外の場合は、このオプションを選択したままにします。
8. (オプション) セキュリティおよび保護オプションで、Transport Layer Security (TLS) で、Amazon SESに安全な接続経由で送信されない受信メッセージを拒否させる場合は必須を選択します。

9. （オプション) スパムおよびウイルススキャンで、Amazon が受信メッセージをスキャンSESしてスパムおよびウイルスがないか確認する場合は、有効を選択します。
10. [次へ] を選択して次のステップに進みます。
11. （オプション）受信者条件の追加 ページで、次の手順を使用して 1 つ以上の受信者の条件を指定します。受信ルールごとに 最大100 人の受信者条件を指定できます。

    a. 受信者条件 で、新しい受信者条件を追加 をクリックして、受信ルールを適用する対象の受信メールアドレスまたはドメインを指定します。次の表では、アドレス user@example.com を使用して受信者条件を指定する方法を示しています。

目的	指定する受取人	コメント
特定の E メールアドレスとマッチさせます。	user@example.com	また、ラベルが含まれるアドレスのバリエーション (user+123@example.com や user+xyz@example.com など) ともマッチさせます。ただし、ラベルが含まれるアドレスを指定すると、その特定のアドレスのみがマッチされます。
ドメイン内のすべてのアドレスとマッチさせるが、そのサブドメイン内のアドレスとはマッチさせません。	example.com	
特定のサブドメイン内のすべてのアドレスとマッチさせるが、親ドメイン内のアドレスとはマッチさせません。	subdomain.example.com	
すべてのサブドメイン内のすべてのアドレスとマッチさせるが、親ドメイン内の	.example.com	ドメイン名の前のピリオド (.) に注意してください。

目的	指定する受取人	コメント
アドレスとはマッチさせません。		
ドメイン内のすべてのアドレスとマッチさせて、さらにそのすべてのサブドメイン内のすべてのアドレスとマッチさせます。	example.com .example.com	2 つの受取人を別個に作成して、1 つにはドメイン名を使用し、別の 1 つにはピリオド (.) に続けてドメイン名を使用します。
すべての検証済みドメイン内のすべての受信者とマッチさせます。	[なし]	受信者フィールドを空にします。

> **Important**
>
> 複数の Amazon SESアカウントが共通ドメインで E メールを受信した場合 (例えば、同じ会社の複数のチームがそれぞれ個別の Amazon SESアカウントを持っている場合) 、Amazon はそれらのアカウントごとに一致するすべての受信ルールを同時にSES処理します。この動作により、あるアカウントではバウンスが生成され、別のアカウントでは E メールが承認されるという状況が発生する場合があります。Amazon を使用する組織内の他のチームと調整SESして、各アカウントが一意の受信ルールを使用し、それらのルールが重複しないようにすることをお勧めします。このような場合、グループやチームに固有のメールアドレスやサブドメインのみを使用するように受信ルールを設定するのが最適です。

b. 追加する受信者条件ごとに、このステップを繰り返します。受信者条件の追加が完了したら、[Next] を選択します。

12. アクションの追加 ページで、次の手順を使用して、受信ルールに 1 つ以上のアクションを追加します。

    a. 新しいアクションの追加 メニューを開き、以下のタイプのアクションの 1 つを選択します。

- ヘッダーを追加する - このアクションは、受信した E メールにカスタムヘッダーを追加します。
- バウンス応答を返す - このアクションは送信者にバウンス応答を返すことによって、受信メールを拒否します。
- Lambda 関数の呼び出し - このアクションは、 AWS Lambda 関数を介してコードを呼び出します。
- S3 バケットに配信する - このアクションは、Amazon Simple Storage Service (S3) バケットに、受信メールを保存します。
- Amazon SNS トピックへの公開 - このアクションは、Amazon Simple Notification Service (SNS) トピックに完全な E メールを発行します。
- ルールセットの停止 - このアクションは、受信ルールセットの評価を終了します。
- Amazon WorkMail との統合 - このアクションは Amazon と統合されます WorkMail。

これらのすべてのアクションの詳細については、アクションのオプション を参照してください。

b. 定義するアクションごとに、このステップを繰り返します。複数のアクションが定義されている場合は、アクションコンテナ内の上/下矢印を使用して、アクションを並べ替えることができます。[次]を選択して [Review (確認)] ページを開きます。

13. 確認 ページで、ルールの設定とアクションを確認します。変更する必要がある場合は、編集 オプションを選択するか、ウィンドウの左側にあるナビゲーションセクションを使用して、編集するコンテンツが含まれているページに直接移動します。オプションで、[Reorder (順序の並び替え)] 列で、上下矢印を使用して、確認 ページのアクション テーブルにリストアップされた順序を変更します。
14. 続行する準備ができたら、[Create rule (ルールの作成)] を選択します。
15. ルールセットをすぐに適用する場合は、ルールセットの確認ページで [Set as active] (アクティブに設定) を選択します。

作成後のルールの変更

ルールセットを作成したら、ルールセットとそれに含まれる受信ルールの両方を編集できます。編集できるだけでなく、新しいルールをすばやく作成できるように、ルールセットまたはそのルールを複製するオプションもあります。次のリストは、ルールセットおよび受信ルールに対して使用可能な変更を示しています。

- ルールセット は、名前、ステータス、作成日とともにリストされます。ルールセットの変更オプションは次のとおりです。
  - アクティブ/非アクティブとして設定 トグルボタンは、ステータスを設定するかどうかを切り替えます。
  - 複製 ボタンはルールセットをコピーします。一意の名前を指定するよう求められます。
  - 削除 ボタンはルールセットを削除します。この不可逆的な操作を確認するよう求められます。
- 受信ルール では、名前、ステータス、セキュリティ、および順序とともに一覧表示されます。受信ルールの変更オプションは次のとおりです。
  - 上/下矢印 をクリックして、ルールセット内のルールの実行順序を変更します。
  - 複製 ボタンは選択したルールのコピーを作成します。一意の名前を指定するよう求められます。
  - 編集 ボタンをクリックすると、選択したルールが開き、ルール設定、受信者の条件、アクションなどのパラメータを編集できます。
  - 削除 ボタンは選択したルールを削除します。この不可逆的な操作を確認するよう求められます。
  - ルールの作成 ボタンを使用して、現在のルールセットに新しいルールを作成し、追加できます。

アクションのオプション

Amazon SES による E メール受信に関する受信ルールにはそれぞれ、順序どおりに並べられたアクションのリストが含まれます。このセクションでは、各アクションタイプに固有のオプションについて説明します。

アクションタイプには以下のものがあります。

- ヘッダー追加アクション
- バウンス応答アクションを返す
- Lambda 関数のアクションを呼び出す
- S3 バケットアクションへの配信
- Amazon SNS トピックアクションへの公開
- ルールセットアクションの停止
- Amazon WorkMail アクションとの統合

ヘッダー追加アクション

ヘッダー追加アクションは、受信した E メールにカスタムヘッダーを追加します。このアクションは通常、他のアクションとの組み合わせでのみ使用します。このアクションには以下のオプションがあります。

- Header name - 追加するヘッダーの名前。1 ~ 50 文字で、英数字 (a-z、A-Z、0-9) とダッシュのみ使用できます。
- Header value - 追加するヘッダーの値。2048 文字未満で、改行文字 ("\r" または "\n") を含めることはできません。

バウンス応答アクションを返す

バウンスアクションは、送信者にバウンス応答を返すことによって E メールを拒否し、オプションで Amazon SNS を通じて通知します。このアクションには以下のオプションがあります。

- SMTP Reply Code - RFC 5321 で定義される SMTP 応答コード。
- SMTP Status Code - RFC 3463 で定義される SMTP 拡張ステータスコード。
- Message - バウンス E メールに含める、人間が読み取れるテキスト。
- Reply Sender - バウンス E メールの送信者の E メールアドレス。これは、バウンス E メールの送信元アドレスです。Amazon SES で検証されている必要があります。
- SNS Topic - バウンス E メールが送信されたときにオプションで通知するための Amazon SNS トピックの名前または ARN。Amazon SNS トピックの ARN の例は、arn:aws:sns:us-east-1:123456789012:MyTopic となります。SNS トピックを作成するを選んでアクションを設定するときに、Amazon SNS トピックを作成することもできます。Amazon SNS トピックの詳細については、Amazon Simple Notification Service デベロッパーガイドを参照してください。

> **Note**
>
> 選択する Amazon SNS トピックは、E メールを受け取る Amazon SES エンドポイントと同じ AWS リージョン内に存在する必要があります。

これらのフィールドに独自の値を入力するか、バウンスの理由に基づいて SMTP Reply Code、SMTP Status Code、Message の各フィールドに値を入力するテンプレートを選択することができます。次のテンプレートを使用できます。

- Mailbox Does Not Exist - SMTP 応答コード = 550、SMTP ステータスコード = 5.1.1
- Message Too Large - SMTP 応答コード = 552、SMTP ステータスコード = 5.3.4
- Message Full - SMTP 応答コード = 552、SMTP ステータスコード = 5.2.2
- Message Content Rejected - SMTP 応答コード = 500、SMTP ステータスコード = 5.6.1
- Unknown Failure - SMTP 応答コード = 554、SMTP ステータスコード = 5.0.0
- Temporary Failure - SMTP 応答コード = 450、SMTP ステータスコード = 4.0.0

フィールドにカスタムの値を入力して使用できる可能性があるその他のバウンスコードについては、RFC 3463 を参照してください。

Lambda 関数のアクションを呼び出す

Lambda アクションは、Lambda 関数を通じてコードを呼び出し、オプションで Amazon SNS を通じて通知します。このアクションには、次のオプションと要件があります。

オプション

- Lambda 関数 — Lambda 関数の ARN。Lambda 関数の ARN の例は、arn:aws:lambda:us-east-1:account-id:function:MyFunction となります。
- Invocation type - Lambda 関数の呼び出しタイプ。RequestResponse の呼び出しタイプは、関数の実行によって即時に応答が得られることを意味します。Event の呼び出しタイプは、関数が非同期に呼び出されることを意味します。ユースケースで同期実行が必要な場合を除き、呼び出しタイプには Event を使用することをお勧めします。

  RequestResponse の呼び出しには 30 秒のタイムアウトがあります。

  詳細については、AWS Lambda デベロッパーガイドの 「Lambda 関数を呼び出す」 を参照してください。
- SNS Topic - 指定されたLambda 関数がトリガーされたときに通知するための Amazon SNS トピックの名前または ARN。Amazon SNS トピックの ARN の例は、arn:aws:sns:us-east-1:123456789012:MyTopic となります。詳細については、Amazon Simple 通知サービス デベロッパーガイドの「Amazon SNS トピックの作成」を参照してください。

要件

- 選択する Lambda 関数は、E メールを受け取る Amazon SES エンドポイントと同じ AWS リージョン内に存在する必要があります。

- 選択する Amazon SNS トピックは、E メールを受け取る Amazon SES エンドポイントと同じ AWS リージョン内に存在する必要があります。

Lambda 関数の記述

E メールを処理するために、Lambda 関数を (Event 呼び出しタイプを使用して) 非同期で呼び出すことができます。Lambda 関数に渡されるイベントオブジェクトには、インバウンド E メールイベントに関係するメタデータが格納されます。このメタデータを使用して、Amazon S3 バケットからメッセージコンテンツにアクセスすることもできます。

メールフローを実際に制御するには、Lambda 関数を同期的に呼びだす (RequestResponse 呼びだしタイプを使用する) 必要があります。Lambda 関数では、2 つの引数 (1 つ目の引数は `callback`、2 つ目の引数は `null`、`disposition`、あるいは `STOP_RULE` に設定された `STOP_RULE_SET` プロパティ) を指定して `CONTINUE` メソッドを呼び出す必要があります。2 つ目の引数が `null` であるか、有効な `disposition` プロパティが指定されていない場合は、`CONTINUE` の場合と同様に、メールフローが継続し、後続のアクションとルールが処理されます。

例えば、Lambda 関数のコードの末尾に次の行を記述することで、受信ルールを停止することができます。

```
callback( null, { "disposition" : "STOP_RULE_SET" });
```

AWS Lambda のコードの例については、「Lambda 関数の例」を参照してください。概要レベルのユースケースの例については、「ユースケースの例」を参照してください。

入力形式

Amazon SES は、Lambda 関数に JSON 形式で情報を渡します。最上位レベルのオブジェクトには `Records` 配列が格納され、この配列には `eventSource`、`eventVersion`、および `ses` のプロパティが入力されます。`ses` オブジェクトには `receipt` オブジェクトと `mail` オブジェクトが格納されますが、これらは「通知の内容」で説明する Amazon SNS 通知とまったく同じ形式です。

Amazon SES が Lambda に渡すデータには、メッセージに関するメタデータと、複数の E メールヘッダーが含まれます。ただし、メッセージの本文は含まれません。

以下に、Amazon SES が Lambda 関数に提供する入力の構造を概要レベルで示します。

```
{
```

```
   "Records": [
      {
        "eventSource": "aws:ses",
        "eventVersion": "1.0",
        "ses": {
           "receipt": {
               <same contents as SNS notification>
            },
           "mail": {
               <same contents as SNS notification>
           }
         }
      }
   ]
}
```

戻り値

Lambda 関数では、次のいずれかの値を返すことにより、メールフローを制御できます。

- STOP_RULE - 現在の受信ルールでの後続のアクションは処理されませんが、後続の受信ルールは処理可能です。
- STOP_RULE_SET - 後続のアクションまたは受信のルールは処理されません。
- CONTINUE またはそれ以外の無効な値 - 後続のアクションおよび受信ルールが処理可能であることを意味します。

次のトピックでは、受信メールイベントのサンプル、高レベルのユースケースの例と、AWS Lambda コード例について説明します。

- ユースケースの例
- Lambda 関数の例

ユースケースの例

以下の例では、Lambda 関数の結果を使用してメールフローを制御するように設定できるルールの概要をいくつか示します。これらの例の多くでは、デモの目的で、S3 アクションを結果として使用します。

ユースケース 1: すべてのドメインにわたってスパムを削除する

この例では、すべてのドメインにわたってスパムを削除するグローバルなルールを示します。ルール 2 と 3 は、すべてのドメインにわたってスパムを削除した後で、ドメイン固有のルールを適用できることを示すために含めてあります。

ルール 1

受信者リスト : 空。したがって、このルールは、検証済みのすべてのドメインに含まれるすべての受取人に適用されます。

アクション

1. E メールがスパムである場合に `STOP_RULE_SET` を返す Lambda アクション (非同期)。そうでない場合は `CONTINUE` を返します。「Lambda 関数の例」で示した、スパムの削除に関する Lambda 関数の例を参照してください。

ルール 2

受信者リスト: example1.com

アクション

1. 任意のアクション。

ルール 3

受信者リスト: example2.com

アクション

1. 任意のアクション。

ユースケース 2: すべてのドメインにわたってスパムをバウンスする

この例では、すべてのドメインにわたってスパムをバウンスするグローバルなルールを示します。ルール 2 と 3 は、すべてのドメインにわたってスパムをバウンスした後で、ドメイン固有のルールを適用できることを示すために含めてあります。

ルール 1

受信者リスト : 空。したがって、このルールは、検証済みのすべてのドメインに含まれるすべての受取人に適用されます。

アクション

1. E メールがスパムである場合に CONTINUE を返す Lambda アクション (非同期)。そうでない場合は STOP_RULE を返します。
2. バウンスアクション ("500 5.6.1. Message content rejected")。
3. 停止アクション。

ルール 2

受信者リスト: example1.com

アクション

1. 任意のアクション

ルール 3

受信者リスト: example2.com

アクション

1. 任意のアクション

ユースケース 3: 固有のルールを適用する

この例では、停止アクションを使用して、E メールが複数のルールによって処理されるのを防ぐ方法を示します。この例では、特定のアドレスに対して 1 つのルールを設定し、ドメインに含まれるすべての E メールアドレスに対して別のルールを設定します。停止アクションを使用することで、特定の E メールアドレスに関するルールと一致するメッセージは、ドメインに適用される汎用的なルールでは処理されません。

ルール 1

受信者リスト: user@example.com

アクション

1. Lambda アクション (非同期)。
2. 停止アクション。

ルール 2

受信者リスト: example.com

アクション

1. 任意のアクション。

ユースケース 4: CloudWatch にメールイベントを口グ付けする

この例では、メールを Amazon SES に保存する前に、システムを通過するすべてのメールの監査口グを保持する方法を示します。

ルール 1

受信者リスト: example.com

アクション

1. CloudWatch log にイベントオブジェクトを書き込む Lambda アクション (非同期)。CloudWatch への Lambda 関数の例 口グ内のサンプル Lambda 関数。
2. S3 アクション。

ユースケース 5: DKIM に適合しないメールを削除する

この例では、すべての受信メールを Amazon S3 バケットに保存するが、特定の E メールアドレスに宛てた E メールのうち DKIM に合格したもののみを、自動化された E メールアプリケーションに送信する方法を示します。

ルール 1

受信者リスト: example.com

アクション

1. S3 アクション。
2. メッセージが DKIM に合格しない場合に `STOP_RULE_SET` を返す Lambda アクション (非同期)。そうでない場合は `CONTINUE` を返します。

ルール 2

受信者リスト: support@example.com

アクション

1. 自動化されたアプリケーションをトリガーする Lambda アクション (非同期)。

ユースケース 6: 件名の行に基づいてメールをフィルタリングする

この例では、ドメインの受信メールのうち、件名行に "discount" という語を含むすべてのメールを削除してから、自動化されたシステム宛てのメールについては特定の処理を実行し、ドメイン内のその他のすべての受信者宛のメールは別の方法で処理する方法を示します。

ルール 1

受信者リスト: example.com

アクション

1. 件名行に "discount" という語が含まれる場合に `STOP_RULE_SET` を返す Lambda アクション (非同期)。そうでない場合は `CONTINUE` を返します。

ルール 2

受信者リスト: support@example.com

アクション

1. バケット 1 を使用する S3 アクション。
2. 自動化されたアプリケーションをトリガーする Lambda アクション (非同期)。
3. 停止アクション。

ルール 3

受信者リスト: example.com

アクション

1. バケット 2 を使用する S3 アクション。
2. ドメインのそれ以外の受取人宛の E メールを処理する Lambda アクション (非同期)。

Lambda 関数の例

このトピックでは、メールフローを制御する Lambda 関数の例を示します。

例 1: スパムを削除する

この例では、少なくとも 1 つのスパムインジケータがあるメッセージの処理を停止します。

```
export const handler = async (event, context, callback) => {
    console.log('Spam filter');

    const sesNotification = event.Records[0].ses;
    console.log("SES Notification:\n", JSON.stringify(sesNotification, null, 2));

    // Check if any spam check failed
    if (sesNotification.receipt.spfVerdict.status === 'FAIL'
            || sesNotification.receipt.dkimVerdict.status === 'FAIL'
            || sesNotification.receipt.spamVerdict.status === 'FAIL'
            || sesNotification.receipt.virusVerdict.status === 'FAIL') {

        console.log('Dropping spam');

        // Stop processing rule set, dropping message
        callback(null, {'disposition':'STOP_RULE_SET'});
    } else {
        callback(null, {'disposition':'CONTINUE'});
    }
};
```

例 2: 特定のヘッダーが見つかった場合に続行する

この例では、E メールに特定のヘッダー値が含まれる場合にのみ、現在のルールの処理を続行します。

```
export const handler = async (event, context, callback) => {
    console.log('Header matcher');

    const sesNotification = event.Records[0].ses;
    console.log("SES Notification:\n", JSON.stringify(sesNotification, null, 2));

    // Iterate over the headers
    for (let index in sesNotification.mail.headers) {
        const header = sesNotification.mail.headers[index];
```

```
        // Examine the header values
        if (header.name === 'X-Header' && header.value === 'X-Value') {
            console.log('Found header with value.');
            callback(null, {'disposition':'CONTINUE'});
            return;
        }
    }

    // Stop processing the rule if the header value wasn't found
    callback(null, {'disposition':'STOP_RULE'});
};
```

例 3: Amazon S3 からの E メールの取得

この例では、Amazon S3 からraw Eメールを取得して処理します。

> **Note**
>
> - 先に S3 アクションを使用して Amazon S3 に E メールを書き込む必要があります。
> - Lambda 関数に S3 バケットからオブジェクトを取得するIAMアクセス許可があることを確認します。詳細については、この AWS re:Post 記事を参照してください。
> - デフォルトの Lambda 実行タイムアウトがワークフローに対して短すぎる可能性があります。それを増やすことを検討してください。

```
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3";
const bucketName = '<Your Bucket Name>';

export const handler = async (event, context, callback) => {
    const client = new S3Client();
    console.log('Process email');

    var sesNotification = event.Records[0].ses;
    console.log("SES Notification:\n", JSON.stringify(sesNotification, null, 2));
    console.log("MessageId: " + sesNotification.mail.messageId)

    const getObjectCommand = new GetObjectCommand({
        Bucket: bucketName,
        Key: sesNotification.mail.messageId
    });
```

```
    try {
        const response = await client.send(getObjectCommand);
        const receivedMail = await response.Body.transformToString();
        console.log(receivedMail);
        callback(null, {'disposition':'CONTINUE'})
    } catch (e) {
        // Perform error handling here
        console.log("Encountered S3 client error: "+ e, e.stack);
        callback(null, {'disposition':'STOP_RULE_SET'})
    }
};
```

例 4: DMARC認証に失敗したメッセージのバウンス

この例では、受信 E メールがDMARC認証に失敗した場合にバウンスメッセージを送信します。

> **Note**
>
> - この例を使用する場合は、E メール受信ドメインに `emailDomain` 環境変数の値を設定します。
> - Lambda 関数に、バウンスメッセージを送信する SES ID の`ses:SendBounce`アクセス許可があることを確認します。

```
import { SESClient, SendBounceCommand } from "@aws-sdk/client-ses";
const sesClient = new SESClient();
// Assign the emailDomain environment variable to a constant.
const emailDomain = process.env.emailDomain;

export const handler = async (event, context, callback) => {
    console.log('Spam filter starting');

    const sesNotification = event.Records[0].ses;
    const messageId = sesNotification.mail.messageId;
    const receipt = sesNotification.receipt;

    console.log('Processing message:', messageId);

    // If DMARC verdict is FAIL and the sending domain's policy is REJECT
    // (p=reject), bounce the email.
    if (receipt.dmarcVerdict.status === 'FAIL'
```

```
        && receipt.dmarcPolicy.status === 'REJECT') {
        // The values that make up the body of the bounce message.
        const sendBounceParams = {
            BounceSender: `mailer-daemon@${emailDomain}`,
            OriginalMessageId: messageId,
            MessageDsn: {
                ReportingMta: `dns; ${emailDomain}`,
                ArrivalDate: new Date(),
                ExtensionFields: [],
            },
            // Include custom text explaining why the email was bounced.
            Explanation: "Unauthenticated email is not accepted due to the sending
domain's DMARC policy.",
            BouncedRecipientInfoList: receipt.recipients.map((recipient) => ({
                Recipient: recipient,
                // Bounce with 550 5.6.1 Message content rejected
                BounceType: 'ContentRejected',
            })),
        };

        console.log('Bouncing message with parameters:');
        console.log(JSON.stringify(sendBounceParams, null, 2));

        const sendBounceCommand = new SendBounceCommand(sendBounceParams);

        // Try to send the bounce.
        try {
          const response = await sesClient.send(sendBounceCommand);
          console.log(response);
          console.log(`Bounce for message ${messageId} sent, bounce message ID:
${response.MessageId}`);
          // Stop processing additional receipt rules in the rule set.
          callback(null, {disposition: 'STOP_RULE_SET'});
        } catch (e) {
          // If something goes wrong, log the issue.
          console.log(`An error occurred while sending bounce for message:
${messageId}`, e);
          // Perform any additional error handling here
          callback(e)
        }

    // If the DMARC verdict is anything else (PASS, QUARANTINE or GRAY), accept
    // the message and process remaining receipt rules in the rule set.
    } else {
```

```
        console.log('Accepting message:', messageId);
        callback(null, {disposition: 'CONTINUE'});
    }
};
```

S3 バケットアクションへの配信

S3 バケットへの配信アクションは、メールを S3 バケットに配信し、必要に応じて SNS などを介してユーザーに通知します。このアクションには以下のオプションがあります。

- S3 バケット - 受信した E メールを保存する S3 バケットの名前。アクションを設定する際に [S3 バケットを作成する] をクリックして、新しい S3 バケットを作成することもできます。Amazon SES は 通常は多目的インターネットメール拡張 (MIME) 形式の、変更を加えていない raw E メールを格納する文字列を提供します。MIME 形式の詳細については、RFC 2045 を参照してください。

  > **Important**
  >
  > - Amazon S3 バケットは、SES the section called “E メールの受信” が利用可能なリージョンに配置されている必要があります。それ以外の場合は、以下で説明する IAM ロールオプションを使用する必要があります。
  > - E メールを S3 バケットに保存する際の E メールの最大サイズ (ヘッダーを含む) は 40 MB です。
  > - SES は、デフォルトの保持期間で設定された Object Lock で有効化された S3 バケットにアップロードする受信ルールをサポートしていません。
  > - 独自の KMS キーを指定して S3 バケットに暗号化を適用するには、KMS キーエイリアスではなく、完全修飾 KMS キー ARN を使用してください。エイリアスを使用すると、バケット管理者ではなく、リクエスタに属する KMS キーでデータが暗号化される可能性があります。「クロスアカウント操作での暗号化の使用」を参照してください。

- オブジェクトキープレフィックス - S3 バケット内で使用するキー名のプレフィックス。キー名のプレフィックスを使用すると、フォルダ構造を使って S3 バケットを分類できます。例えば、E メールをオブジェクトキープレフィックスとして使用する場合、E メールは S3 バケットの 「E メール」という名前のフォルダに表示されます。
- メッセージの暗号化 – 受信した E メールメッセージを S3 バケットに配信する前に暗号化するオプション。

- KMS 暗号化キー – (メッセージの暗号化が選択されている場合に利用可能)。E メールを S3 バケットに保存する前に暗号化するために SES が使用すべき AWS KMS キー。KMS で作成したデフォルトの KMS キーまたはカスタマーマネージドキーを使用できます。

> **Note**
>
> 選択する KMS キーは、E メールを受け取る SES エンドポイントと同じ AWS リージョン内に配置されている必要があります。

- デフォルトの KMS キーを使用するには、SES コンソールで受信ルールを設定する際に、[aws/ses] を選択します。SES API を使用する場合は、`arn:aws:kms:REGION:AWSACCOUNTID:alias/aws/ses` 形式で ARN を指定することでデフォルトの KMS キーを指定できます。たとえば、AWS アカウント ID が 123456789012 であり、us-east-1 リージョンのデフォルトの KMS キーを使用する場合は、デフォルトの KMS キーの ARN は `arn:aws:kms:us-east-1:123456789012:alias/aws/ses` となります。デフォルトの KMS キーを使用する場合、キーの使用についての SES へのアクセス許可付与に関する追加のステップを実行する必要はありません。
- KMS で作成したカスタマーマネージドキーを使用するには、KMS キーの ARN を指定し、キーのポリシーにステートメントを追加して、SES にキーを使用するアクセス許可を付与します。アクセス権限の付与の詳細については、「E メール受信SESのために Amazon にアクセス許可を付与する」を参照してください。

SES での KMS の使用の詳細については、「AWS Key Management Service デベロッパーガイド」を参照してください。コンソールまたは API で KMS キーを指定しないと、SES は E メールの暗号化は行いません。

> **Important**
>
> メールは、保存のために S3 に送信される前に、SES が S3 暗号化クライアントを使用して暗号化します。S3 のサーバー側の暗号化を使用して暗号化されるわけではありません。つまり、このサービスには復号に KMS キーを使用するアクセス権がないため、S3 から E メールを取得した後、S3 暗号化クライアントを使用してメールを復号する必要があります。この暗号化クライアントは、AWS SDK for Java および AWS SDK for Ruby でのみ使用できます。詳細については、Amazon Simple Storage Serviceユーザーガイドを参照してください。

- IAM ロール – SES が S3 への配信アクション (Amazon S3 バケット、SNS トピック、KMS キー) のリソースにアクセスするために使用する IAM ロール。指定しない場合、各リソースに個別にアクセスするためのアクセス許可を SES に明示的に付与する必要があります。「E メール受信SES のために Amazon にアクセス許可を付与する」を参照してください。

  E メール受信が利用できないリージョンに配置された S3 バケットに書き込む場合は、ロールのインラインポリシーとして S3 アクセス許可ポリシーへの書き込み権限が付与された IAM ロールを使用する必要があります。このアクションのアクセス許可ポリシーは、次のとおり、コンソールから直接適用できます。

  1. [IAM ロール] フィールドで、[新しいロールの作成] を選択して名前を入力してから、[ロールの作成] を選択します。(このロールの IAM 信頼ポリシーは、バックグラウンドで自動的に生成されます)。
  2. IAM 信頼ポリシーが自動的に生成されたため、ユーザーが実行する必要があるのは、このアクションのアクセス許可ポリシーをロールに追加することのみです。[IAM ロール] フィールドで、[ロールの表示] をクリックして、IAM コンソールを開きます。
  3. [許可] タブで、[アクセス許可の追加]、[インラインポリシーを作成] を選択します。
  4. [アクセス許可を指定] ページの [ポリシーエディタ] で、[JSON] を選択します。
  5. IAM S3 アクションの ロールのアクセス許可 からアクセス許可ポリシーをコピーし、[ポリシーエディタ] に貼り付けて、赤字のテキストのデータを独自のデータに置き換えます。(エディタ内のコード例は、必ずすべて削除します)。
  6. [Next] を選択します。
  7. この IAM ロールのアクセス許可ポリシーの内容を確認し、[ポリシーの作成] をクリックしてポリシーを作成します。
  8. SES [ルールの作成]–[アクションの追加] ページが開いているブラウザタブをクリックして、ルール作成の残りのステップの実行を続行します。

- SNS トピック – E メールが S3 バケットに保存された際の通知のための Amazon SNS トピックの名前または ARN。SNS トピック ARN の例は、arn:aws:sns:us-east-1:123456789012:MyTopic となります。アクションを設定する際に [SNS トピックの作成] をクリックして、新しい SNS トピックを作成することもできます。SNS トピックの詳細については、「Amazon Simple Notification Service デベロッパーガイド」を参照してください。

  > **Note**
  >
  > - 選択する SNS トピックは、E メールを受け取る SES エンドポイントと同じ AWS リージョン内に配置されている必要があります。

- SES 受信ルールに関連付ける SNS トピックでは、お客様が管理する KMS キー暗号化のみを使用してください。SES が SNS に発行できるように、KMS キーポリシーを編集する必要があるためです。これと対照的に、AWS が管理する KMS キーポリシーは、設計上編集できません。

Amazon SNS トピックアクションへの公開

SNS アクションは、Amazon SNS 通知を使用してメールを公開します。通知には、完全なメールコンテンツが含まれます。このアクションには以下のオプションがあります。

- SNS トピック - E メールを公開する Amazon SNS トピックの名前または ARN。Amazon SNS 通知には、変更を加えていない raw E メールのコピーが含まれます。これは通常においては多目的インターネットメール拡張 (MIME) 形式です。MIME 形式の詳細については、RFC 2045 を参照してください。

> **Important**
>
> Amazon SNS の通知を通じて E メールを受信することを選択した場合、E メールの最大サイズ (ヘッダーを含む) は 150 KB です。それよりも大きいメールはバウンスします。このサイズよりも大きい E メールが予想される場合は、代わりに Amazon S3 バケットに E メールを保存してください。

Amazon SNS トピックの ARN の例は、arn:aws:sns:us-east-1:123456789012:MyTopic となります。SNS トピックを作成するを選んでアクションを設定するときに、Amazon SNS トピックを作成することもできます。Amazon SNS トピックの詳細については、Amazon Simple Notification Service デベロッパーガイドを参照してください。

> **Note**
>
> - 選択する Amazon SNS トピックは、E メールを受け取る Amazon SES エンドポイントと同じ AWS リージョン内に存在する必要があります。
> - SES 受信ルールに関連付ける SNS トピックでは、お客様が管理する KMS キー暗号化のみを使用してください。SES が SNS に発行できるように、KMS キーポリシーを編集する必要があるためです。これと対照的に、AWS が管理する KMS キーポリシーは、設計上編集できません。

- エンコーディング - Amazon SNS 通知内で E メールに使用するエンコーディング。UTF-8 は使用が容易ですが、別のエンコード形式でメッセージがエンコードされた場合にすべての特殊文字が保存されないことがあります。Base64 ではすべての特殊文字が保存されます。UTF-8 および Base64 の詳細については、それぞれ RFC 3629 および RFC 4648 を参照してください。

E メールを受信すると、Amazon SES はアクティブな受信ルールセットのルールを実行します。Amazon SNS を使用して通知を送信するように受信ルールを設定できます。受信ルールでは、次の 2 つの異なるタイプの通知を送信できます。

- SNS アクションから送信される通知 - 受信ルールに SNS アクションを追加すると、E メールとその内容に関する情報が送信されます。メッセージが 150KB 以下の場合、この通知タイプには E メールの完全な MIME 本文も含まれます。
- 他のアクションタイプから送信される通知 - 他のアクションタイプ (バウンス、Lambda、ルールセットの停止または WorkMail アクションなど) を受信ルールに追加すると、オプションで Amazon SNS トピックを指定することができます。指定した場合は、これらのアクションが実行されると通知が送信されます。これらの通知には E メールに関する情報が含まれますが、E メールのコンテンツは含まれません。

次のトピックでは、これら通知の内容を説明し、それぞれの通知タイプの例を示します。

- Amazon SES の E メール受信通知の内容
- Amazon E SESメール受信の通知の例

Amazon SES の E メール受信通知の内容

E メール受信に関するすべての通知は、JavaScript Object Notation (JSON) 形式で Amazon Simple Notification Service (Amazon SNS) トピックに公開されます。

通知の例については、「通知の例」を参照してください。

目次

  - spamVerdict オブジェクト
  - spfVerdict オブジェクト
  - virusVerdict オブジェクト
- Mail オブジェクト
  - commonHeaders オブジェクト

トップレベル JSON オブジェクト

最上位の JSON オブジェクトには以下のフィールドが含まれます。

フィールド名	説明
`notificationType`	通知タイプ。このタイプの通知の場合、この値は常に `Received` です。
`receipt`	E メール配信に関する情報を格納するオブジェクト。
`mail`	通知が関係する E メールについての情報を格納するオブジェクト。
`content`	通常は多目的インターネットメール拡張 (MIME) 形式の、変更を加えていない raw E メールを格納する文字列。MIME 形式の詳細については、RFC 2045 を参照してください。 **Note** このフィールドは、通知が SNS アクションによってトリガーされたときだけ使用されます。その他のすべてのアクションによってトリガーされる通知には、このフィールドは含まれません。

receipt オブジェクト

receipt オブジェクトには以下のフィールドがあります。

フィールド名	説明
action	実行されたアクションに関する情報をカプセル化するオブジェクト。可能な値の一覧については、アクションオブジェクト を参照してください。
dkimVerdict	ドメインキーアイデンティファイドメール (DKIM) のチェックに合格したどうかを示すオブジェクト。可能な値の一覧については、dkimVerdict オブジェクト を参照してください。
dmarcPolicy	送信元ドメインの Domain-based Message Authentication、Reporting & Conformance (DMARC) の設定を示します。このフィールドは、メッセージが DMARC 認証に失敗した場合にのみ表示されます。 このフィールドに可能な値は次のとおりです。 • none: 送信元ドメインの所有者は、DMARC 認証に失敗するメッセージに対して特定のアクションをリクエストしません。 • quarantine : 送信元ドメインの所有者は、DMARC 認証に失敗するメッセージが受信者によって疑わしいものとして扱われるようリクエストします。 • reject: 送信元ドメインの所有者は、DMARC 認証に失敗するメッセージが拒否されるようリクエストします。
dmarcVerdict	Domain-based Message Authentication、Reporting & Conformance (DMARC) のチェックに

フィールド名	説明
	合格したかどうかを示すオブジェクト。可能な値の一覧については、dmarcVerdict オブジェクト を参照してください。
processingTimeMillis	Amazon SES がメッセージを受信した時点からアクションをトリガーした時点までの期間をミリ秒で表す文字列。
recipients	アクティブな受信ルールによって一致した受信者 (具体的には、エンベロープの RCPT TO アドレス)。ここにリストされるアドレスは、「destination 」の [the section called “Mail オブジェクト”] フィールドに記載されているものとは異なる場合があります。
spamVerdict	メッセージがスパムであるかどうかを示すオブジェクト。可能な値の一覧については、spamVerdict オブジェクト を参照してください。
spfVerdict	Sender Policy Framework (SPF) のチェックに合格したかどうかを示すオブジェクト。可能な値の一覧については、spfVerdict オブジェクト を参照してください。
timestamp	ISO 8601 フォーマットでアクションのトリガーが認定された日時を指定する文字列。
virusVerdict	メッセージにウイルスが含まれているかどうかを示すオブジェクト。可能な値の一覧については、virusVerdict オブジェクト を参照してください。

アクションオブジェクト

action オブジェクトには以下のフィールドがあります。

フィールド名	説明
type	実行されたアクションのタイプを示す文字列。使用できる値は、S3、SNS、Bounce、Lambda、Stop、および WorkMail です。
topicArn	通知が公開された Amazon SNS トピックの Amazon リソースネーム (ARN) を格納する文字列。
bucketName	メッセージが公開された Amazon S3 バケットの名前を格納する文字列。S3 アクションタイプでのみ使用されます。
objectKey	Amazon S3 バケット内の E メールを一意に識別する名前を格納する文字列。これは、「 messageId 」の the section called “Mail オブジェクト” と同じです。S3 アクションタイプでのみ使用されます。
smtpReplyCode	RFC 5321 で定義されている SMTP 応答コードを格納する文字列。バウンスアクションタイプでのみ使用されます。
statusCode	RFC 3463 で定義されている SMTP 拡張ステータスコードを格納する文字列。バウンスアクションタイプでのみ使用されます。
message	バウンスメッセージに含める、人間が読み取れるテキストを格納する文字列。バウンスアクションタイプでのみ使用されます。
sender	バウンスした E メールの送信元 E メールアドレスを格納する文字列。これは、バウンスメッセージの送信元のアドレスです。バウンスアクションタイプでのみ使用されます。

フィールド名	説明
functionArn	トリガーされた Lambda 関数の ARN を格納する文字列。Lambda アクションタイプでのみ使用されます。
invocationType	Lambda 関数の呼び出しタイプを格納する文字列。指定できる値は RequestResponse および Event です。Lambda アクションタイプでのみ使用されます。
organizationArn	Amazon WorkMail 組織の ARN を格納する文字列。WorkMail アクションタイプでのみ使用されます。

dkimVerdict オブジェクト

dkimVerdict オブジェクトには以下のフィールドがあります。

フィールド名	説明
status	DKIM 判定を格納する文字列。想定される値は次のとおりです。 • PASS: メッセージは DKIM 認証に合格しています。 • FAIL: メッセージは DKIM 認証に失敗しました。 • GRAY: メッセージが DKIM で署名されていないか、送信元ドメインと DKIM 署名ドメインが一致しません。 • PROCESSING_FAILED : Amazon SES で DKIM 署名をチェックできない問題が存在します。たとえば、DNS クエリに失敗しているか、DKIM 署名ヘッダーが正しくフォーマットされていません。

dmarcVerdict オブジェクト

dmarcVerdict オブジェクトには以下のフィールドがあります。

フィールド名	説明
status	DMARC 判定を格納する文字列。想定される値は次のとおりです。 • PASS: メッセージは DMARC 認証に合格しています。 • FAIL: メッセージは DMARC 認証に失敗しました。 • GRAY: 少なくとも 1 つの SPF または DKIM が認証にパスしましたが、送信元ドメインに DMARC ポリシーが存在しないか、p=none ポリシーを使用しています。 • PROCESSING_FAILED : Amazon SES で DMARC の判定を提供できない問題が存在します。

spamVerdict オブジェクト

spamVerdict オブジェクトには以下のフィールドがあります。

フィールド名	説明
status	スパムスキャンの結果を格納する文字列。想定される値は次のとおりです。 • PASS: スパムスキャンにより、メッセージにスパムが含まれている可能性は低いと判定されました。 • FAIL: スパムスキャンにより、メッセージにスパムが含まれている可能性が高いと判定されました。

フィールド名	説明
	• GRAY: Amazon SES でメールをスキャンしましたが、スパムかどうかを断定できませんでした。 • PROCESSING_FAILED : Amazon SES でメールの内容をスキャンできませんでした。たとえば、メールが有効な MIME メッセージではありません。

spfVerdict オブジェクト

spfVerdict オブジェクトには以下のフィールドがあります。

フィールド名	説明
status	SPF 判定を格納する文字列。想定される値は次のとおりです。 • PASS: メッセージは SPF 認証に合格しています。 • FAIL: メッセージは SPF 認証に失敗しました。 • GRAY: SPF の結果は none、softfail、または neutral です。 • PROCESSING_FAILED : Amazon SES で SPF レコードをチェックできない問題が存在します。たとえば、DNS クエリに失敗しています。

virusVerdict オブジェクト

virusVerdict オブジェクトには以下のフィールドがあります。

フィールド名	説明
status	ウイルススキャンの結果を格納する文字列。想定される値は次のとおりです。 • PASS: メッセージにウイルスは含まれていません。 • FAIL: メッセージにウイルスが含まれています。 • GRAY: Amazon SES でメールをスキャンしましたが、ウイルスが含まれているかどうかを断定できませんでした。 • PROCESSING_FAILED : Amazon SES でメールの内容をスキャンできません。たとえば、メールが有効な MIME メッセージではありません。

Mail オブジェクト

mail オブジェクトには以下のフィールドがあります。

フィールド名	説明
destination	受信 E メールの MIME ヘッダーからのすべての受信者アドレス (To: および CC: 受信者を含む) の完全なリスト。
messageId	Amazon SES によって E メールに割り当てられる一意の ID を格納する文字列。E メールが Amazon S3 に配信された場合、このメッセージ ID は、Amazon S3 バケットにメッセージを書き込むために使用される Amazon S3 オブジェクトキーにもなります。

フィールド名	説明
source	E メールの送信元の E メールアドレス (具体的には、エンベロープ MAIL FROM アドレス) を格納する文字列。
timestamp	E メールを受信した時刻を ISO8601 形式で格納する文字列。
headers	Amazon SES ヘッダーおよびカスタムヘッダー。各ヘッダーには、name および value のフィールドがあります。
commonHeaders	すべての E メールに共通するヘッダー。各ヘッダーには、name および value のフィールドがあります。
headersTruncated	通知でヘッダーが切り捨てられたかどうかを示す文字列。切り捨ては、ヘッダーが 10 KB を超える場合に発生します。指定できる値は true および false です。

commonHeaders オブジェクト

commonHeaders オブジェクトには、次の表に示すフィールドを含めることができます。このオブジェクトに存在するフィールドは、受信 E メールにどのフィールドが存在するかに応じて異なります。

フィールド名	説明
messageId	元のメッセージの ID。
date	Amazon SES がメッセージを受信した日時。
to	E メールの To ヘッダー。
cc	E メールの CC ヘッダー。

フィールド名	説明
bcc	E メールの BCC ヘッダー。
from	E メールの From ヘッダー。
sender	E メールの Sender ヘッダー。
returnPath	E メールの Return-Path ヘッダー。
replyTo	E メールの Reply-To ヘッダー。
subject	E メールの Subject ヘッダー。

Amazon E SESメール受信の通知の例

このセクションでは、以下のタイプの通知の例を紹介します。

- SNS アクションの結果として送信される通知。
- 別のタイプのアクションの結果として送信される通知 (アラート通知)

SNS アクションの通知

このセクションでは、SNSアクション通知の例を示します。前述のアラート通知とは異なり、E メールを含むcontentセクションが含まれています。これは、通常、多目的インターネットメール拡張 (MIME) 形式です。

```
{
  "notificationType":"Received",
  "receipt":{
    "timestamp":"2015-09-11T20:32:33.936Z",
    "processingTimeMillis":222,
    "recipients":[
      "recipient@example.com"
    ],
    "spamVerdict":{
      "status":"PASS"
    },
    "virusVerdict":{
      "status":"PASS"
```

```
    },
    "spfVerdict":{
      "status":"PASS"
    },
    "dkimVerdict":{
      "status":"PASS"
    },
    "action":{
      "type":"SNS",
      "topicArn":"arn:aws:sns:us-east-1:012345678912:example-topic"
    }
  },
  "mail":{
    "timestamp":"2015-09-11T20:32:33.936Z",
    "source":"61967230-7A45-4A9D-BEC9-87CBCF2211C9@example.com",
    "messageId":"d6iitobk75ur44p8kdnnp7g2n800",
    "destination":[
      "recipient@example.com"
    ],
    "headersTruncated":false,
    "headers":[
      {
        "name":"Return-Path",

 "value":"<0000014fbe1c09cf-7cb9f704-7531-4e53-89a1-5fa9744f5eb6-000000@amazonses.com>"
      },
      {
        "name":"Received",
        "value":"from a9-183.smtp-out.amazonses.com (a9-183.smtp-out.amazonses.com
 [54.240.9.183]) by inbound-smtp.us-east-1.amazonaws.com with SMTP id
 d6iitobk75ur44p8kdnnp7g2n800 for recipient@example.com; Fri, 11 Sep 2015 20:32:33
 +0000 (UTC)"
      },
      {
        "name":"DKIM-Signature",
        "value":"v=1; a=rsa-sha256; q=dns/txt; c=relaxed/simple;
 s=ug7nbtf4gccmlpwj322ax3p6ow6yfsug; d=amazonses.com; t=1442003552;
 h=From:To:Subject:MIME-Version:Content-Type:Content-Transfer-Encoding:Date:Message-
ID:Feedback-ID; bh=DWr3IOmYWoXCA9ARqGC/UaODfghffiwFNRIb2Mckyt4=;
 b=p4ukUDSFqhqiub+zPR0DW1kp7oJZakrzupr6LBe6sUuvqpBkig56UzUwc29rFbJF
 hlX3Ov7DeYVNoN38stqwsF8ivcajXpQsXRC1cW9z8x875J041rClAjV7EGbLmudVpPX
 4hHst1XPyX5wmgdHIhmUuh8oZKpVqGi6bHGzzf7g="
      },
      {
```

```
        "name":"From",
        "value":"sender@example.com"
      },
      {
        "name":"To",
        "value":"recipient@example.com"
      },
      {
        "name":"Subject",
        "value":"Example subject"
      },
      {
        "name":"MIME-Version",
        "value":"1.0"
      },
      {
        "name":"Content-Type",
        "value":"text/plain; charset=UTF-8"
      },
      {
        "name":"Content-Transfer-Encoding",
        "value":"7bit"
      },
      {
        "name":"Date",
        "value":"Fri, 11 Sep 2015 20:32:32 +0000"
      },
      {
        "name":"Message-ID",
        "value":"<61967230-7A45-4A9D-BEC9-87CBCF2211C9@example.com>"
      },
      {
        "name":"X-SES-Outgoing",
        "value":"2015.09.11-54.240.9.183"
      },
      {
        "name":"Feedback-ID",
        "value":"1.us-east-1.Krv2FKpFdWV+KUYw3Qd6wcpPJ4Sv/pOPpEPSHn2u2o4=:AmazonSES"
      }
    ],
    "commonHeaders":{

"returnPath":"0000014fbe1c09cf-7cb9f704-7531-4e53-89a1-5fa9744f5eb6-000000@amazonses.com",
      "from":[
```

```
        "sender@example.com"
      ],
      "date":"Fri, 11 Sep 2015 20:32:32 +0000",
      "to":[
        "recipient@example.com"
      ],
      "messageId":"<61967230-7A45-4A9D-BEC9-87CBCF2211C9@example.com>",
      "subject":"Example subject"
    }
  },
  "content":"Return-Path: <61967230-7A45-4A9D-BEC9-87CBCF2211C9@example.com>\r
\nReceived: from a9-183.smtp-out.amazonses.com (a9-183.smtp-out.amazonses.com
 [54.240.9.183])\r\n by inbound-smtp.us-east-1.amazonaws.com with SMTP id
 d6iitobk75ur44p8kdnnp7g2n800\r\n for recipient@example.com;\r\n Fri, 11 Sep 2015
 20:32:33 +0000 (UTC)\r\nDKIM-Signature: v=1; a=rsa-sha256; q=dns/txt; c=relaxed/
simple;\r\n\ts=ug7nbtf4gccmlpwj322ax3p6ow6yfsug; d=amazonses.com; t=1442003552;\r\n
\th=From:To:Subject:MIME-Version:Content-Type:Content-Transfer-Encoding:Date:Message-
ID:Feedback-ID;\r\n\tbh=DWr3IOmYWoXCA9ARqGC/UaODfghffiwFNRIb2Mckyt4=;\r\n
\tb=p4ukUDSFqhqiub+zPR0DW1kp7oJZakrzupr6LBe6sUuvqpBkig56UzUwc29rFbJF\r\n
\thlX3Ov7DeYVNoN38stqwsF8ivcajXpQsXRC1cW9z8x875J041rClAjV7EGbLmudVpPX\r\n
\t4hHst1XPyX5wmgdHIhmUuh8oZKpVqGi6bHGzzf7g=\r\nFrom: sender@example.com\r\nTo:
 recipient@example.com\r\nSubject: Example subject\r\nMIME-Version: 1.0\r\nContent-
Type: text/plain; charset=UTF-8\r\nContent-Transfer-Encoding: 7bit\r\nDate: Fri, 11 Sep
 2015 20:32:32 +0000\r\nMessage-ID: <61967230-7A45-4A9D-BEC9-87CBCF2211C9@example.com>
\r\nX-SES-Outgoing: 2015.09.11-54.240.9.183\r\nFeedback-ID: 1.us-east-1.Krv2FKpFdWV
+KUYw3Qd6wcpPJ4Sv/pOPpEPSHn2u2o4=:AmazonSES\r\n\r\nExample content\r\n"
}
```

アラート通知

このセクションでは、S3 アクションによってトリガーできる Amazon SNS通知の例を示します。Lambda アクション、バウンスアクション、停止アクション、および WorkMail アクションによってトリガーされる通知は似ています。通知には E メールに関する情報が含まれますが、E メールのコンテンツそのものは含まれません。

```
{
   "notificationType": "Received",
    "receipt": {
      "timestamp": "2015-09-11T20:32:33.936Z",
      "processingTimeMillis": 406,
      "recipients": [
        "recipient@example.com"
      ],
```

```
      "spamVerdict": {
        "status": "PASS"
      },
      "virusVerdict": {
        "status": "PASS"
      },
      "spfVerdict": {
        "status": "PASS"
      },
      "dkimVerdict": {
        "status": "PASS"
      },
      "action": {
        "type": "S3",
        "topicArn": "arn:aws:sns:us-east-1:012345678912:example-topic",
        "bucketName": "amzn-s3-demo-bucket",
        "objectKey": "\email"
      }
    },
    "mail": {
      "timestamp": "2015-09-11T20:32:33.936Z",
      "source":
"0000014fbe1c09cf-7cb9f704-7531-4e53-89a1-5fa9744f5eb6-000000@amazonses.com",
      "messageId": "d6iitobk75ur44p8kdnnp7g2n800",
      "destination": [
        "recipient@example.com"
      ],
      "headersTruncated": false,
      "headers": [
        {
          "name": "Return-Path",
          "value":
"<0000014fbe1c09cf-7cb9f704-7531-4e53-89a1-5fa9744f5eb6-000000@amazonses.com>"
        },
        {
          "name": "Received",
          "value": "from a9-183.smtp-out.amazonses.com (a9-183.smtp-out.amazonses.com
[54.240.9.183]) by inbound-smtp.us-east-1.amazonaws.com with SMTP id
d6iitobk75ur44p8kdnnp7g2n800 for recipient@example.com; Fri, 11 Sep 2015 20:32:33
+0000 (UTC)"
        },
        {
          "name": "DKIM-Signature",
```

```
          "value": "v=1; a=rsa-sha256; q=dns/txt; c=relaxed/simple;
 s=ug7nbtf4gccmlpwj322ax3p6ow6yfsug; d=amazonses.com; t=1442003552;
 h=From:To:Subject:MIME-Version:Content-Type:Content-Transfer-Encoding:Date:Message-
ID:Feedback-ID; bh=DWr3IOmYWoXCA9ARqGC/UaODfghffiwFNRIb2Mckyt4=;
 b=p4ukUDSFqhqiub+zPR0DW1kp7oJZakrzupr6LBe6sUuvqpBkig56UzUwc29rFbJF
 hlX3Ov7DeYVNoN38stqwsF8ivcajXpQsXRC1cW9z8x875J041rClAjV7EGbLmudVpPX
 4hHst1XPyX5wmgdHIhmUuh8oZKpVqGi6bHGzzf7g="
        },
        {
          "name": "From",
          "value": "sender@example.com"
        },
        {
          "name": "To",
          "value": "recipient@example.com"
        },
        {
          "name": "Subject",
          "value": "Example subject"
        },
        {
          "name": "MIME-Version",
          "value": "1.0"
        },
        {
          "name": "Content-Type",
          "value": "text/plain; charset=UTF-8"
        },
        {
          "name": "Content-Transfer-Encoding",
          "value": "7bit"
        },
        {
          "name": "Date",
          "value": "Fri, 11 Sep 2015 20:32:32 +0000"
        },
        {
          "name": "Message-ID",
          "value": "<61967230-7A45-4A9D-BEC9-87CBCF2211C9@example.com>"
        },
        {
          "name": "X-SES-Outgoing",
          "value": "2015.09.11-54.240.9.183"
        },
```

```
        {
          "name": "Feedback-ID",
          "value": "1.us-east-1.Krv2FKpFdWV+KUYw3Qd6wcpPJ4Sv/pOPpEPSHn2u2o4=:AmazonSES"
        }
      ],
      "commonHeaders": {
        "returnPath":
"0000014fbe1c09cf-7cb9f704-7531-4e53-89a1-5fa9744f5eb6-000000@amazonses.com",
        "from": [
          "sender@example.com"
        ],
        "date": "Fri, 11 Sep 2015 20:32:32 +0000",
        "to": [
          "recipient@example.com"
        ],
        "messageId": "<61967230-7A45-4A9D-BEC9-87CBCF2211C9@example.com>",
        "subject": "Example subject"
      }
    }
  }
```

ルールセットアクションの停止

停止アクションは、受信ルールセットの評価を終了し、オプションで、Amazon SNS を通じて通知します。このアクションには以下のオプションがあります。

- SNS Topic - 停止アクションが実行されたときに通知するための Amazon SNS トピックの名前または ARN。Amazon SNS トピックの ARN の例は、arn:aws:sns:us-east-1:123456789012:MyTopic となります。SNS トピックを作成するを選んでアクションを設定するときに、Amazon SNS トピックを作成することもできます。Amazon SNS トピックの詳細については、Amazon Simple Notification Service デベロッパーガイドを参照してください。

> **Note**
>
> 選択する Amazon SNS トピックは、E メールを受け取る Amazon SES エンドポイントと同じ AWS リージョン内に存在する必要があります。

Amazon WorkMail アクションとの統合

[WorkMail] アクションは、Amazon WorkMail と連携します。Amazon WorkMail ですべての E メール処理を実行する場合は、Amazon WorkMail がこの設定を担当するため、通常このアクションを直接使用することはありません。このアクションには以下のオプションがあります。

- Organization ARN — Amazon WorkMail の組織の ARN。Amazon WorkMail の組織の ARN は、`arn:aws:workmail:`*`region`*`:`*`account_ID`*`:organization/`*`organization_ID`* となります。ここで、
  - `region` は、Amazon SES および Amazon WorkMail を使用しているリージョンです。(これらは同じリージョンから使用する必要があります。) 例えば、us-east-1 です。
  - `account_ID` は、AWS アカウント ID です。AWS アカウント ID は、AWS マネジメントコンソールの「アカウント」ページを参照してください。
  - `organization_ID` は、組織を作成したときに Amazon WorkMail によって生成される一意の識別子です。組織 ID は、Amazon WorkMail コンソールで、組織の [Organization 設定] ページから確認できます。

  組織の完全な Amazon WorkMail ARN の例は、arn:aws:workmail:us-east-1:123456789012:organization/m-68755160c4cb4e29a2b2f8fb58f359d7 となります。Amazon WorkMail 組織の詳細については、Amazon WorkMail 管理者ガイドを参照してください。
- SNS Topic - Amazon WorkMail アクションが実行されたときに通知するための Amazon SNS トピックの名前または ARN。Amazon SNS トピックの ARN の例は、arn:aws:sns:us-east-1:123456789012:MyTopic となります。SNS トピックを作成するを選んでアクションを設定するときに、Amazon SNS トピックを作成することもできます。Amazon SNS トピックの詳細については、Amazon Simple Notification Service デベロッパーガイドを参照してください。

> **Note**
>
> 選択する Amazon SNS トピックは、E メールを受け取る Amazon SES エンドポイントと同じ AWS リージョン内に存在する必要があります。

> **Note**
>
> Amazon SES は WorkMail が利用可能なリージョンでのみ WorkMail アクションをサポートします。AWS 全般のリファレンス の Amazon WorkMail エンドポイントとクォータを参照してください。

IP アドレスフィルタのコンソールウォークスルーの作成

ここでは、Amazon SES コンソールを使用して IP アドレスフィルタを設定する手順について説明します。IP アドレスフィルタリングにより、広範な制御が可能になります。これらの IP フィルタを使用すると、特定の IP アドレスまたは IP アドレス範囲からのすべてのメッセージを明示的にブロックまたは許可できます。

オプションで、Amazon Simple Email Service APIリファレンスで説明されているように、`CreateReceiptFilter`APIを使用してIPアドレスフィルターを作成できます。

> **Note**
>
> 既知の IP アドレスで構成される有限のリストからのメールのみを受信する場合は、`0.0.0.0/0` を含むブロックリストをセットアップし、信頼する IP アドレスを含む許可リストをセットアップします。この設定では、デフォルトではすべての IP アドレスがブロックされ、明示的に指定する IP アドレスからのメールのみが許可されます。

前提条件

IP アドレスフィルタを使用した受信者ベースのEメール制御の設定に進む前に、次の前提条件を満たす必要があります。

1. まず、Amazon SESでドメインIDの作成と検証 を行う必要があります。
2. 次に、ドメインのメールを受け付けることができるメールサーバーをMX レコードの発行 でドメインの DNS 設定に追加します。(MX レコードは、Amazon SES を使用する AWS リージョンに関し、メールを受信する Amazon SES エンドポイントを参照する必要があります。)

IP アドレスフィルタの作成

コンソールを使用してIPアドレスフィルターを作成するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションペインで、[Eメール受信]を選択します。
3. [IP アドレスフィルタ]タブを選択します。
4. [フィルタの作成]を選択します。
5. フィルタの一意の名前を入力します。フィールドの凡例に構文要件が示されます。 (名前は 64 文字未満で、英数字、ハイフン (-)、アンダースコア (_)、ピリオド (.) を使用する必要があります。名前は文字または数字で始まり、完結していなければなりません。)
6. IPアドレスまたは IP アドレス範囲を入力します。フィールドの凡例には、Classless Inter-Domain Routing(CIDR) 構文で指定されている例が示されます。1 つの IP アドレスの例は 10.0.1 です。 IP アドレス範囲の例は 10.0.0.1/24 となります。 CIDR 表記の詳細については、RFC 2317 を参照してください。)
7. ブロックまたは許可 ラジオボタンを選択して、ポリシータイプ を選択します。
8. [フィルターの作成] をクリックします。
9. 別の IP フィルタを追加する場合は、フィルターの作成 を選択し、追加するフィルタごとに、上記のステップを繰り返します。
10. IP アドレスフィルターを削除する場合は、フィルターを選択して、削除 ボタンを使用します。

Amazon SES によるメール受信のメトリクスの表示

Amazon SES で E メール受信を有効にしていて、E メールの受信ルールを作成している場合は、Amazon CloudWatch を使用してそれらの受信ルールセットとルールのメトリクスを表示できます。

CloudWatch コンソールで、[メトリクス] > [すべてのメトリクス] > [SES] > [受信ルールセットのメトリクス] および [受信ルールのメトリクス] の順に選択すると、メトリクスが表示されます。

> **Note**
>
> [受信ルールセットのメトリクス] と [受信ルールのメトリクス] は、以下に該当する場合にのみ、SES に表示されます。

- E メール受信を有効にした
- 受信ルールを作成した
- ルールのいずれかに一致するメールを受信した

以下のメッセージのメトリクスが利用可能です。

- メッセージの受信

スコープ	メトリクス	説明	ディメンション
受信ルールセットのメトリクス	受信済み	SES は、少なくとも 1 つのルールが適用されるメッセージを正常に受信しました。このメトリクスは、値として 1 のみを持つことができます。	RuleSetName
受信ルールのメトリクス	受信済み	SES はメッセージを正常に受信し、適用されたルールの処理を試みます。このメトリクスは、値として 1 のみを持つことができます。	RuleName

- メッセージの発行

スコープ	メトリクス	説明	ディメンション
受信ルールセットのメトリクス	PublishSuccess	SES は、ルールセット内の適用されるすべてのルールを正常に実行しました。	RuleSetName
受信ルールのメトリクス	PublishSuccess	SES は、受信メッセージに適用されるルールを正常に実行しました。	RuleName
受信ルールセットのメトリクス	PublishFailure	SES がルールセット内のルールを実行しようとしてエラーが発生しました。実行は再試行されます。	RuleSetName

スコープ	メトリクス	説明	ディメンション
受信ルールのメトリクス	PublishFailure	SES がルール内のアクションを実行しようとして エラーが発生しました。エラーによっては、実行が再試行される場合があります。	RuleName
受信ルールセットのメトリクス	PublishExpired	ルールの実行が 36 時間以内に成功しなかったか、再試行不可能なエラーが発生したため、SES は、これ以上ルールの実行を再試行しません。	RuleSetName
受信ルールのメトリクス	PublishExpired	ルールの実行が 36 時間以内に成功しなかったため、SES は、これ以上ルールのアクションの実行を再試行しません。	RuleName

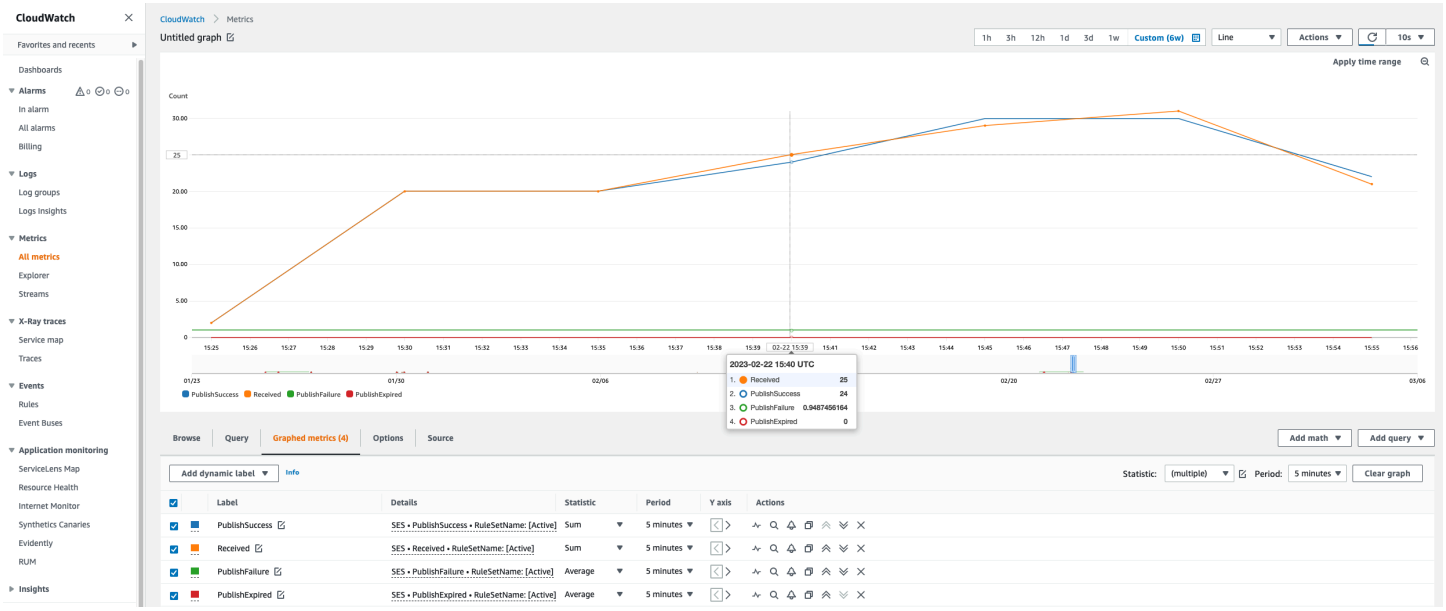
Note

- 上に示した表で、適用という用語は、送信者が IP フィルターによってブロックリストに登録されていないか、SES の内部ブロックリストに含まれていないこと、およびルール内に一致する受信者条件と TLS ポリシーがあることを意味します。
- 発行の失敗エラーは、いずれかの受信ルールのアクションで使用するよう設定されている Amazon S3 バケット、Amazon SNS トピック、または Lambda 関数へのアクセス許可を削除したか取り消した場合などに、発生することがあります。
- 一度にアクティブにできるルールセットは 1 つのみであるため、SES は集約メトリクスを発行します。これは、CloudWatch で選択した時間範囲でアクティブであったすべてのルールセットに対して RuleSetName:[Active] と表示されます。これには、アラーム設定を変更せずにルールセットを自由に変更できるという利点があります。

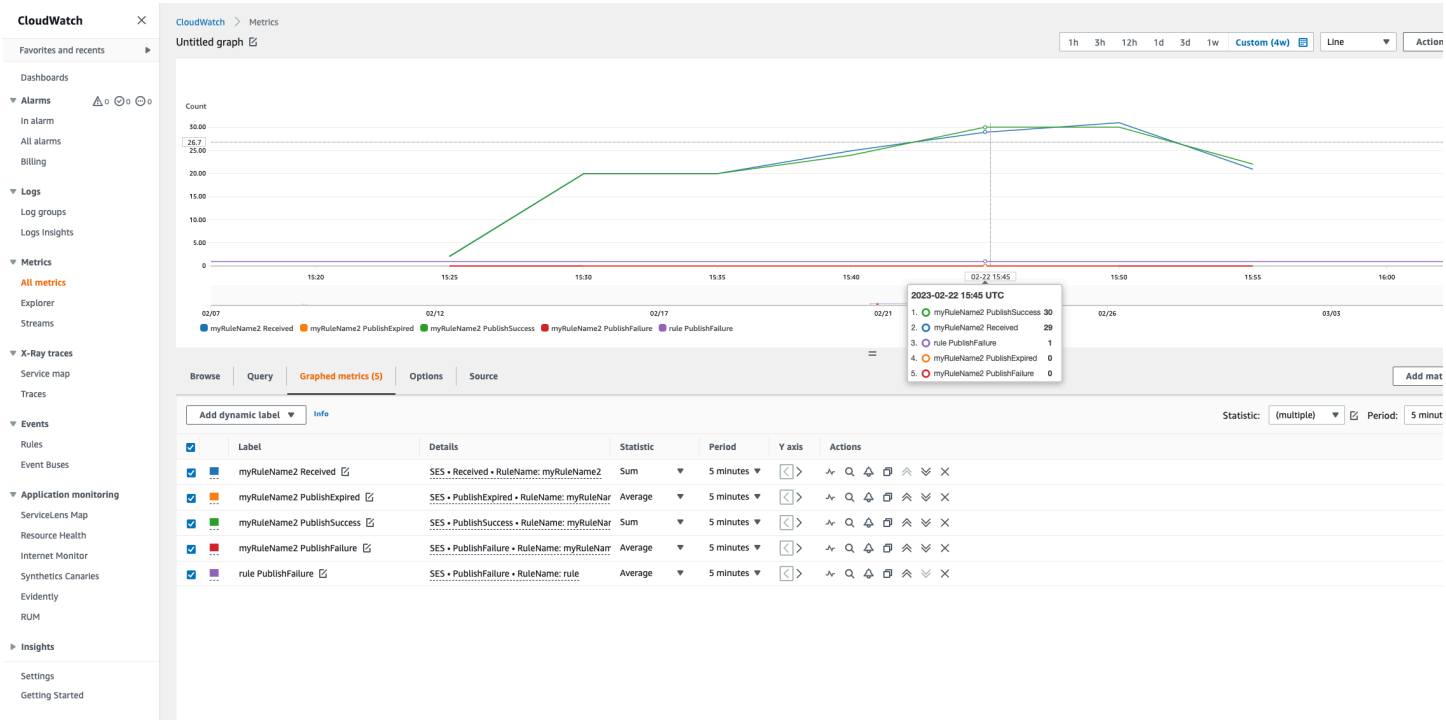
Important

受信ルールセットを修正するために行う変更は、更新後に Amazon SES が受信する E メールにのみ適用されます。E メールの評価は常に、その E メールが受信された時点で有効な受信ルールセットに対して行われます。

CloudWatch コンソールに表示される SES 受信ルールセットのメトリクス。



CloudWatch コンソールに表示される SES 受信ルールのメトリクス。



Amazon SES の検証済みID

Amazon SES では、検証済みID は E メールの送受信に使用する ドメインまたは E メールアドレスです。Amazon SES を使用して E メールを送信する前に、「From」、「Source」、「Sender」、または「Return-Path」のアドレスとして使用する各 ID を作成し、検証する必要があります。Amazon SESでIDを検証することにより、IDを所有していることを確認し、不正使用を防止します。

アカウントがまだ Amazon SES サンドボックス内にある場合は、Amazon SES メールボックスシミュレーター から提供されたテストインボックスへ送信するメールを除いて、すべてのE メール予定送信先のアドレスも検証する必要があります。詳細については、「the section called “手動でメールボックスシミュレーターを使用する”」を参照してください。

Amazon SES コンソールまたは Amazon SES API を使用して、ID を作成することができます。ID 検証プロセスは、作成する ID のタイプによって異なります。

> **Tip**
>
> SES を初めて使用する場合は、使用開始ウィザードを使用して最初の ID (E メールアドレスまたはドメイン) を作成および検証できます。

内容

Amazon での ID の作成と検証 SES

Amazon ではSES、ドメインレベルで ID を作成することも、E メールアドレス ID を作成することもできます。これらの ID タイプは相互に排他的ではありません。ほとんどの場合、ドメイン ID を作成することで、特定の E メールアドレスにカスタム構成を適用しない限り、個々の E メールアドレス ID の作成と検証を行う必要がなくなります。ドメインを作成してそのドメインに基づいて E メールアドレスを利用する場合も、個別の E メールアドレスを作成する場合もメリットがあります。選択する方法は、以下で説明するように、特定のニーズによって異なります。

E メールアドレス ID の作成と検証は、 で開始する最も速い方法ですがSES、ドメインレベルで ID を検証する利点があります。E メールアドレス ID を検証する場合は、その E メールアドレスのみでしかメールを送信できませんが、ドメイン ID を検証する場合、個別に検証することなく、検証済みドメインの任意のサブドメインまたは E メールアドレスから E メールを送信できます。例えば、example.com というドメイン ID を作成して検証する場合、a.example.com、a.b.example.com などの個別のサブドメイン ID や、user@example.com、user@a.example.com などの個別の E メールアドレス ID を作成する必要はありません。

ただし、ドメインから継承された検証を使用している E メールアドレス ID は、単純な E メール送信に限定されることに注意してください。より高度な送信を行う場合は、E メールアドレス ID として明示的に検証する必要もあります。高度な送信には、設定セットでの E メールアドレスの使用、委任送信のポリシー承認、およびドメイン設定を上書きする設定が含まれます。

上記の検証の引継ぎと E メール送信機能を明確にするために、次の表では、ドメイン/E メールアドレス検証の各組み合わせを分類し、それぞれの継承、送信レベル、および表示ステータスを一覧化しています。

	ドメインの検証のみ	E メールアドレスの検証のみ	ドメインと E メールアドレスの両方の検証
継承レベル	サブドメインと E メールアドレスは、親ドメインからの検証を継承します。	E メールアドレスは明示的に検証されます。	• サブドメインは親ドメインから検証を継承します。 • E メールアドレスは明示的に検証されます。
送信レベル	E メールアドレスは単純な E メール送信に限定されます。	E メールアドレスは高度な送信*に使用できます。	E メールアドレスは高度な送信*に使用できます。
表示ステータス	コンソール/APIステータス： • ドメイン/サブドメイン = 検証済み	コンソール/APIステータス： • E メールアドレス = 検証済み	コンソール/APIステータス： • ドメイン/サブドメイン = 検証済み

	ドメインの検証のみ	E メールアドレスの検証のみ	ドメインと E メールアドレスの両方の検証
	• E メールアドレス = 未検証。		• E メールアドレス = 検証済み。

*高度な送信には、設定セットでの E メールアドレスの使用、委任送信のポリシー承認、およびドメイン設定を上書きする設定が含まれます。

複数の で同じドメインまたは E メールアドレスから E メールを送信するには AWS リージョン、リージョンごとに個別の ID を作成して検証する必要があります。各地域で 10,000 個までの ID を検証することができます。

ドメインと E メールアドレス ID を作成および検証するときは、次の点を考慮します。

- 個別に検証することなく、検証済みドメインの任意のサブドメインまたは E メールアドレスから E メールを送信できます。たとえば、example.com の ID を作成して検証する場合、a.example.com、a.b.example.com、user@example.com、user@a.example.com などの個別の ID を作成する必要はありません。
- RFC 1034 で指定されているように、各DNSラベルには最大 63 文字を含めることができ、ドメイン名全体が合計 255 文字を超えることはできません。
- ルートドメインを共有するドメイン、サブドメイン、および E メールアドレスを検証する場合、ID 設定 (フィードバック通知など) は検証した最も細かいレベルで適用されます。
  - 検証済み E メールアドレスID設定は、検証済みドメインのID設定をオーバーライドします。
  - 検証済みサブドメインID設定は、検証済みドメインのID設定をオーバーライドします。より低いレベルのサブドメイン設定は、より高いレベルのサブドメイン設定をオーバーライドします。

    たとえば、user@a.b.example.com、a.b.example.com、b.example.com、および example.com を検証するとします。これらは、以下のシナリオで使用される検証済みのアイデンティティ設定です。

    - user@example.com (特別には検証されないEメールアドレス) から送信される E メールは、example.com の設定を使用します。
    - user@a.b.example.com (特別に検証されるEメールアドレス) から送信される E メールは、user@a.b.example.com の設定を使用します。

  - user@b.example.com (特別には検証されないEメールアドレス) から送信される E メールは、b.example.com の設定を使用します。
- 追加の検証手順を実行せずに、検証済みの E メールアドレスにラベルを追加することができます。E メールアドレスにラベルを追加するには、アカウント名と「at」記号 (@) の間にプラス記号 (+) を付け、続いてテキストラベルを付けます。例えば、sender@example.com をすでに検証している場合は、sender+myLabel@example.comを E メールの「From」または「Return-Path」アドレスとして使用できます。この機能を使用して、可変エンベロープリターンパス () を実装できますVERP。その後、 VERPを使用して、メーリングリストから配信不能の E メールアドレスを検出して削除できます。
- ドメイン名では大文字と小文字は区別されません。example.com を検証する場合は、EXAMPLE.comから送信することもできます。
- E メールアドレスでは、大文字と小文字が区別されます。sender@EXAMPLE.comを検証する場合、sender@example.com も検証しない限り、sender@example.com から E メールを送信することはできません。
- 各 で AWS リージョン、最大 10,000 個の ID (任意の組み合わせでドメインと E メールアドレス) を検証できます。

> **Tip**
>
> を初めて使用する場合はSES、開始方法ウィザードを使用して、最初の ID (E メールアドレスまたはドメイン) を作成して検証できます。

内容

ドメイン ID の作成

ドメイン ID の作成には、 DKIMベースの検証の設定が含まれます。 DomainKeys識別メール (DKIM) は、Amazon がドメインの所有権を検証SESするために使用する E メール認証方法であり、受信メールサーバーは E メールの信頼性を検証するために使用します。Easy DKIMまたは Bring-Your-Own DKIM (BYODKIM) のいずれかDKIMを使用して を設定することを選択できます。また、選択に応じて、プライベートキーの署名キーの長さを次のように設定する必要があります。

- Easy DKIM - Amazon のSESデフォルトである 2048 ビットを受け入れるか、1024 ビットを選択して上書きします。
- BYODKIM - プライベートキーの長さは、1024 ビット以上 2048 ビット以下である必要があります。

DKIM 署名キーの長さとその変更方法the section called “DKIM 署名キーの長さ”の詳細については、「」を参照してください。

次の手順は、Amazon SESコンソールを使用してドメイン ID を作成する方法を示しています。

- ドメインを既に作成していて、それを検証するだけの場合は、このページの「the section called “ドメイン ID の検証”」の手順に進んでください。

ドメイン ID の作成方法

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [設定] で、[ID] を選択します。
3. IDの作成を選択します。
4. ID の詳細で、作成するIDのタイプでドメインを選択します。ドメイン検証プロセスを完了するには、ドメインDNSの設定にアクセスできる必要があります。
5. ドメインまたはサブドメインの名前をドメインフィールドに入力します。

> **Tip**
>
> ドメインが www.example.com の場合は、ドメインとして「example.com」と入力します。「www.」の部分を含めないでください。含めてしまうと、ドメイン検証プロセスが失敗します。

6. (オプション)デフォルト設定セットを割り当てるを選択する場合、チェックボックスをオンにします。

   1. デフォルト設定セットで、ID に割り当てる既存の設定セットを選択します。設定セットを作成していない場合は、「設定セット」を参照してください。

      > **Note**
      >
      > Amazon は、送信時に他のセットが指定されていない場合にのみ、割り当てられた設定セットをSESデフォルトにします。設定セットが指定されている場合、Amazon はデフォルトのセットの代わりに指定されたセットSESを適用します。

7. （オプション) カスタムMAILFROMドメインを使用する場合は、チェックボックスをオンにして次のステップを完了します。詳細については、「the section called “カスタムMAILFROMドメインの使用”」を参照してください。

   1. MAIL FROM ドメインには、ドメインとして使用するサブMAILFROMドメインを入力します。これは、検証するドメインIDのサブドメインでなければなりません。MAIL FROM ドメインは、E メールを送信するドメインにすることはできません。
   2. MX 障害時の動作 では、送信時に必要な MX レコードが見つからない場合に Amazon がSES実行するアクションを指定します。以下のオプションのいずれかを選択します。
      - デフォルトMAILFROMドメインを使用する - カスタムMAILFROMドメインの MX レコードが正しく設定されていない場合、Amazon SESは amazonses.com のサブドメインを使用します。サブドメインは、Amazon AWS リージョン を使用する によって異なりますSES。
      - メッセージを拒否する - カスタムMAILFROMドメインの MX レコードが正しく設定されていない場合、Amazon SESは`MailFromDomainNotVerified`エラーを返します。このオプションを選択した場合、このドメインから送信しようとしたEメールは自動的に拒否されます。
   3. Route53 へのDNSレコードの発行では、ドメインが Amazon Route 53 を介してホストされている場合、Enabled をオンのままにして、作成時に が関連する TXTおよび MX レコードSESを発行するように選択できます。後でこれらのレコードを発行する場合は、[Enabled] (有効) のチェックボックスをオフにします。(後で ID を編集して、Route 53 にレコードを発行する場合は、the section called “コンソールを使用して ID を編集する” をご覧ください。)

8. （オプション) カスタマイズされた DKIMベースの検証を設定するには、2048 ビットの歌う長さの Easy DKIM を使用するSESデフォルト設定を除き、「ドメインの検証」で、詳細DKIM設定を展開し、設定DKIMする のタイプを選択します。

a. Easy DKIM：

   i. ID タイプで、Easy DKIMを選択します。

   ii. DKIM 署名キーの長さフィールドで、RSA_2048_BIT または RSA_1024_BITを選択します。

   iii. Route53 へのDNSレコードの発行では、ドメインが Amazon Route 53 を介してホストされている場合、Enabled をオンのままにして、作成時に が関連するCNAMEレコードSESを発行するように選択できます。後でこれらのレコードを発行する場合は、[Enabled] (有効) のチェックボックスをオフにします。(後で ID を編集して、Route 53 にレコードを発行する場合は、the section called “コンソールを使用して ID を編集する” をご覧ください。)

b. Deterministic Easy DKIM (DEED) ：

> **Tip**
>
> この形式の DKIMは、グローバル (レプリカ) ID を作成する場合に使用されます。DEED は、親リージョンから同じ名前の既存の ID を簡単にDKIMセットアップし、追加のDNSセットアップを必要とせずに新しい ID に署名します。詳細については、「DEED」を参照してください。

   i. ID タイプで、決定論的簡易 DKIMを選択します。

   ii. 親リージョンドロップダウンメニューから、グローバル (レプリカ) ID に入力したのと同じ名前の Easy DKIM署名付き ID が存在する親リージョンを選択します。(レプリカリージョンは、デフォルトでSESコンソールにサインインしたリージョンになります) 。

c. DKIM 認証トークン (BYODKIM) を指定します。

   i. パブリックキーとプライベートキーのペアが既に生成されており、DNSホストプロバイダーにパブリックキーが追加されていることを確認します。詳細については、「the section called “BYODKIM - 自分のDKIM を使用する”」を参照してください。

   ii. ID タイプで、DKIM認証トークンの提供 (BYODKIM) を選択します。

   iii. プライベートキーには、パブリックキーとプライベートのキーのペアから生成されたプライベートキーを貼り付けます。プライベートキーは、少なくとも 1024 ビットのRSA

暗号化と最大 2048 ビットの暗号化を使用し、base64 （PEM） エンコードを使用してエンコードする必要があります。

> **Note**
>
> 生成されたプライベートキーの最初と最後の行 (それぞれ `-----BEGIN PRIVATE KEY-----` と `-----END PRIVATE KEY-----`) を削除する必要があります。さらに、生成されたプライベートキーの改行を削除する必要があります。結果の値は、スペースや改行を含まない文字列になります。

iv. Selector name に、ドメインDNSの設定で指定するセレクタの名前を入力します。

9. DKIM 署名フィールドで有効ボックスがオンになっていることを確認します。
10. (オプション) 1 つ以上のタグをドメインIDに追加するには、タグキーとキーのオプションの値を指定します。
    1. 新しいタグを追加を選択し、キーを入力します。オプションで、タグに値を追加できます。
    2. 追加のタグが 50 を超えないように繰り返すか、削除を選択し、タグを削除します。
11. IDの作成を選択します。

でドメイン ID を作成して設定したのでDKIM、DNSプロバイダーとの検証プロセスを完了する必要があります。次に、ID をDKIM設定したタイプのDNS認証手順に進んでthe section called “ドメイン ID の検証”ください。

> **Note**

DNS プロバイダーとのDKIMドメイン ID の検証

で設定されたドメイン ID を作成したらDKIM、DKIM選択した のタイプに対応する認証手順に従って、DNSプロバイダーとの検証プロセスを完了する必要があります。

ドメイン ID をまだ作成していない場合は、「the section called “ドメイン ID の作成”」を参照してください。

> **Note**
>
> - ドメイン ID を検証するには、ドメインDNSの設定にアクセスする必要があります。これらの設定への変更が反映されるまでに最大 72 時間かかる場合があります。
> - Deterministic Easy () を使用してグローバル DKIM (レプリカ) ID を作成した場合DEED、追加のDNSセットアップは必要ありません。このステップはスキップできます。詳細については、「DEED」を参照してください。

DNS プロバイダーでDKIMドメイン ID を検証するには

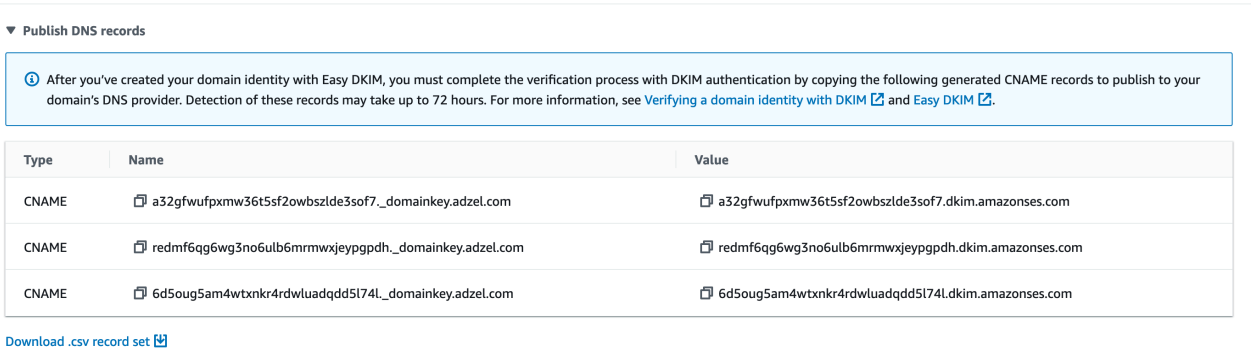
1. 検証済み ID テーブルで、検証するドメインを選択します。
2. ID の詳細ページの認証タブで、DNSレコードの発行を展開します。
3. Easy または DKIMでドメインDKIMを設定したフレーバーに応じてBYODKIM、それぞれの手順に従ってください。

   Easy DKIM

   Easy で設定されたドメインを検証するには DKIM

   1. Publish DNS records テーブルから、このセクションに表示される 3 つのCNAMEレコードをコピーしてDNS、プロバイダーに発行 (追加) します。または、[レコードセットを CSV としてダウンロード] を選択してコンピューターにこのレコードのコピーを保存することもできます。

      次の図は、DNSプロバイダーに発行するCNAMEレコードの例を示しています。

   2. DNS ホストプロバイダーそれぞれのドメインDNS設定にCNAMEレコードを追加します。

- すべてのDNSホストプロバイダー (Route 53 を除く) – ドメインの DNSまたはウェブホスティングプロバイダーにログインし、以前にコピーまたは保存した値を含むCNAMEレコードを追加します。DNS レコードを更新する手順はプロバイダーによって異なります。以下の手順に従って、DNS/Hosting provider の表を参照してください。

  > **Note**
  >
  > 少数のDNSプロバイダーでは、レコード名にアンダースコア (_) を含めることはできません。ただし、DKIMレコード名にはアンダースコアが必要です。DNS プロバイダーがレコード名にアンダースコアを入力できない場合は、プロバイダーのカスタマーサポートチームにお問い合わせください。

- DNS ホストプロバイダーとしての Route 53 – を使用して E メールを送信するときに使用するのと同じアカウントで Route 53 を使用しSES、ドメインが登録されている場合、作成時に SESがドメインの公開を有効にしている場合、 SES は自動的にドメインDNSの設定を更新します。そうでない場合は、作成後にボタンをクリックするだけで簡単に Route 53 に発行できます。「the section called “コンソールを使用して ID を編集する”」を参照してください。DNS 設定が自動的に更新されない場合、または を使用して E メールを送信するときに使用するアカウントと同じではないCNAMEレコードを Route 53 に追加する場合はSES、「レコードの編集」の手順を完了します。
- DNS プロバイダーが不明な場合は、システム管理者に問い合わせてください。

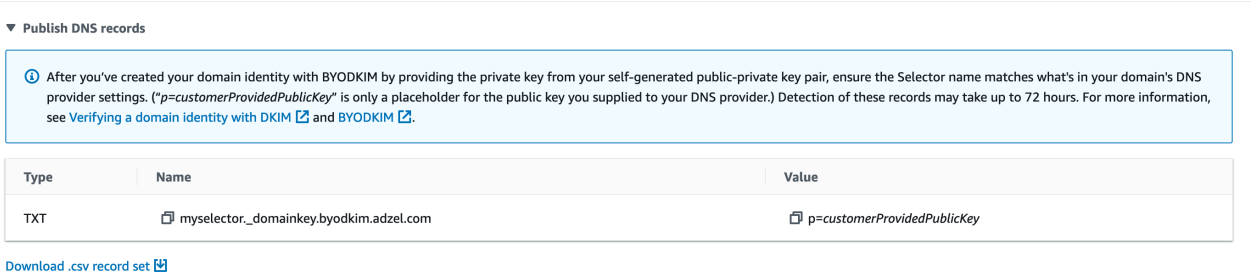
BYODKIM

で設定されたドメインを検証するには BYODKIM

1. 要約するために、 でドメインを作成したときBYODKIM、または で既存のドメインを設定したときにBYODKIM、コンソールの事前DKIM設定ページのそれぞれのフィールドにプライベートキー (自己生成パブリック/プライベートキーペアから) SES とセレクタ名のプレフィックスを追加しました。次に、DNSホストプロバイダーの次のレコードを更新して、検証プロセスを完了する必要があります。
2. DNS レコードの発行 テーブルから、発行 (追加) する名前列に表示されるセレクタ名レコードをDNSプロバイダーにコピーします。または、[Download .csv record set] (.csv

のレコードセットをダウンロード) を選択してコンピュータにこのレコードのコピーを保存することもできます。

次の図は、DNSプロバイダーに発行するセレクタ名レコードの例を示しています。



3. ドメインの DNSまたはウェブホスティングプロバイダーにログインし、以前にコピーまたは保存したセレクタ名レコードを追加します。DNS レコードを更新する手順はプロバイダーによって異なります。以下の手順に従って、DNS/Hosting provider の表を参照してください。

> **Note**
>
> 少数のDNSプロバイダーでは、レコード名にアンダースコア (_) を含めることはできません。ただし、DKIMレコード名にはアンダースコアが必要です。DNS プロバイダーがレコード名にアンダースコアを入力できない場合は、プロバイダーのカスタマーサポートチームにお問い合わせください。

4. まだ行っていない場合は、自己生成のパブリック/プライベートキーペアの のパブリックキーをドメインの DNSまたはウェブホスティングプロバイダーに追加してください。

Publish records DNSテーブルでは、Value 列に表示されるパブリックキーレコードには、コンピュータに保存またはDNSプロバイダーに提供されたパブリックキー値のプレースホルダーとして「p=customerProvidedPublicKey」のみが表示されることに注意してください。

> **Note**
>
> パブリックキーをDNSプロバイダーに公開 (追加) するときは、次のようにフォーマットする必要があります。
>
> - 生成されたパブリックキーの最初と最後の行 (それぞれ `-----BEGIN PUBLIC KEY-----` と `-----END PUBLIC KEY-----`) を削除する必要があ

ります。さらに、生成されたパブリックキーの改行を削除する必要があります。結果の値は、スペースや改行を含まない文字列になります。

- Publish records テーブルの Value 列に示すように、p=プレフィックスを含める必要があります。 DNS

4. DNS 設定の変更が反映されるまでに最大 72 時間かかる場合があります。Amazon がドメイン DNSの設定に必要なすべてのDKIMレコードSESを検出すると、検証プロセスは完了です。ドメインDKIMの設定は成功と表示され、ID ステータスは検証済みと表示されます。
5. カスタムMAILFROMドメインを設定および検証する場合は、「」の手順に従いますカスタムMAILFROMドメインの設定。

次の表には、広く使用されているいくつかのDNSプロバイダーのドキュメントへのリンクが含まれています。このリストは網羅的なものではなく、承認を意味するものでもありません。同様に、DNSプロバイダーがリストされていない場合、Amazon でドメインを使用できないことを意味するものではありませんSES。

DNS/ホスティングプロバイダー	ドキュメントのリンク
GoDaddy	CNAME レコードを追加する (外部リンク)
DreamHost	カスタムDNSレコードを追加する方法 (外部リンク)
Cloudflare	Cloudflare でのDNSレコードの管理 (外部リンク)
HostGator	HostGator/ を使用してDNSレコードを管理するeNom (外部リンク)
Namecheap	ドメインのTXT/SPF/DKIM/DMARCレコードを追加する方法 (外部リンク)
Names.co.uk	ドメインの変更 DNS 設定 (外部リンク)
Wix	Wix アカウントのCNAMEレコードの追加または更新 (外部リンク)

ドメイン検証のトラブルシューティング

上記のステップを完了してから 72 時間経過してもドメインが検証されない場合は、以下を確認してください。

- レコードの値をDNS正しいフィールドに入力していることを確認してください。一部のDNSプロバイダーは、名前/ホストフィールドをホストまたはホスト名と呼んでいます。また、一部のプロバイダーは [Record value] フィールドをポイント先 または結果と呼んでいます。
- プロバイダーが、DNSレコードに入力した名前/ホスト値にドメイン名を自動的に追加していないことを確認してください。プロバイダーによっては、ドメイン名が追加され、そのことが通知されない場合もあります。プロバイダーがドメイン名を名前/ホスト値に追加した場合は、この値の末尾からドメイン名を削除します。DNS レコードの値の末尾にピリオドを追加することもできます。このピリオドは、ドメイン名が完全に修飾されたことをプロバイダーに示します。
- 各DNSレコードの名前/ホスト値にはアンダースコア文字 (_) が必要です。プロバイダーがDNSレコード名にアンダースコアを許可していない場合は、プロバイダーのカスタマーサポート部門に連絡してサポートを依頼してください。
- ドメインDNSの設定に追加する必要がある検証レコードは、それぞれ異なります AWS リージョン。ドメインを使用して複数の から E メールを送信する場合は AWS リージョン、それらのリージョンごとに個別のドメイン ID を作成して検証する必要があります。

Eメールアドレス ID の作成

Amazon SESコンソールを使用して E メールアドレス ID を作成するには、次の手順を実行します。

E メールアドレスの ID を作成するには (コンソール)

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの設定で、検証済み ID を選択します。
3. IDの作成を選択します。
4. ID の詳細 で、作成するIDタイプとしてE メールアドレスを選択します。
5. [E メールアドレス] に、使用する E メールアドレスを入力します。E メールアドレスは、アクセスでき、メールを受信できるアドレスでなければなりません。
6. (オプション)デフォルト設定セットを割り当てるを選択する場合、チェックボックスをオンにします。

1. デフォルト設定セットで、ID に割り当てる既存の設定セットを選択します。設定セットを作成していない場合は、「設定セット」を参照してください。

> **Note**
>
> Amazon は、送信時に他のセットが指定されていない場合にのみ、割り当てられた設定セットをSESデフォルトにします。設定セットが指定されている場合、Amazon はデフォルトのセットの代わりに指定されたセットSESを適用します。

7. (オプション) 1 つ以上のタグをドメインIDに追加するには、タグキーとキーのオプションの値を指定します。

   1. 新しいタグを追加を選択し、キーを入力します。オプションで、タグに値を追加できます。
   2. 追加のタグが 50 を超えないように繰り返すか、削除を選択し、タグを削除します。

8. メールアドレス ID を作成するには、[Create identity] (ID を作成) を選択します。作成後、確認メールが 5 分以内に届きます。次のステップは、次のセクションの確認手順に従って、Eメールアドレスを確認することです。

> **Note**
>
> 検証しようとしている E メールアドレスに送信されるメッセージをカスタマイズすることができます。詳細については、「the section called “カスタム検証 E メールテンプレートの使用”」を参照してください。

これで E メールアドレス ID を作成したので、検証プロセスを完了する必要があります。「the section called “E メールアドレス ID の検証”」に進みます。

E メールアドレス ID の検証

E メールアドレス ID を作成したら、検証プロセスを完了する必要があります。

E メールアドレス ID をまだ作成していない場合は、「the section called “Eメールアドレス ID の作成”」を参照してください。

E メールアドレス ID を検証するには

1. ID の作成に使用した E メールアドレスの受信トレイを確認し、 no-reply-aws@amazon.com からの E メールを探します。
2. E メールを開き、E メールのリンクをクリックして、E メールアドレスの検証プロセスを完了します。完了したら、ID の状態が検証済みに更新されます。

E メールアドレス検証のトラブルシューティング

ID の作成から 5 分以内に確認メールが届かなかった場合は、次のトラブルシューティング手順を試してください。

- アドレスを正しく入力されていることを確認します。
- 確認しようとしている E メールアドレスが E メールを受信できることを確認してください。これは、確認するアドレスに別の E メールアドレスからテストメールを送信することでテストできます。
- 迷惑 E メールフォルダを確認します。
- 検証 E メールのリンクは 24 時間後に期限切れになります。新しい確認メールを送信するには、ID の詳細ページの上部にある再送信を選択します。

ID を作成して検証し、同時にデフォルトの設定セットを割り当てる

Amazon v2 の CreateEmailIdentityオペレーションを使用して、新しい E SES API メール ID を作成し、そのデフォルト設定セットを同時に設定できます。

> **Note**
>
> このセクションの手順を完了する前に、まず AWS CLIをインストールして設定する必要があります。詳細については、AWS Command Line Interface ユーザーガイドをご参照ください。

を使用してデフォルト設定セットを設定するには AWS CLI

- コマンドラインで、次のコマンドを入力して CreateEmailIdentityオペレーションを使用します。

```
aws sesv2 create-email-identity --email-identity ADDRESS-OR-DOMAIN --configuration-set-
name CONFIG-SET
```

上記のコマンドで、 を、検証する E メール ID *ADDRESS-OR-DOMAIN*に置き換えます。を、ID のデフォルト設定セットとして設定する設定セットの名前*CONFIG-SET*に置き換えます。

コマンドが正常に実行された場合、何の出力もなく終了します。

E メールアドレスを検証するには

1. 受信箱で、検証するメールアドレスをチェックします。「Amazon Web Services - Email Address Verification Request in region 」という件名のメッセージが表示されます。ここで*RegionName*、*RegionName*は、E メールアドレスを検証 AWS リージョン しようとした の名前です。

   メッセージを開き、メッセージ内のリンクをクリックします。

   > **Note**
   >
   > 検証メッセージ内のリンクは、メッセージが送信されてから 24 時間で期限切れになります。確認メールを受信してから 24 時間が経過した場合は、手順 1 ～ 5 を繰り返して、有効なリンクが設定された確認メールを受信してください。

2. Amazon SESコンソールの ID 管理で、E メールアドレスを選択します。E メールアドレスのリストで、検証する E メールアドレスを探します。E メールアドレスが検証された場合は、[Status] 列の値が 「verified」になります。

ドメインを検証するには

上記のコマンドライン手順で `--email-identity` パラメータにドメイン名を入力した場合、詳細については「ドメイン ID の検証」を参照してください。

カスタム検証 E メールテンプレートの使用

E メールアドレスを検証しようとすると、Amazon は次の図に示す例のような E メールをそのアドレスSESに送信します。

Dear Amazon Web Services Customer,

We have received a request to authorize this email address for use with Amazon SES and Amazon Pinpoint in region US West (Oregon). If you requested this verification, please go to the following URL to confirm that you are authorized to use this email address:

https://email-verification.us-west-2.amazonaws.com/?AWSAccessKeyId=AKIADQKE4EXAMPLE&Context=10987654321&Identity.IdentityName=recipient%40example.com&Identity.IdentityType=EmailAddress&Namespace=Bacon&Operation=ConfirmVerification&Signature=TJDufFhYYK1fSHCSBq4cjbodBQq%2FnyyZgzjqZ%2BXsDYEXAMPLE&SignatureMethod=HmacSHA256&SignatureVersion=2&Timestamp=2017-12-06T19%3A53%3A12.311Z

Your request will not be processed unless you confirm the address using this URL. This link expires 24 hours after your original verification request.

If you did NOT request to verify this email address, do not click on the link. Please note that many times, the situation isn't a phishing attempt, but either a misunderstanding of how to use our service, or someone setting up email-sending capabilities on your behalf as part of a legitimate service, but without having fully communicated the procedure first. If you are still concerned, please forward this notification to aws-email-domain-verification@amazon.com and let us know in the forward that you did not request the verification.

To learn more about sending email from Amazon Web Services, please refer to the Amazon SES Developer Guide at http://docs.aws.amazon.com/ses/latest/DeveloperGuide/Welcome.html and Amazon Pinpoint Developer Guide at http://docs.aws.amazon.com/pinpoint/latest/userguide/welcome.html.

Sincerely,

The Amazon Web Services Team.

複数の Amazon のSESお客様は、Amazon を通じて E メールを送信するアプリケーション (E メールマーケティングスイートやチケット発行システムなど) を構築SESしています。これらのアプリケーションのエンドユーザーにとって、E メール検証プロセスは混乱を招く可能性があります。検証 E メールは、アプリケーションのSESブランドではなく Amazon ブランドを使用しており、これらのエンドユーザーが Amazon SESを直接使用するようにサインアップすることはありません。

Amazon SESユースケースで、Amazon での使用のために E メールアドレスの検証を顧客に求める場合はSES、カスタマイズされた検証 E メールを作成できます。これらのカスタマイズされた E メールは、顧客の混乱を減らし、顧客が登録手続きを完了する率を上げるのに役立ちます。

> **Note**
>
> この機能を使用するには、Amazon SESアカウントがサンドボックスの外にある必要があります。詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。

このセクションのトピック:

- カスタム検証 E メールテンプレートの作成
- カスタム検証 E メールテンプレートの編集
- カスタムテンプレートを使用した検証 E メールの送信
- カスタム検証 E メールに関するよくある質問

カスタム検証 E メールテンプレートの作成

カスタム検証 E メールを作成するには、 CreateCustomVerificationEmailTemplateAPIオペレーションを使用します。このオペレーションでは、次の入力を使用します。

属性	説明
TemplateName	テンプレートの名前。指定する名前は一意である必要があります。
FromEmailAddress	確認 E メールが送信された E メールアドレス。指定するアドレスまたはドメインは、Amazon SESアカウントでの使用が検証されている必要があります。 **Note** FromEmailAddress 属性は表示名 (「差出人」名とも呼ばれます) をサポートしていません。
TemplateSubject	確認 E メールの件名。
TemplateContent	E メールの本文。E メール本文には を含めることができますが HTML、特定の制限があります。詳細については、「カスタム検証 E メールに関するよくある質問」を参照してください。
SuccessRedirectionURL	E メールアドレスが正常に検証された場合に、URLユーザーが送信される 。
FailureRedirectionURL	E メールアドレスが正常に検証されない場合に、URLユーザーが送信される 。

または を使用して AWS SDKs AWS CLI 、 CreateCustomVerificationEmailTemplateオペレーションでカスタム検証 E メールテンプレートを作成できます。の詳細については AWS SDKs、「アマゾン ウェブ サービスのツール」を参照してください。の詳細については AWS CLI、AWS 「 コマンドラインインターフェイス」を参照してください。

次のセクションでは、 AWS CLIを使用してカスタム確認 E メールを作成する手順について説明します。これらの手順は、 AWS CLIがインストールされ、設定されていることを前提としています。

のインストールと設定の詳細については AWS CLI、「AWS Command Line Interface ユーザーガイド」を参照してください。

> **Note**
>
> このセクションの手順を完了するには、 AWS CLIのバージョン 1.14.6 以降を使用する必要があります。最良の結果を得るには、 AWS CLIの最新バージョンにアップグレードします。の更新の詳細については AWS CLI、「AWS Command Line Interface ユーザーガイド」の「AWS Command Line Interfaceのインストール」を参照してください。

1. テキストエディタで新規ファイルを作成します。エディタに、以下の内容を貼り付けます。

```
{
  "TemplateName": "SampleTemplate",
  "FromEmailAddress": "sender@example.com",
  "TemplateSubject": "Please confirm your email address",
  "TemplateContent": "<html>
                      <head></head>
                      <body style='font-family:sans-serif;'>
                        <h1 style='text-align:center'>Ready to start sending
                        email with ProductName?</h1>
                        <p>We here at Example Corp are happy to have you on
                          board! There's just one last step to complete before
                          you can start sending email. Just click the following
                          link to verify your email address. Once we confirm that
                          you're really you, we'll give you some additional
                          information to help you get started with ProductName.</p>
                      </body>
                      </html>",
  "SuccessRedirectionURL": "https://www.example.com/verifysuccess",
  "FailureRedirectionURL": "https://www.example.com/verifyfailure"
}
```

> **Important**
>
> 上記の例を読みやすくするために、TemplateContent 属性に改行が含まれています。前述の例をテキストファイルに貼り付ける場合は、続行する前に改行を削除してください。

TemplateName、FromEmailAddress、TemplateSubject、TemplateContent、SuccessRedir
および FailureRedirectionURL の値を独自の値に置き換えます。

> **Note**
>
> FromEmailAddress パラメータで指定する E メールアドレス　は、検証される必要があるか、検証済みドメインのアドレスである必要があります。詳細については、「Amazon SES の検証済みID」を参照してください。

終了したら、customverificationemail.json としてファイルを保存します。

2. コマンドラインで次のコマンドを入力して、カスタム検証 E メールテンプレートを作成します。

```
aws sesv2 create-custom-verification-email-template --cli-input-json file://
customverificationemail.json
```

3. （オプション）次のコマンドを入力して、テンプレートが作成されたことを確認できます。

```
aws sesv2 list-custom-verification-email-templates
```

カスタム検証 E メールテンプレートの編集

UpdateCustomVerificationEmailTemplateオペレーションを使用して
カスタム確認 E メールテンプレートを編集できます。このオペレーションで
は、CreateCustomVerificationEmailTemplate オペレーション (つまり
TemplateName、FromEmailAddress、TemplateSubject、TemplateContent、SuccessRedirecti
および FailureRedirectionURL 属性) と同じ入力を受け入れます。ただ
し、UpdateCustomVerificationEmailTemplate オペレーションでは、これらの属性は必要で
はありません。既存のカスタム確認 E メールテンプレートの名前と同じ TemplateName 値を渡す
と、指定した属性がテンプレートに元々含まれていた属性を上書きします。

カスタムテンプレートを使用した検証 E メールの送信

少なくとも 1 つのカスタム検証 E メールテンプレートを作成したら、
SendCustomVerificationEmailAPIオペレーションを呼び出して顧客に送信できま

す。SendCustomVerificationEmail オペレーションは、 または AWS SDKsのいずれかを使用して呼び出すことができます AWS CLI。SendCustomVerificationEmail オペレーションでは、次の入力を使用します。

属性	説明
EmailAddress	確認されている E メールアドレス。
TemplateName	確認されている E メールアドレスに送信されるカスタム確認 E メールテンプレートの名前。
ConfigurationSetName	(オプション) 確認 E メールを送信するときに使用する設定セットの名前。

たとえば、顧客がアプリケーションのフォームを使用してサービスに登録するとします。顧客がフォームを完成させて送信すると、アプリケーションは SendCustomVerificationEmail オペレーションを呼び出し、顧客の E メールアドレスと使用するテンプレートの名前を渡します。

顧客は、作成したカスタマイズされた E メールテンプレートを使用する E メールを受信します。Amazon は、受信者に一意のリンクと簡単な免責事項SESを自動的に追加します。次の図は、カスタム検証 E メールテンプレートの作成で作成されたテンプレートを使用する確認 E メールのサンプルを示しています。

Ready to start sending email with ProductName?

We here at Example Corp are happy to have you on board! There's just one last step to complete before you can start sending email. Just click the following link to verify your email address. Once we confirm that you're really you, we'll give you some additional information to help you get started with ProductName.

https://email-verification.us-west-2.amazonaws.com/?AWSAccessKeyId=AKIADQKE4EXAMPLE&Context=10987654321&Identity.IdentityName=recipient%40example.com&Identity.IdentityType=EmailAddress&Namespace=Bacon&Operation=ConfirmVerification&Signature=TJDufFhYYK1fSHCSBq4cjbodBQq%2FnyyZgzjqZ%2BXsDYEXAMPLE&SignatureMethod=HmacSHA256&SignatureVersion=2&Timestamp=2017-12-06T19%3A53%3A12.311Z

If you did not request to verify this email address, please disregard this message. If you have any concerns, please forward this message to the following email address along with your questions or concerns.

カスタム検証 E メールに関するよくある質問

このセクションでは、カスタム確認 E メールテンプレート機能に関するよくある質問に対する回答を示します。

Q1. 作成できるカスタム確認 E メールテンプレートの数はいくつですか?

Amazon SESアカウントごとに最大 50 個のカスタム検証 E メールテンプレートを作成できます。

Q2. カスタム確認 E メールはどのように受信者に表示されますか?

カスタム確認 E メールには、テンプレートを作成したときに指定したコンテンツと、メールアドレスを確認するために受信者がクリックする必要があるリンクが含まれます。

Q3. カスタム確認 E メールをプレビューすることはできますか?

カスタム確認 E メールをプレビューするには、`SendCustomVerificationEmail` オペレーションを使用して確認 E メールを所有しているアドレスに送信します。検証リンクをクリックしない場合、Amazon SES は新しい ID を作成しません。確認リンクをクリックすると、`DeleteIdentity` オペレーションを使用して、オプションとして新しく作成された ID を削除することもできます。

Q4. カスタム確認 E メールテンプレートに画像を含めることはできますか?

base64 エンコーディングを使用して、テンプレートHTMLの にイメージを埋め込むことができます。この方法でイメージを埋め込むと、Amazon SESは自動的にイメージをアタッチメントに変換します。次のコマンドのいずれかを発行して、画像をコマンドラインでエンコードすることができます。

Linux, macOS, or Unix

```
base64 -i imagefile.png | tr -d '\n' > output.txt
```

Windows

```
certutil -encodehex -f imagefile.png output.txt 0x40000001
```

imagefile.png を、エンコードするファイルの名前に置き換えます。上記の両方のコマンドで、Base64 でエンコードされた画像は`output.txt`に保存されます。

テンプレートの に以下を含めることで、base64 でエンコードされたイメージHTMLを埋め込むことができます。 `<img src="`*base64EncodedImage*`"/>`

前述の例では、*png* をエンコードされた画像のファイルタイプ (jpg や gif など) に置き換え、*base64EncodedImage*をBase64 でエンコードされた画像 (つまり、前述のコマンドの `output.txt` 内容) に置き換えます。

Q5. カスタム検証 E メールテンプレートに含めることのできるコンテンツに制限はありますか?

カスタム確認 E メールテンプレートのサイズは 10 MB を超えることはできません。さらに、 を含むカスタム検証 E メールテンプレートHTMLは、次の表に示すタグと属性のみを使用できます。

HTML タグ	許可された属性
abbr	class, id, style, title
acronym	class, id, style, title
address	class, id, style, title
area	class, id, style, title
b	class, id, style, title
bdo	class, id, style, title
big	class, id, style, title
blockquote	cite, class, id, style, title
body	class, id, style, title
br	class, id, style, title
button	class, id, style, title
caption	class, id, style, title
center	class, id, style, title
cite	class, id, style, title
code	class, id, style, title
col	class, id, span, style, title, width

HTML タグ	許可された属性
colgroup	class, id, span, style, title, width
dd	class, id, style, title
del	class, id, style, title
dfn	class, id, style, title
dir	class, id, style, title
div	class, id, style, title
dl	class, id, style, title
dt	class, id, style, title
em	class, id, style, title
fieldset	class, id, style, title
font	class, id, style, title
form	class, id, style, title
h1	class, id, style, title
h2	class, id, style, title
h3	class, id, style, title
h4	class, id, style, title
h5	class, id, style, title
h6	class, id, style, title
head	class, id, style, title
hr	class, id, style, title

HTML タグ	許可された属性
html	class, id, style, title
i	class, id, style, title
img	align, alt, class, height, id, src, style, title, width
input	class, id, style, title
ins	class, id, style, title
kbd	class, id, style, title
label	class, id, style, title
legend	class, id, style, title
li	class, id, style, title
map	class, id, style, title
menu	class, id, style, title
ol	class, id, start, style, title, type
optgroup	class, id, style, title
option	class, id, style, title
p	class, id, style, title
pre	class, id, style, title
q	cite, class, id, style, title
s	class, id, style, title
samp	class, id, style, title

HTML タグ	許可された属性
select	class, id, style, title
small	class, id, style, title
span	class, id, style, title
strike	class, id, style, title
strong	class, id, style, title
sub	class, id, style, title
sup	class, id, style, title
table	class, id, style, summary, title, width
tbody	class, id, style, title
td	abbr, axis, class, colspan, id, rowspan, style, title, width
textarea	class, id, style, title
tfoot	class, id, style, title
th	abbr, axis, class, colspan, id, rowspan, scope, style, title, width
thead	class, id, style, title
tr	class, id, style, title
tt	class, id, style, title
u	class, id, style, title
ul	class, id, style, title, type

HTML タグ	許可された属性
var	class, id, style, title

> **Note**
> カスタム検証 E メールテンプレートには、コメントタグを含めることはできません。

Q6. 確認済み E メールアドレスをアカウントにいくつ持つことができますか?

Amazon SESアカウントは、各 AWS リージョンで最大 10,000 個の検証済み ID を持つことができます。Amazon ではSES、ID には検証済みドメインと E メールアドレスの両方が含まれます。

Q7. Amazon SESコンソールを使用してカスタム検証 E メールテンプレートを作成できますか ?

現在、Amazon を使用してカスタム検証 E SES メールを作成、編集、削除できるのは のみです API。

Q8. 顧客がカスタム確認 E メールを受け取ったときに発生するオープンイベントとクリックイベントを追跡できますか?

カスタム検証E メールには、オープンまたはクリックの追跡を含めることはできません。

Q9. カスタム検証E メールにカスタムヘッダーを含めることはできますか?

カスタム検証E メールにカスタムヘッダーを含めることはできません。

Q10. カスタム検証E メールの下部に表示されるテキストを削除できますか?

次のテキストは、すべてのカスタム検証E メールの末尾に自動的に追加され、削除することはできません。

このE メールアドレスの確認をリクエストしていない場合は、このメッセージを無視してください。

Q11. カスタム検証 E メールDKIMは署名されていますか ?

検証 E メールDKIMに署名するには、検証 E メールテンプレートの作成時に `FromEmailAddress` 属性で指定する E メールアドレスを、DKIM署名を生成するように設定する必要があります。ドメ

インと E メールアドレスの設定の詳細については、DKIM「」を参照してくださいthe section called “を使用した E メールの認証 DKIM”。

Q12. カスタム検証 E メールテンプレートAPIオペレーションが SDKまたは に表示されないのはなぜですかCLI？

SDK または でカスタム検証 E メールテンプレートオペレーションを使用できない場合は AWS CLI、古いバージョンの SDKまたは を使用している可能性がありますCLI。カスタム検証 E メールテンプレートオペレーションは、次の SDKsおよび で使用できますCLIs。

- のバージョン 1.14.6 以降 AWS Command Line Interface
- のバージョン 3.3.205.0 以降 AWS SDK for .NET
- for C++ のバージョン 1.3.20170531.19 以降 AWS SDK
- のバージョン 1.12.43 以降 AWS SDK for Go
- のバージョン 1.11.245 以降 AWS SDK for Java
- のバージョン 2.166.0 以降 AWS SDK for JavaScript
- のバージョン 3.45.2 以降 AWS SDK for PHP
- のバージョン 1.5.1 以降 AWS SDK for Python (Boto)
- `aws-sdk-ses` の AWS SDK for Ruby Gem のバージョン 1.5.0 以降

Q13. カスタム確認 E メールを送信するときに **`ProductionAccessNotGranted`** エラーが発生するのはなぜですか?

`ProductionAccessNotGranted` エラーは、アカウントがまだ Amazon SESサンドボックスにあることを示します。カスタム確認 E メールは、アカウントがサンドボックスから削除されている場合にのみ送信できます。詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。

Amazon SES の ID の管理

Amazon SES コンソールでは、各 AWS リージョン に作成した ID を表示したり、ID を開いて詳細設定を表示して編集したり、デフォルトの設定セットを関連付けたり、単数または複数の ID を削除したりできます。

> **Note**
>
> このセクションで説明する手順は、選択した AWS リージョン の ID にのみ適用されます。複数の地域で作成したIDを管理するには、各AWS リージョンの手順を参照してください。

内容

SES コンソールを使用して ID を表示する

Amazon SES コンソールを使用して、検証済みまたは検証待ちのドメイン ID や E メールアドレス ID を表示できます。また、検証が失敗したIDを表示することもできます。

ドメインとメールアドレスの ID を表示するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. コンソールで、地域セレクターを使用して、ID のリストを表示するAWS リージョンを選択します。

   > **Note**
   >
   > この手順では、選択したAWS リージョンのIDリストのみを表示します。

3. ナビゲーションペインの [設定] で、[検証済み ID] を選択します。[検証済みの ID] テーブルには、ドメイン ID と E メールアドレス ID の両方が表示されます。ステータス列には、ID が検証済みか、検証保留中か、検証プロセスに失敗したかが表示されます - すべての可能性のあるステータス値の定義は以下の通りです :

   - 検証済み — SES での送信について、アイデンティティが正常に検証されました。

- 失敗 — SES はお客様のアイデンティティを検証できませんでした。ドメインの場合は、SES が 72 時間以内に DNS レコードを検出できなかったことを意味します。E メールアドレスの場合は、E メールアドレスに送信された検証メールが 24 時間以内に確認されなかったことを意味します。
- 保留 — SES はまだアイデンティティを確認中です。
- 一時的な失敗 — 以前に検証されたドメインの場合、SES は検証に必要な DNS レコードを定期的にチェックします。ある時点で SES がレコードを検出できない場合、ステータスは一時的な失敗に変更されます。SES は 72 時間 DNS レコードを再チェックし、レコードを検出できない場合、ドメインのステータスは失敗に変更されます。レコードを検出できる場合、ドメインのステータスは検証済みに変更されます。
- 未開始 — 検証プロセスをまだ開始していません。

4. ID を検証ステータス別に並べ替えるには、ステータス列を選択します。
5. ID の詳細ページを表示するには、表示する ID を選択します。

SES コンソールを使用して ID を削除する

Amazon SES コンソールを使用して、選択した AWS リージョンのアカウントからドメインまたは E メールアドレス ID を削除できます。

ドメインまたはEメールアドレス ID を削除するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. コンソールで、地域セレクターを使用して、1 つまたは複数の ID を削除するには、AWS リージョンを選択します。
3. ナビゲーションペインの [設定] で、[検証済み ID] を選択します。

   [検証済みの ID] テーブルには、ドメイン ID とメールアドレス ID の両方が表示されます。
4. ID列で、削除する ID を選択します。複数のIDを削除するには、削除する各IDの横にあるチェックボックスをオンにします。
5. [削除] を選択します。

SES コンソールを使用して ID を編集する

Amazon SES コンソールを使用して、選択した AWS リージョン のアカウントにあるドメインまたはメールアドレス ID を編集できます。

ドメインまたはメールアドレス ID を編集するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. コンソールで、地域セレクターを使用して、 1つまたは複数のIDを編集するには、AWS リージョン を選択します。
3. ナビゲーションペインの [設定] で、[検証済み ID] を選択します。

   [検証済みの ID] テーブルには、ドメイン ID とメールアドレス ID の両方が表示されます。
4. ID 列で、編集する ID を選択します (チェックボックスを選択するのではなく、ID の名前を直接クリックします)。
5. ID の詳細ページで、編集するカテゴリが含まれているタブを選択します。
6. 選択したタブにある、カテゴリごとに分けられたコンテナのいずれかで、編集したい属性の[Edit] (編集) ボタンを選択して変更を行い、[Save changes] (変更の保存) を選択します。

   a. ドメイン ID が Amazon Route 53 によりホストされていて、まだ DNS レコードを発行していない場合に [Authentication] (認証) タブで属性を編集したいときは、[DomainKeys Identified Mail (DKIM)] (ドメインキーアイデンティファイドメール (DKIM)) コンテナ、または [Custom MAIL FROM domain] (カスタム MAIL FROM ドメイン) コンテナのいずれか、または両方に、[Publish DNS records to Route53] (DNS レコードを Route53 に発行する) ボタンが表示されています ([Edit] (編集) ボタンの隣)。

   > **Note**
   >
   > [Authentication] (認証) タブは、アカウントに検証済みドメインがある場合か、アカウント内の E メールアドレスが検証済みドメインを使用している場合にのみ表示されます。

   b. [Publish DNS records to Route53] (DNS レコードを Route53 に発行する) ボタンで DNS レコードを直接発行できます。ボタンをクリックすると確認バナーが表示され、[Publish DNS records to Route53] (DNS レコードを Route53 に発行する) ボタンはそれぞれのコンテナに表示されなくなります。

7. 編集する ID のそれぞれの属性について、手順 5 と 6 を繰り返します。

SES API を使用して、デフォルトの設定セットを使用するように ID を編集する

PutEmailIdentityConfigurationSetAttributes オペレーションを使用して、既存の電子メール ID からデフォルト設定セットを追加または削除できます。

> **Note**
>
> このセクションの手順を完了する前に、まず AWS CLI をインストールして設定する必要があります。詳細については、AWS Command Line Interface ユーザーガイドを参照してください。

AWS CLI を使用してデフォルト設定セットを追加するには

- コマンドラインで以下のコマンドを入力してPutEmailIdentityConfigurationSetAttributes オペレーションを使用します。

```
aws sesv2 put-email-identity-configuration-set-attributes --email-identity ADDRESS-OR-DOMAIN --configuration-set-name CONFIG-SET
```

上記のコマンドで、*ADDRESS-OR-DOMAIN* を、検証する E メール ID に置き換えます。*CONFIG-SET* を、ID のデフォルト設定セットとして設定する設定セット名に置き換えます。

コマンドが正常に実行された場合、何の出力もなく終了します。

AWS CLI を使用してデフォルト設定セットを削除するには

- コマンドラインで以下のコマンドを入力してPutEmailIdentityConfigurationSetAttributes オペレーションを使用します。

```
aws sesv2 put-email-identity-configuration-set-attributes --email-identity ADDRESS-OR-DOMAIN
```

上記のコマンドで、*ADDRESS-OR-DOMAIN* を、検証する E メール ID に置き換えます。

コマンドが正常に実行された場合、何の出力もなく終了します。

ID が使用するデフォルトの設定セットを SES API を使用して取得する

GetEmailIdentity オペレーションを使用して、電子メール ID のデフォルト設定セットを返します (該当する場合)。

> **Note**
>
> このセクションの手順を完了する前に、まず AWS CLI をインストールして設定する必要があります。詳細については、AWS Command Line Interface ユーザーガイドを参照してください。

AWS CLI を使用してデフォルト設定セットを返すには

- コマンドラインで以下のコマンドを入力して GetEmailIdentity オペレーションを使用します。

```
aws sesv2 get-email-identity --email-identity ADDRESS-OR-DOMAIN
```

上記のコマンドで、*ADDRESS-OR-DOMAIN* を、デフォルト設定セットが知りたい電子メールIDに置き換えます (存在する場合) 。

コマンドが正常に実行された場合、E メール ID の詳細を含む JSON オブジェクトが提供されます。

SES API を使用して ID で現在使用されているデフォルトの設定セットをオーバーライドする

SendEmail オペレーションを使用して、異なる設定セットで電子メールを送信できます。そうした場合、指定した設定セットが、ID のデフォルト設定セットをオーバーライドします。

> **Note**
>
> このセクションの手順を完了する前に、まず AWS CLI をインストールして設定する必要があります。詳細については、AWS Command Line Interface ユーザーガイドを参照してください。

AWS CLI を使用してデフォルト設定セットをオーバーライドするには

- コマンドラインで以下のコマンドを入力して、SendEmail オペレーションを使用します。

```
aws sesv2 send-email --destination file://DESTINATION-JSON --content file://CONTENT-JSON --from-email-address ADDRESS-OR-DOMAIN --configuration-set-name CONFIG-SET
```

上記のコマンドで、*DESTINATION-JSON* を宛先の JSON ファイルに置き換え、*CONTENT-JSON* をコンテンツ JSON ファイルに置き換え、*ADDRESS-OR-DOMAIN* を FROM メールアドレスに置き換え、*CONFIG-SET* を、ID のデフォルト設定セットではなく、使用する設定セットの名前に置き換えます。

コマンドが正常に実行された場合、MessageId を出力します。

Amazon SES での ID の設定

Amazon Simple Email Service (Amazon SES) は、Simple Mail Transfer Protocol (SMTP) を使用してEメールを送信します。SMTP 自体は認証を提供しません。実際の送信元を隠蔽したスパムの発信者から、他人を装って E メールメッセージが送信される可能性があります。スパムの発信者は、Eメールヘッダーを改ざんし、送信元 IP アドレスを偽装することにより、そのメールメッセージが本物であると受取人に思い込ませることができます。

メールトラフィックを転送するほとんどのISP は、E メールの正当性を評価するための対策を講じています。E メールが認証されているかどうかの確認は、ISP が講じているそうした対策の 1 つです。認証において送信者は、自分がその送信元アカウントの所有者であることを証明する必要があります。場合によっては、ISP は認証されない E メールの送信を拒否します。配信性能を最大限確保するために、E メールを認証することをお勧めします。

以降のセクションでは、ISP で使用される 2 つの認証メカニズム、Sender Policy Framework (SPF) とドメインキーアイデンティファイドメール (DKIM)、について説明するとともに、Amazon SES でそれらの標準を使用する方法について説明します。

- Eメールメッセージをその送信元のシステムにまでさかのぼって追跡するSPF については、「Amazon SES における SPF を使った E メールの認証」を参照してください。
- メールメッセージに署名し、自分のメッセージが本物であることと送信中に改ざんされていないことを ISP に証明するための標準規格であるDKIM は、「Amazon DKIMでの による E メールの認証 SES」を参照してください。

- SPF および DKIM に基づく DMARC (Domain-based Message Authentication, Reporting and Conformance) に準拠する方法については、「Amazon SES の DMARC 認証プロトコルへの準拠」を参照してください。

E メールの認証方法

Amazon Simple Email Service (Amazon SES) は、Simple Mail Transfer Protocol (SMTP) を使用して E メールを送信します。SMTP 自体は認証を提供しません。実際の送信元を隠蔽したスパムの発信者から、他人を装って E メールメッセージが送信される可能性があります。スパムの発信者は、E メールヘッダーを改ざんし、送信元 IP アドレスを偽装することにより、そのメールメッセージが本物であると受取人に思い込ませることができます。

メールトラフィックを転送するほとんどのISP は、E メールの正当性を評価するための対策を講じています。E メールが認証されているかどうかの確認は、ISP が講じているそうした対策の 1 つです。認証において送信者は、自分がその送信元アカウントの所有者であることを証明する必要があります。場合によっては、ISP は認証されない E メールの送信を拒否します。配信性能を最大限確保するために、E メールを認証することをお勧めします。

内容

Amazon DKIMでの による E メールの認証 SES

DomainKeys 識別メール (DKIM) は、特定のドメインからの E メールが実際にそのドメインの所有者によって承認されていることを確認するために設計された E メールセキュリティ標準です。パブリックキー暗号を使用して、プライベートキーで Eメールに署名します。その後、受信者サーバーはドメインの に公開されたパブリックキーを使用してDNS、E メールの一部が転送中に変更されていないことを確認できます。

DKIM 署名はオプションです。署名を使用して E メールに署名することで、 DKIM準拠の E メールプロバイダーの配信性能DKIMを向上させることができます。Amazon SES には、DKIM署名を使用してメッセージに署名するための 3 つのオプションがあります。

- Easy DKIM: はパブリックキーとプライベートキーのペアSESを生成し、そのアイデンティティから送信するすべてのメッセージDKIMに署名を自動的に追加します。「」を参照してくださいAmazon DKIMで簡単 SES。
- Deterministic Easy DKIM (DEED) : Easy を使用している親 ID としてDKIM署名属性を自動的に継承するレプリカ ID を作成 AWS リージョン することでDKIM、複数の にわたって一貫したDKIM署名を維持できます。「」を参照してくださいAmazon での確定的 Easy DKIM (DEED) の使用 SES。
- BYODKIM (Bring Your Own DKIM) : 独自のパブリック/プライベートキーペアを指定し、そのアイデンティティから送信するすべてのメッセージDKIMに署名SESを追加します。「」を参照してくださいAmazon SES で独自の DKIM 認証トークン (BYODKIM) を提供する。
- DKIM 署名を手動で追加する: を使用して送信する E メールに独自のDKIM署名を追加します。`SendRawEmail`API「」を参照してくださいAmazon SES 内で手動での DKIM 署名。

DKIM 署名キーの長さ

多くのDNSプロバイダーが 2048 DKIM ビットRSA暗号化を完全にサポートするようになったため、Amazon SESは 2048 もサポートし、E DKIM メールのより安全な認証を可能にするため、 APIまたはコンソールDKIMから Easy を設定するときにデフォルトのキー長として使用します。2048 ビットキーは、署名キーの長さが 1024 ビット以上 2048 ビット以下である Bring Your Own DKIM (BYODKIM) でも設定して使用できます。

セキュリティと E メールの配信性能のために、Easy で設定するとDKIM、1024 ビットと 2048 ビットのキー長のいずれかを使用することができ、2048 をまだサポートしていないDNSプロバイダーによって問題が発生した場合に 1024 に戻す柔軟性があります。新しい ID を作成すると、1024 DKIM を指定しない限り、デフォルトで 2048 で作成されます。

転送中の E メールの配信可能性を維持するために、DKIMキーの長さを変更できる頻度には制限があります。制限事項は次のとおりです。

- 既に設定されているキーの長さと同じ長さに切り替えることはできません。
- 24時間の間に複数回異なるキーの長さに切り替えることはできません (その期間で1024への最初のダウングレードでない限り) 。

E メールが転送中の場合、 DNSはパブリックキーを使用して E メールを認証します。したがって、キーを頻繁にまたは頻繁に変更すると、 は E メールをDKIM認証できないDNS可能性があります。以前のキーはすでに無効になっている可能性があるため、これらの制限によって E メールが保護されます。

DKIM に関する考慮事項

DKIM を使用して E メールを認証する場合、次のルールが適用されます。

- 設定する必要があるのは、「FromDKIM」アドレスで使用するドメインのみです。「Return-Path」または「Reply-to」アドレスで使用するドメインDKIMに を設定する必要はありません。
- Amazon SES は複数の AWS リージョンで利用できます。複数の AWS リージョンを使用して E メールを送信する場合は、それらの各リージョンでDKIMセットアッププロセスを完了して、すべての E メールが DKIM署名されていることを確認する必要があります。
- DKIM プロパティは親ドメインから継承されるため、DKIM認証を使用してドメインを検証すると、次のようになります。
  - DKIM 認証は、そのドメインのすべてのサブドメインにも適用されます。
    - DKIM サブドメインの設定は、サブドメインでDKIM認証を使用しない場合や、後で再度有効にする場合は、継承を無効にすることで親ドメインの設定を上書きできます。
  - DKIM 認証は、そのアドレスでDKIM検証済みドメインを参照する E メール ID から送信されたすべての E メールにも適用されます。
    - DKIM E メールアドレスの設定は、DKIM認証なしでメールを送信する場合に継承を無効にすることで、サブドメイン (該当する場合) と親ドメインの設定を上書きできます。また、後で再度有効にすることもできます。

継承されたDKIM署名プロパティについて

Easy DKIMまたは が使用されたかどうかにかかわらず、そのドメインが で設定されている場合 DKIM、E メールアドレス ID BYODKIMは親ドメインからDKIMその署名プロパティを継承することを理解することが重要です。したがって、E メールアドレス ID でDKIMの署名を無効化または有効にすると、次の重要な事実に基づいてドメインDKIMの署名プロパティが上書きされます。

- E メールアドレスが属するドメインDKIM用に を既に設定している場合は、E メールアドレス ID DKIMの署名を有効にする必要もありません。
  - ドメインDKIMに を設定すると、Amazon は親ドメインから継承されたDKIMプロパティを通じて、そのドメインのすべてのアドレスからのすべての E メールSESを自動的に認証します。
- DKIM 特定の E メールアドレス ID の設定は、アドレスが属する親ドメインまたはサブドメイン (該当する場合) の設定を自動的に上書きします。

E メールアドレス ID DKIMの署名プロパティは親ドメインから継承されるため、これらのプロパティの上書きを計画している場合は、以下の表で説明されているように、上書きの階層ルールに留意する必要があります。

親ドメインでDKIM署名が有効になっていない	親ドメインでDKIM署名が有効になっている
E メールアドレス ID でDKIM署名を有効にすることはできません。	E メールアドレス ID DKIMの署名を無効にすることができます。
	E メールアドレス ID でDKIM署名を再度有効にできます。

通常、DKIM署名を無効にすることは推奨されません。これは、送信者の評価が損なわれるリスクがあり、送信されたメールが迷惑メールやスパムフォルダに送られたり、ドメインがスプーフィングされたりするリスクが高まるためです。

ただし、特定のユースケースや、署名を永続的または一時的に無効にしたり、後で再度有効にしたりする必要があるビジネス上の決定が古くなる場合に、E メールアドレス ID のドメインで継承されたDKIM署名DKIMプロパティを上書きする機能があります。「the section called “E メールアドレスに DKIM 署名を上書きする”」を参照してください。

Amazon DKIMで簡単 SES

ドメイン ID DKIMに Easy を設定すると、Amazon はその ID から送信するすべての E メールに2048 ビットのDKIMキーSESを自動的に追加します。Amazon SESコンソールまたは DKIMを使用して Easy を設定できますAPI。

> **Note**
>
> Easy を設定するにはDKIM、ドメインDNSの設定を変更する必要があります。Route 53 をDNSプロバイダーとして使用すると、Amazon SES は適切なレコードを自動的に作成できます。別のDNSプロバイダーを使用している場合は、プロバイダーのドキュメントを参照して、ドメインのDNS設定を変更する方法の詳細を確認してください。

> **Warning**
>
> 現在 BYODKIMを有効にしていて、Easy に移行している場合はDKIM、Easy のセットアップ中、DKIMステータスが保留状態の間SES、Amazon DKIM は BYODKIMを使用して E メールに署名しないことに注意してください。Easy DKIM を有効にするための呼び出し (APIまたは コンソール経由) を行った時点から、 SESがDNS設定を確認するまでの間、E メールはDKIM署名SESなしで によって送信される場合があります。したがって、中間ステップを使用して、あるDKIM署名方法から別の署名方法に移行するか (たとえば、 BYODKIMを有効にしたドメインのサブドメインを使用して、Easy DKIM verification に合格したら削除します) 、アプリケーションのダウンタイム中にこのアクティビティを実行することをお勧めします。

検証済みドメイン ID DKIMの Easy のセットアップ

このセクションの手順は、既に作成したドメイン ID DKIMで Easy を設定するために必要なステップを示すように合理化されています。ドメイン ID をまだ作成していない場合、またはデフォルト設定セット、カスタムMAILFROMドメイン、タグなど、ドメイン ID をカスタマイズするために使用可能なすべてのオプションを表示する場合は、「」を参照してくださいthe section called “ドメイン ID の作成”。

Easy DKIMドメイン ID の作成の一環として、Amazon のSESデフォルトである 2048 ビットを受け入れるか、1024 ビットを選択してデフォルトを上書きするかを選択できる DKIMベースの検証を設定します。DKIM 署名キーの長さとその変更方法the section called “DKIM 署名キーの長さ”の詳細については、「」を参照してください。

ドメインDKIMに Easy を設定するには

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [設定] で、[ID] を選択します。
3. ID のリストで、ID のタイプをドメインでIDを選択します。

> **Note**
>
> ドメインを作成または検証する必要がある場合は、「ドメイン ID の作成」を参照してください。

4. 認証タブのDomainKeys識別メール (DKIM) コンテナで、編集を選択します。
5. 詳細DKIM設定コンテナで、ID タイプフィールドの Easy DKIM ボタンを選択します。
6. DKIM 署名キーの長さフィールドで、RSA_2048_BIT または RSA_1024_BITを選択します。
7. DKIM 署名フィールドで、有効チェックボックスをオンにします。
8. [Save changes] (変更の保存) をクリックします。
9. Easy でドメイン ID を設定したのでDKIM、DNSプロバイダーとの検証プロセスを完了する必要があります。Easy のDNS認証手順the section called “ドメイン ID の検証”に進んでくださいDKIM。

ID の Easy DKIM署名キーの長さを変更する

このセクションの手順では、署名アルゴリズムに必要な Easy DKIMビットを簡単に変更する方法を示します。提供されるセキュリティの強化には、常に 2048 ビットの署名長が優先されますが、1024 ビット長を使用する必要がある状況があります。たとえば、1024 DKIM のみをサポートするDNSプロバイダーを使用する必要がある場合があります。

転送中の E メールの配信可能性を維持するために、DKIMキーの長さを変更または反転できる頻度には制限があります。

E メールが転送中の場合、 DNSはパブリックキーを使用して E メールを認証します。したがって、キーを頻繁にまたは頻繁に変更すると、 は E メールをDKIM認証できないDNS場合があります。以前のキーは無効になっている可能性があるため、次の制限により保護されます。

- 既に設定されているキーの長さと同じキーの長さに切り替えることはできません。
- 24 時間の間に複数回異なるキーの長さに切り替えることはできません (ただし、その間の1024への最初のダウングレードでない限り) 。

次の手順を使用してキーの長さを変更する際、これらの制限のいずれかを無視すると、コンソールでは、無効だった理由とともに「the input you provided is invalid (指定した入力は無効です)」と示すエラーバナーが返されます。

DKIM 署名キーの長さビットを変更するには

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの設定で、検証済み ID を選択します。

3. ID のリストで、DKIM署名キーの長さを変更する ID を選択します。
4. 認証タブのDomainKeys識別メール (DKIM) コンテナで、編集を選択します。
5. 詳細DKIM設定コンテナで、DKIM署名キーの長さフィールドで RSA_2048_BIT または RSA_1024_BIT を選択します。
6. [Save changes] (変更の保存) をクリックします。

Amazon での確定的 Easy DKIM (DEED) の使用 SES

Deterministic Easy DKIM (DEED) は、複数の にわたるDKIM設定を管理するためのソリューションを提供します AWS リージョン。DNS 管理を簡素化し、一貫したDKIM署名を確保することで、 は、堅牢な E メール認証プラクティスを維持しながら、マルチリージョンの E メール送信オペレーションを合理化するDEEDのに役立ちます。

Deterministic Easy DKIM (DEED) とは

Deterministic Easy DKIM (DEED) は、Easy DKIMで設定された親ドメイン AWS リージョン に基づいて、すべての で一貫したDKIMトークンを生成する機能です。これにより、Easy で現在設定 AWS リージョン されている親 ID と同じDKIM署名設定を自動的に継承して維持する異なる で ID をレプリケートできますDKIM。ではDEED、親 ID に対してDNSレコードを 1 回発行するだけで、レプリカ ID は同じDNSレコードを使用してドメインの所有権を検証し、DKIM署名を管理します。

DNS 管理を簡素化し、一貫したDKIM署名を確保することで、 は、E メール認証のベストプラクティスを維持しながら、マルチリージョンの E メール送信オペレーションを合理化するDEEDのに役立ちます。

主要な用語

について話すときに使用される用語DEED :

- 親 ID – レプリカ ID DKIMの設定のソースとして機能する Easy で設定された検証DKIM済み ID。
- レプリカ ID – 同じDNS設定とDKIM署名設定を共有する親 ID のコピー。
- 親リージョン - AWS リージョン 親 ID が設定されている 。
- レプリカリージョン – AWS リージョン レプリカ ID がセットアップされている 。
- DEED identity – 親アイデンティティまたはレプリカアイデンティティとして使用されるアイデンティティ。 (新しい ID が作成されると、最初は通常の (非DEED) ID として扱われます。 ただし、レプリカが作成されると、アイデンティティはDEEDアイデンティティと見なされます) 。

DEED を使用する利点

を使用する主な利点DEEDは次のとおりです。

- DNS 管理の簡素化 – 親 ID に対してDNSレコードを 1 回だけ発行します。
- より簡単なマルチリージョンオペレーション – E メール送信オペレーションを新しいリージョンに拡張するプロセスを簡素化します。
- 管理オーバーヘッドの削減 – 親 ID からDKIM設定を一元管理します。

Deterministic Easy DKIM (DEED) の仕組み

レプリカ ID を作成すると、Amazon はDKIM署名キーを親 ID からレプリカ ID SESに自動的にレプリケートします。親 ID に加えられた後続のDKIMキーローテーションまたはキー長の変更は、すべてのレプリカ ID に自動的に伝播されます。

このプロセスには、次のワークフローが含まれます。

1. Easy AWS リージョン を使用して で親 ID を作成しますDKIM。
2. 親 ID に必要なDNSレコードを設定します。
3. 他の にレプリカ ID を作成し AWS リージョン、親 ID のドメイン名とDKIM署名リージョンを指定します。
4. Amazon は、親DKIMの設定をレプリカ ID SESに自動的にレプリケートします。

重要な考慮事項

- すでにレプリカになっている ID のレプリカを作成することはできません。
- 親 ID には Easy DKIM が有効になっている必要があります。ID のレプリカを作成BYODKIMしたり、手動で署名したりすることはできません。
- 親 ID は、すべてのレプリカ ID が削除されるまで削除できません。

を使用したレプリカ ID の設定 DEED

このセクションでは、 を使用してレプリカ ID を作成および検証する方法DEEDと、必要なアクセス許可の例を示します。

レプリカ ID の作成

レプリカ ID を作成するには：

1. レプリカアイデンティティ AWS リージョン を作成する で、 で SESコンソールを開きますhttps://console.aws.amazon.com/ses/。

   （SESコンソールでは、レプリカ ID はグローバル ID と呼ばれます）。
2. ナビゲーションペインで、ID を選択します。
3. [ID の作成] を選択します。
4. ID タイプでドメインを選択し、レDKIMプリケートして親として機能する Easy で設定された既存の ID のドメイン名を入力します。
5. 詳細DKIM設定を展開し、決定論的簡易 DKIMを選択します。
6. 親リージョンドロップダウンメニューから、グローバル (レプリカ) ID に入力したのと同じ名前の Easy DKIM署名付き ID が存在する親リージョンを選択します。（レプリカリージョンは、デフォルトでSESコンソールにサインインしたリージョンになります）。
7. DKIM 署名が有効になっていることを確認します。
8. （オプション) ドメイン ID に 1 つ以上のタグを追加します。
9. 設定を確認し、アイデンティティの作成を選択します。

の使用 AWS CLI：

Easy で設定された親 ID に基づいてレプリカ ID を作成するにはDKIM、次の例に示すように、親のドメイン名、レプリカ ID を作成するリージョン、および親DKIMの署名リージョンを指定する必要があります。

```
aws sesv2 create-email-identity --email-identity example.com --region us-west-2 --dkim-signing-attributes '{"DomainSigningAttributesOrigin": "AWS_SES_US_EAST_1"}'
```

前の例では、以下のようになっています。

1. を、レプリケートされる親ドメイン ID *example.com*に置き換えます。
2. を、レプリカドメイン ID が作成されるリージョン*us-west-2*に置き換えます。
3. を、レプリカアイデンティティにレプリケートされる Easy DKIM 署名設定を表す親DKIMの署名リージョン*AWS_SES_US_EAST_1*に置き換えます。

> **Note**
>
> AWS_SES_ プレフィックスは、 DKIMが Easy を使用して親 ID 用に設定されたことを示し DKIM、 US_EAST_1は作成された AWS リージョン です。

レプリカ ID 設定の検証

レプリカ ID を作成したら、親 ID DKIMの署名設定で正しく設定されていることを確認できます。

レプリカ ID を検証するには：

1. レプリカアイデンティティを AWS リージョン 作成した で、 で SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインで、ID を選択し、ID テーブルから検証する ID を選択します。
3. 認証タブで、DKIM設定フィールドにはステータスが表示され、親リージョンフィールドには、 を使用したアイデンティティDKIMの署名設定に使用されているリージョンが表示されますDEED。

の使用 AWS CLI：

レプリカのドメイン名とリージョンを指定する get-email-identity コマンドを使用します。

```
aws sesv2 get-email-identity --email-identity example.com --region us-west-2
```

レスポンスには、レプリカ ID が親 ID DKIMの署名設定で正常に設定されたことを示す親リージョンの値がSigningAttributesOriginパラメータに含まれます。

```
{
  "DkimAttributes": {
    "SigningAttributesOrigin": "AWS_SES_US_EAST_1"
  }
}
```

を使用するために必要なアクセス許可 DEED

DEED を使用するには、以下が必要です：

1. レプリカリージョンで E メール ID を作成するための標準アクセス許可。
2. 親リージョンからDKIM署名キーをレプリケートするアクセス許可。

DKIM レプリケーションIAMポリシーの例

次のポリシーでは、親 ID から指定されたレプリカリージョンへのキーレプリケーションDKIMの署名を許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDKIMReplication",
      "Effect": "Allow",
      "Action": "ses:ReplicateEmailIdentityDKIMSigningKey",
      "Resource": "arn:aws:ses:us-east-1:123456789124:identity/example.com",
      "Condition": {
        "ForAllValues:StringEquals": {
           "ses:ReplicaRegion": ["us-west-2", "eu-west-1"]
        }
      }
    }
  ]
}
```

ベストプラクティス

以下のベストプラクティスが推奨されます。

- 親リージョンとレプリカリージョンを計画する – 選択した親リージョンを考慮します。これは、レプリカリージョンで使用されるDKIM設定の信頼できるソースになるためです。
- 一貫したIAMポリシーを使用する – IAMポリシーで、意図したすべてのリージョンでDKIMレプリケーションが許可されていることを確認します。
- 親 ID をアクティブにしておく – レプリカ ID は親 ID DKIMの署名設定を継承することに注意してください。この依存関係により、すべてのレプリカ ID が削除されるまで親 ID を削除することはできません。

トラブルシューティング

で問題が発生した場合はDEED、次の点を考慮してください。

- 検証エラー — DKIMレプリケーションに必要なアクセス許可があることを確認します。
- レプリケーションの遅延 — 特に新しいレプリカ ID を作成する場合は、レプリケーションが完了するまでしばらく待ちます。
- DNS 問題 – 親 ID のDNSレコードが正しく設定され、伝達されていることを確認します。

Amazon SES で独自の DKIM 認証トークン (BYODKIM) を提供する

Easy DKIM を使用する代わりに、独自のパブリックキーとプライベートキーのペアを使用して DKIM 認証を設定することもできます。このプロセスは、「Bring Your Own DKIM(BYODKIM)」として知られています。

BYODKIM では、単一の DNS レコードを使用してドメインの DKIM 認証を設定できます。一方 Easy DKIM では、3 つの個別の DNS レコードを発行する必要があります。さらに、BYODKIM を使用すると、ドメインの DKIM キーを必要な回数だけローテーションできます。

このセクションのトピック:

- ステップ 1: キーペアを作成する
- ステップ 2: DNS プロバイダーのドメイン設定にセレクタおよびパブリックキーを追加する
- ステップ 3: BYODKIM を使用するようにドメインを設定し、検証する

> **Warning**
>
> 現在 Easy DKIM が有効になっていて、BYODKIM に移行している場合、BYODKIM のセットアップ中で、DKIM ステータスが保留中になっている間、Amazon SES は Easy DKIM を使用してメールに署名しないことに注意してください。BYODKIM を有効にする電話を (API またはコンソールを介して) かけたときと SES が DNS 設定を確認できるときまでの間に、SES によって DKIM 署名なしで E メールが送信されることがあります。したがって、中間ステップを使用して、一方の DKIM 署名方法からもう一方の DKIM 署名方法に移行する (例えば、Easy DKIM を有効にしてドメインのサブドメインを使用し、BYODKIM 検証に合格したら削除する) か、アプリケーションのダウンタイム中にこのアクティビティを実行することをお勧めします (存在する場合)。

ステップ 1: キーペアを作成する

独自の DKIM 機能を使用するには、まず RSA キーペアを作成する必要があります。

生成するプライベートキーは、PKCS #1 または PKCS #8 形式で、少なくとも 1024 ビットの RSA 暗号化と最大 2048 ビットを使用し、base64 (PEM) エンコードを使用してエンコードする必要があります。DKIM 署名キーの長さとその変更方法の詳細は、the section called “DKIM 署名キーの長さ”を参照してください。

> **Note**
>
> プライベートキーが最低 1024 ビットの RSA 暗号化、最大 2048 ビットで、base64 (PEM) エンコーディングを使用して生成されている限り、サードパーティーのアプリケーションおよびツールを使用して RSA キーペアを生成できます。

次の手順では、ほとんどの Linux、macOS、または Unix オペレーティングシステムに組み込まれている `openssl genrsa` コマンドを使用してキーペアを作成するコード例では、自動的に base64 (PEM) エンコードが使用されます。

Linux、macOS、またはUnix コマンドラインからキーペアを作成するには

1. コマンドラインで、以下のコマンドを入力して、*nnnn*を少なくとも1024のビット長で、2048までのビット長のプライベートキーに置き換えます。

   ```
   openssl genrsa -f4 -out private.key nnnn
   ```

2. コマンドラインで、以下のコマンドを入力してパブリックキーを生成します。

   ```
   openssl rsa -in private.key -outform PEM -pubout -out public.key
   ```

ステップ 2: DNS プロバイダーのドメイン設定にセレクタおよびパブリックキーを追加する

キーペアを作成したので、パブリックキーを TXT レコードとしてドメインの DNS 設定に追加する必要があります。

ドメインの DNS 設定にパブリックキーを追加するには

1. DNS またはホスティングプロバイダーの管理コンソールにサインインします。
2. ドメインの DNS 設定に新しいテキストレコードを追加します。レコードは、次の形式を使用する必要があります。

名前	タイプ	値
selector._domainkey.*example.com*	TXT	p=*yourPublicKey*

上の例に、以下の変更を加えます。

- *selector*は、キーを識別する一意の名前に置き換えます。

> **Note**
>
> いくつかの DNS プロバイダーでは、レコード名に下線 (_) を含めることが許可されていません。ただし、DKIM レコード名には下線が必要となります。DNS プロバイダーがレコード名に下線を含めることを許可しない場合、プロバイダーのカスタマーサポートにお問い合わせください。

- *example.com*をドメインに置き換えます。
- 上記の [Value] (値) 列に示すように、*yourPublicKey* を、以前に作成したパブリックキーに置き換え、プレフィックス p= を含めます。

> **Note**
>
> パブリックキーを DNS プロバイダーに公開 (追加) するときは、次のようにフォーマットする必要があります。
>
> - 生成されたパブリックキーの最初と最後の行 (それぞれ `-----BEGIN PUBLIC KEY-----` と `-----END PUBLIC KEY-----`) を削除する必要があります。さらに、生成されたパブリックキーの改行を削除する必要があります。結果の値は、スペースや改行を含まない文字列になります。
> - 上記の表の [Value] (値) 列に示すように、プレフィックス p= を含める必要があります。

プロバイダーによって、DNS レコードを更新する手順は異なります。次の表には、いくつかの広く使用されている DNS プロバイダーに関するドキュメントへのリンクが含まれています。このリストは網羅的なものではなく、推奨を意味するものでもありません。同様に、DNS プロバ

イダーがリストされていない場合、Amazon SES でドメインを使用できないことを意味するものでもありません。

DNS/ホスティングプロバイダー	ドキュメントのリンク
Amazon Route 53	詳細については、Amazon Route 53 デベロッパーガイドの「レコードの編集」を参照してください。
GoDaddy	Add a TXT record (外部リンク)
DreamHost	カスタム DNS レコードを追加する方法 (外部リンク)
Cloudflare	Managing DNS records in Cloudflare (外部リンク)
HostGator	HostGator/eNom で DNS レコードを管理する (外部リンク)
Namecheap	ドメインの TXT/SPF/DKIM/DMARC レコードを追加する方法 (外部リンク)
Names.co.uk	ドメイン DNS 設定の変更 (外部リンク)
Wix	Wix アカウントの TXT レコードの追加または更新 (外部リンク)

ステップ 3: BYODKIM を使用するようにドメインを設定し、検証する

BYODKIM は、新しいドメイン (現在 Amazon SES を経由した E メールの送信に使用していないドメイン) と既存のドメイン (コンソールまたはAWS CLIを使って、Amazon SES を介して設定済みのドメイン) の両方に対して設定できます。このセクションのAWS CLI手順を使用する前に、まずAWS CLIをインストールして設定する必要があります。詳細については、AWS Command Line Interface ユーザーガイドを参照してください。

オプション 1: BYODKIM を使用する新しいドメイン ID を作成する

このセクションでは、BYODKIM を使用する新しいドメイン ID を作成する手順について説明します。新しいドメイン ID とは、Amazon SES を使用して E メールを送信するように設定していないドメインです。

BYODKIM を使用するように既存のドメインを設定する場合は、代わりに「オプション 2: 既存のドメイン ID を設定する」の手順を実行します。

コンソールから BYODKIM を使用して ID を作成するには

- 「ドメイン ID の作成」の手順に従い、ステップ 8 に到達したら、BYODKIM 固有の指示に従います。

AWS CLIからBYODKIMを使用してIDを作成するには

新しいドメインを設定するには、Amazon SES API でCreateEmailIdentity オペレーションを実行します。

1. テキストエディタで、次のコードを貼り付けます。

```
{
    "EmailIdentity":"example.com",
    "DkimSigningAttributes":{
        "DomainSigningPrivateKey":"privateKey",
        "DomainSigningSelector":"selector"
    }
}
```

上の例に、以下の変更を加えます。

- *example.com*を、作成するドメインに置き換えます。
- *privateKey*をプライベートキーに置き換えます。

> **Note**
>
> 生成されたプライベートキーの最初と最後の行 (それぞれ -----BEGIN PRIVATE KEY----- と -----END PRIVATE KEY-----) を削除する必要があります。さら

に、生成されたプライベートキーの改行を削除する必要があります。結果の値は、スペースや改行を含まない文字列になります。

- *selector*は、ドメインの DNS 設定で TXT レコードを作成したときに指定した一意のセレクタに置き換えます。

終了したら、create-identity.jsonとしてファイルを保存します。

2. コマンドラインで以下のコマンドを入力します。

```
aws sesv2 create-email-identity --cli-input-json file://path/to/create-identity.json
```

上記のコマンドで、*path/to/create-identity.json*を、前のステップで作成したファイルの完全なパスに置き換えます。

オプション 2: 既存のドメイン ID を設定する

このセクションでは、BYODKIM を使用するために既存のドメイン ID を更新する手順について説明します。既存のドメイン ID は、Amazon SES を使用して E メールを送信するようにすでに設定されているドメインです。

コンソールから BYODKIM を使用してドメイン ID を更新するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインの設定で、検証済み ID を選択します。
3. ID のリストで、ID のタイプをドメインでIDを選択します。

> **Note**
>
> ドメインを作成または検証する必要がある場合は、「ドメイン ID の作成」を参照してください。

4. [認証] タブの [DomainKeys Identified Mail (DKIM)] ペインで、[編集] を選択します。
5. [DKIM の詳細設定] ペインで、[ID のタイプ] フィールドの [DKIM 認証トークン (BYODKIM) の提供] ボタンを選択します。
6. [プライベートキー] に、以前に生成したプライベートキーを貼り付けます。

> **Note**
>
> 生成されたプライベートキーの最初と最後の行 (それぞれ `-----BEGIN PRIVATE KEY-----` と `-----END PRIVATE KEY-----`) を削除する必要があります。さらに、生成されたプライベートキーの改行を削除する必要があります。結果の値は、スペースや改行を含まない文字列になります。

7. セレクタ名については、ドメインの DNS 設定で指定したセレクタの名前を入力します。
8. DKIM 署名フィールドで、[Enabled]ボックスにチェックを入れます。
9. [Save changes] をクリックします。

AWS CLIからBYODKIM を使用してドメイン ID を更新するには

既存のドメインを設定するには、Amazon SES API でのPutEmailIdentityDkimSigningAttributesオペレーションを使用します。

1. テキストエディタで、次のコードを貼り付けます。

```
{
    "SigningAttributes":{
        "DomainSigningPrivateKey":"privateKey",
        "DomainSigningSelector":"selector"
    },
    "SigningAttributesOrigin":"EXTERNAL"
}
```

上の例に、以下の変更を加えます。

- *privateKey*をプライベートキーに置き換えます。

> **Note**
>
> 生成されたプライベートキーの最初と最後の行 (それぞれ `-----BEGIN PRIVATE KEY-----` と `-----END PRIVATE KEY-----`) を削除する必要があります。さらに、生成されたプライベートキーの改行を削除する必要があります。結果の値は、スペースや改行を含まない文字列になります。

- *selector*は、ドメインの DNS 設定で TXT レコードを作成したときに指定した一意のセレクタに置き換えます。

終了したら、update-identity.jsonとしてファイルを保存します。

2. コマンドラインで以下のコマンドを入力します。

```
aws sesv2 put-email-identity-dkim-signing-attributes --email-identity example.com
 --cli-input-json file://path/to/update-identity.json
```

上のコマンドに、以下の変更を加えます。

- *path/to/update-identity.json*を、前のステップで作成したファイルへの完全なパスに置き換えます。
- *example.com*を、更新するドメインに置き換えます。

BYODKIM を使用するドメインの DKIM ステータスを検証する

コンソールからドメインの DKIM ステータスを検証するには

BYODKIM を使用するようにドメインを設定したら、SES コンソールを使用して、DKIM が正しく設定されていることを検証できます。

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインの設定で、検証済み ID を選択します。
3. ID のリストで、検証する DKIM ステータスの ID を選択します。
4. DNS 設定への変更が反映されるまでに最大 72 時間かかる場合があります。Amazon SES がドメインの DNS 設定の中から必要な DKIM レコードをすべて検出するとすぐに、検証プロセスは完了します。すべてが正しく設定されていれば、[DomainKeys Identified Mail (DKIM)] ペインで、ドメインの [DKIM configuration] (DKIM 設定) フィールドには [Successful] (成功) と表示され、[Summary] (概要) ペインで、[Identity status] (ID ステータス) フィールドには [Verified] (検証済み) と表示されます。

AWS CLI を使用してドメインの DKIM ステータスを検証するには

BYODKIM を使用するようにドメインを設定したら、GetEmailIdentity オペレーションを実行して、DKIM が正しく設定されていることを検証できます。

- コマンドラインで以下のコマンドを入力します。

  ```
  aws sesv2 get-email-identity --email-identity example.com
  ```

  上記のコマンドで、*example.com*をドメインに置き換えます。

  このコマンドは、次の例のようなセクションを含む JSON オブジェクトを返します。

  ```
  {
      ...
      "DkimAttributes": {
          "SigningAttributesOrigin": "EXTERNAL",
          "SigningEnabled": true,
          "Status": "SUCCESS",
          "Tokens": [ ]
      },
      ...
  }
  ```

  BYODKIMは、以下のすべてに該当する場合、ドメインに対して適切に設定されています。

  - `SigningAttributesOrigin`プロパティの値は`EXTERNAL`です。
  - `SigningEnabled`の値は`true`です。
  - `Status`の値は`SUCCESS`です。

Easy DKIM と BYODKIM の管理

Easy DKIM または BYODKIM で認証された ID の DKIM 設定は、ウェブベースの Amazon SES コンソールを使用するか、Amazon SES API を使用して管理できます。これらのいずれかの方法を使用して ID の DKIM レコードを取得するか、または、ID に対して DKIM 署名を有効または無効にすることができます。

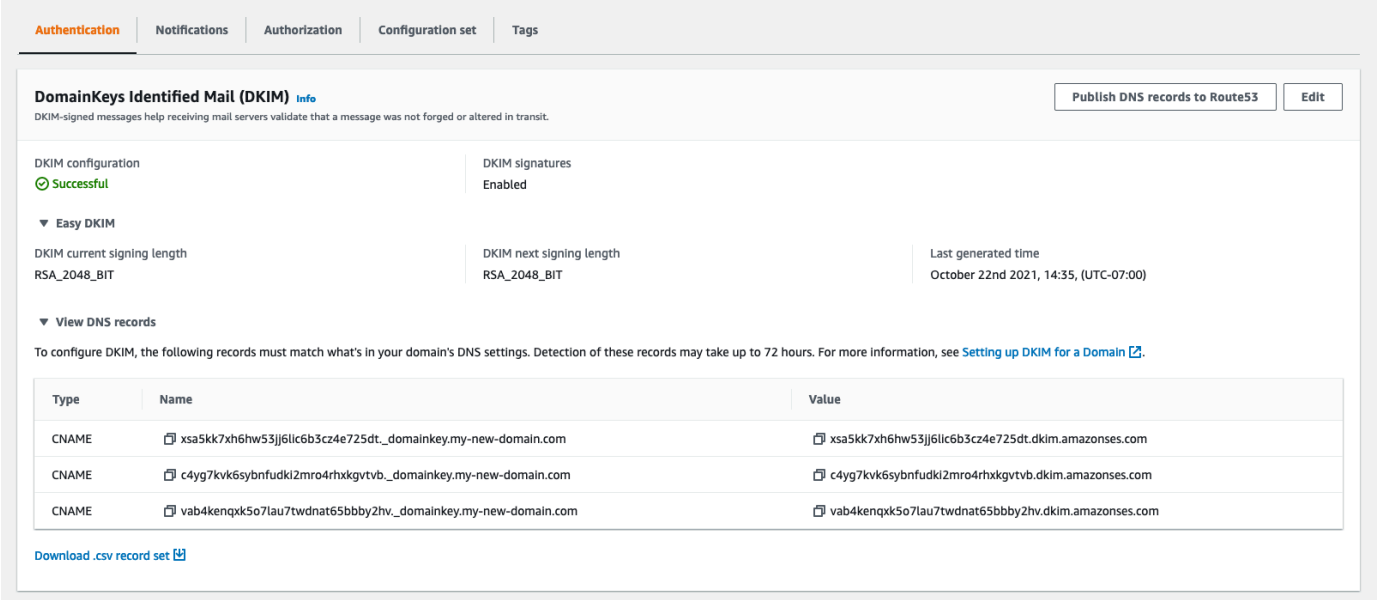
ID の DKIM レコードを取得する

Amazon SES コンソールを使用して、ドメインあるいは E メールアドレスの DKIM レコードをいつでも取得できます。

コンソールを使用して、ID の DKIM レコードを取得するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインの [設定] で、[検証済み ID] を選択します。
3. ID のリストで、DKIM レコードを取得する ID を選択します。
4. ID詳細ページの認証タブで、DNS レコードを表示するを拡大します。
5. Easy DKIM を使用した場合は 3 つの CNAME レコードをコピーし、このセクションに表示されている BYODKIM を使用した場合は TXT レコードをコピーします。または、[レコードセットを CSV としてダウンロード] を選択してコンピューターにこのレコードのコピーを保存することもできます。

   次の図で、[View DNS records] (DNS レコードの表示) セクションを拡大して、Easy DKIM に関連付けられた CNAME レコードの例を示します。



Amazon SES API を使用して、ID の DKIM レコードを取得することもできます。API とやり取りする一般的な方法は、AWS CLIを使用することです。

AWS CLIを使用して、ID の DKIM レコードを取得するには

1. コマンドラインから、以下のコマンドを入力します。

```
aws ses get-identity-dkim-attributes --identities "example.com"
```

前述の例で、*example.com* を DKIM レコードを取得する ID で置き換えます。E メールアドレスまたはドメインを指定できます。

2. このコマンドの出力には、次の例に示すように、DkimTokensセクションが含まれています。

```
{
    "DkimAttributes": {
        "example.com": {
            "DkimEnabled": true,
            "DkimVerificationStatus": "Success",
            "DkimTokens": [
                "hirjd4exampled5477y22yd23ettobi",
                "v3rnz522czcl46quexamplek3efo5o6x",
                "y4examplexbhyhnsjcmtvzotfvqjmdqoj"
            ]
        }
    }
}
```

トークンを使用してドメインの DNS 設定に追加する CNAME レコードを作成します。CNAME レコードを作成するには、次のテンプレートを使用します。

```
token1._domainkey.example.com CNAME token1.dkim.amazonses.com
token2._domainkey.example.com CNAME token2.dkim.amazonses.com
token3._domainkey.example.com CNAME token3.dkim.amazonses.com
```

token1 の各インスタンスを、get-identity-dkim-attributes コマンドを実行したときに受け取ったリストの最初のトークンに置き換え、*token2* のインスタンスすべてを、リストの 2 番目のトークンに置き換え、*token3* のインスタンスすべてを、リストの 3 番目のトークンに置き換えます。

例えば、前述の例に示されるトークンにこのテンプレートを適用すると、次のレコードが生成されます。

```
hirjd4exampled5477y22yd23ettobi._domainkey.example.com CNAME
 hirjd4exampled5477y22yd23ettobi.dkim.amazonses.com
v3rnz522czcl46quexamplek3efo5o6x._domainkey.example.com CNAME
 v3rnz522czcl46quexamplek3efo5o6x.dkim.amazonses.com
y4examplexbhyhnsjcmtvzotfvqjmdqoj._domainkey.example.com CNAME
 y4examplexbhyhnsjcmtvzotfvqjmdqoj.dkim.amazonses.com
```

> **Note**
>
> すべての AWS リージョンがデフォルトの SES DKIM ドメインである `dkim.amazonses.com` を使用するとは限りません。現在利用しているリージョンで、リージョン固有の DKIM ドメインが使用されているかを確認するには、「AWS 全般のリファレンス」の「DKIM domains table」を確認してください。

ID の Easy DKIM を無効にする

Amazon SES コンソールを使用して、ID の DKIM 認証を素早く無効にできます。

ID で DKIM を無効にするには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインの [設定] で、[検証済み ID] を選択します。
3. ID のリストで、DKIM を無効にする ID を選択します。
4. 認証タブのDomainKeys アイデンティファイドメール (DKIM)コンテナで、編集を選択します。
5. [Advanced DKIM settings] (DKIM の詳細設定) で、DKIM 署名フィールドの [Enabled] (有効) ボックスをオフにします。

また、Amazon SES API を使用して、ID の DKIM を無効にすることもできます。API とやり取りする一般的な方法は、AWS CLIを使用することです。

AWS CLI を使用して、ID の DKIM を無効にするには

- コマンドラインから、以下のコマンドを入力します。

```
aws ses set-identity-dkim-enabled --identity example.com --no-dkim-enabled
```

前述の例で、*example.com* を DKIM を無効にする ID で置き換えます。E メールアドレスまたはドメインを指定できます。

ID の Easy DKIM を有効にする

前に ID で DKIM を無効化した場合、Amazon SES コンソールを使用して再度有効化できます。

ID で DKIM を有効にするには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインの [設定] で、[検証済み ID] を選択します。
3. ID のリストで、DKIM を有効にする ID を選択します。
4. 認証タブのDomainKeys アイデンティファイドメール (DKIM)コンテナで、編集を選択します。
5. DKIM の詳細設定で、DKIM 署名フィールドの[Enabled]ボックスにチェックを入れます。

また、Amazon SES API を使用して、ID の DKIM を有効にすることもできます。API とやり取りする一般的な方法は、AWS CLIを使用することです。

AWS CLI を使用して、ID の DKIM を有効にするには

- コマンドラインから、以下のコマンドを入力します。

```
aws ses set-identity-dkim-enabled --identity example.com --dkim-enabled
```

前述の例で、*example.com* を DKIM を有効にする ID で置き換えます。E メールアドレスまたはドメインを指定できます。

E メールアドレス ID に継承された DKIM 署名を上書きする

このセクションでは、Amazon SES で検証済みの特定の E メールアドレス ID に、親ドメインから継承された DKIM 署名プロパティを上書き (無効化または有効化) する方法を説明します。DNS 設定はドメインレベルで構成されているため、既に所有するドメインに属する E メールアドレス ID に対してのみこれを行うことができます。

> **Important**
>
> E メールアドレス ID の DKIM 署名を無効化/有効化することはできません。
>
> - 所有していないドメインの場合。例えば、gmail.com や hotmail.com アドレスの DKIM 署名を切り替えることはできません。
> - 所有するドメインだが、Amazon SES でまだ検証されていない場合。
> - 所有するドメインだが、そのドメインで DKIM 署名が有効になっていない場合。

このセクションは、以下のトピックで構成されます。

- 継承された DKIM 署名プロパティについて理解する
- E メールアドレス ID に DKIM 署名を上書きする (コンソール)
- E メールアドレス ID に DKIM 署名を上書きする (AWS CLI)

継承された DKIM 署名プロパティについて理解する

まず、Easy DKIM または BYODKIM が使用されているかどうかにかかわらず、そのドメインが DKIM で設定されている場合、E メールアドレス ID が親ドメインから DKIM 署名プロパティを継承することを理解することが重要です。したがって、E メールアドレス ID で DKIM 署名を無効または有効にすると、次の重要な点に基づいてドメインの DKIM 署名プロパティが上書きされます。

- E メールアドレスが属するドメインで既に DKIM をセットアップしている場合には、E メールアドレス ID にも同じく DKIM 署名を有効にする必要はありません。
  - ドメインで DKIM をセットアップする場合、Amazon SES が親ドメインから継承された DKIM プロパティを介して、このドメインのすべてのアドレスからのすべての E メールを自動的に認証します。
- 特定の E メールアドレス ID の DKIM 設定により、そのアドレスが属する親ドメインまたはサブドメイン (該当する場合) の設定が自動的に上書きされます。

E メールアドレス ID の DKIM 署名プロパティが親ドメインから継承されるため、これらのプロパティを上書きする場合は、次の表で説明された上書きの階層ルールに注意する必要があります。

親ドメインで DKIM 署名が有効になっていない	親ドメインで DKIM 署名が有効になっている
E メールアドレス ID で DKIM 署名を有効にすることができません。	E メールアドレス ID で DKIM 署名を無効にできます。
	E メールアドレス ID で DKIM 署名を再度有効にできます。

通常、DKIM 署名の無効化はお勧めしません。送信者の評価を損うリスクがあるだけでなく、送信したメールが迷惑メールやスパムフォルダに移動されたり、ドメインが偽装されたりするリスクが高まるためです。

ただし、特定のユースケースや、DKIM 署名を永続的または一時的に無効にしたり、後で再有効化したりする必要があるような通常とは異なるビジネス上の意思決定がなされた場合に備えて、ドメインから継承された DKIM 署名プロパティを E メールアドレス ID に上書きするための機能があります。

E メールアドレス ID に DKIM 署名を上書きする (コンソール)

以下の SES コンソールの手順では、Amazon SES で既に検証済みの特定の E メールアドレス ID に、親ドメインから継承された DKIM 署名プロパティを上書き (無効化または有効化) する方法を説明しています。

コンソールを使用して E メールアドレス ID の DKIM 署名を無効/有効にするには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインの [設定] で、[検証済み ID] を選択します。
3. ID のリストで、[Identity type] (ID のタイプ) が [Email address] (E メールアドレス) であり、認証済みドメインのいずれかに属する ID を選択します。
4. 認証タブのDomainKeys アイデンティファイドメール (DKIM)コンテナで、編集を選択します。

> **Note**
>
> [Authentication] (認証) タブは、選択した電子メールアドレス ID が SES によって既に検証済みのドメインに属している場合にのみ表示されます。ドメインをまだ検証していない場合は、「ドメイン ID の作成」を参照してください。

5. [Advanced DKIM settings] (DKIM の詳細設定) の [DKIM signatures] (DKIM 署名) フィールドで、[Enabled] (有効) チェックボックスをオフにして DKIM 署名を無効にします。DKIM 署名を再度有効にするにはオンにします (以前に上書きされていた場合)。
6. [Save changes] (変更の保存) をクリックします。

E メールアドレス ID に DKIM 署名を上書きする (AWS CLI)

次の例では、 AWS CLI を介し SES API コマンドとパラメータで、親ドメインから継承された DKIM 署名プロパティを、SES で検証済みの特定の E メールアドレス ID に上書き (無効化または有効化) します。

AWS CLI を使用して E メールアドレス ID の DKIM 署名を無効/有効にするには

- 例えば、example.com ドメインを所有していて、そのドメインの E メールアドレスのいずれかに対して DKIM 署名を無効にする場合は、コマンドラインで次のコマンドを入力します。

```
aws sesv2 put-email-identity-dkim-attributes --email-identity marketing@example.com
 --no-signing-enabled
```

  a. *marketing@example.com* を DKIM 署名を無効にする E メールアドレス ID に置き換えます。
  b. `--no-signing-enabled` を使用すると、DKIM 署名が無効化されます。DKIM 署名を再度有効にするには、`--signing-enabled` を使用します。

Amazon SES 内で手動での DKIM 署名

Easy DKIM を使用する代わりに、手動で DKIM 署名をメッセージに追加し、Amazon SES を使用してこのメッセージを送信することもできます。手動でメッセージを署名することを選択する場合、まず DKIM 署名を作成する必要があります。メッセージと DKIM 署名を作成したら、SendRawEmail API を使用してメッセージを送信できます。

E メールを手動で署名する場合、次の要素を考慮してください。

- Amazon SES を使用して送信するすべてのメッセージには、amazonses.comのドメインの署名を参照する DKIM ヘッダーが含まれています(つまり、d=amazonses.comという文字列が含まれています)。そのため、メッセージを手動で署名する場合、そのメッセージには、ドメインのヘッダーと amazonses.comのためにAmazon SESが自動で作成するヘッダーという、2つのDKIM ヘッダーが含まれます。

- Amazon SES は、メッセージに手動で追加した DKIM 署名を検証しません。メッセージの DKIM 署名にエラーがある場合、このメッセージはEメールプロバイダーによって拒否される可能性があります。
- メッセージに署名する場合は、少なくとも 1024 ビットのビット長を使用する必要があります。
- 次のフィールドには署名しないでください。メッセージ ID、日付、Return-Path、Bounces-To。

> **Note**
>
> Amazon SES SMTP インタフェースでの E メールの送信に E メールクライアントを使用する場合、クライアントはメッセージで自動的に DKIM 署名を実行することがあります。一部のクライアントは上述のフィールドのいずれかに署名することがあります。デフォルトで署名されたフィールドについての情報は、Eメールのクライアントのドキュメンテーションを参照してください。

Amazon SES における SPF を使った E メールの認証

Sender Policy Framework(SPF) は、E メールのなりすましを防ぐために設計された E メールの検証標準です。ドメイン所有者は SPF を使用して、自分のドメインからメールを送信できるサーバーをメールプロバイダーに通知します。SPF はRFC 7208で定義されています。

Amazon SES から送信するメッセージには、MAIL FROM ドメインとして `amazonses.com` のサブドメインが自動的に使用されます。デフォルトの MAIL FROM ドメインがメールを送信するアプリケーション (この場合 Amazon SES) と一致するため、これらのメッセージは SPF (Sender Policy Framework) 認証に合格します。したがって、SES では、SPF は暗黙的に自動設定されます。

SES のデフォルトの MAIL FROM ドメインを使用せず、所有しているドメインのサブドメインを使用する場合、SES ではカスタム MAIL FROM ドメインを使用すると認識されます。そのためには、カスタム MAIL FROM ドメインに独自の SPF レコードを公開する必要があります。また、SES ではカスタム MAIL FROM ドメインがメールプロバイダーから送信されたバウンスと苦情の通知を受信できるように、MX レコードを設定する必要があります。

SPF 認証をセットアップする方法の説明

SPF を使用してドメインを設定する手順と、MX レコードと SPF (タイプ TXT) レコードを発行する方法は、「the section called “カスタムMAILFROMドメインの使用”」で説明されています。

カスタムMAILFROMドメインの使用

E メールが送信されると、送信元を示す 2 つのアドレスがあります。メッセージ受信者に表示される送信元アドレスと、メッセージの送信元を示すMAILFROMアドレスです。MAIL FROM アドレスは、エンベロープ送信者、エンベロープ送信元、バウンスアドレス、リターンパスアドレスと呼ばれることもあります。メールサーバーはMAILFROM、アドレスを使用してバウンスメッセージやその他のエラー通知を返します。MAIL FROM アドレスは通常、メッセージのソースコードを表示する受信者のみが表示できます。

Amazon は、独自の (カスタム) MAILFROMドメインを指定しない限り、送信するメッセージのドメインをデフォルト値SESに設定します。このセクションでは、カスタムMAILFROMドメインを設定する利点と、セットアップ手順について説明します。

カスタムMAILFROMドメインを使用する理由

Amazon 経由で送信するメッセージは、デフォルトのドメイン`amazonses.com`として のサブMAILFROMドメインSESを自動的に使用します。送信者ポリシーフレームワーク (SPF) 認証は、デフォルトのMAILFROMドメインが E メールを送信したアプリケーションと一致するため、これらのメッセージを正常に検証します。この場合、 ですSES。

SES デフォルトのMAILFROMドメインを使用せず、所有するドメインのサブドメインを使用する場合は、 でカスタムMAILFROMドメインを使用するSESと呼ばれます。そのためには、カスタムMAILFROMドメインの独自のSPFレコードを発行する必要があります。さらに、 SESでは、ドメインが E メールプロバイダーから送信されるバウンスや苦情の通知を受信できるように、MX レコードを設定する必要があります。

カスタムMAILFROMドメインを使用すると、SPF、DKIM、またはその両方を使用して、ドメインベースのメッセージ認証、レポート、および適合性 (DMARC) 検証を柔軟に実行できます。 DMARC では、送信者のドメインが、ドメインから送信された E メールが 1 つ以上の認証システムによって保護されていることを示すことができます。DMARC 検証を行うには、 the section called “SPF を介した DMARC への準拠”と の 2 つの方法がありますthe section called “DKIM を介した DMARC への準拠”。

カスタムMAILFROMドメインの選択

以下では、MAILFROMドメインという用語は、常に所有するドメインのサブドメインを指します。カスタムドメインに使用するこのサブMAILFROMドメインは、他のものには使用せず、次の要件を満たしている必要があります。

- MAIL FROM ドメインは、検証済み ID (E メールアドレスまたはドメイン) の親ドメインのサブドメインである必要があります。
- MAIL FROM ドメインは、E メールの送信にも使用するサブドメインにしないでください。
- MAIL FROM ドメインは、E メールの受信に使用するサブドメインにすることはできません。

カスタムMAILFROMドメインSPFでの の使用

Sender Policy Framework (SPF) は、E メールのなりすましを防ぐように設計された E メール検証標準です。カスタムMAILFROMドメインを で設定SPFして、カスタムMAILFROMドメインから E メールを送信できるサーバーを E メールプロバイダーに指示できます。 SPFは RFC7208 で定義されています。

を設定するにはSPF、カスタムMAILFROMドメインDNSの設定にTXTレコードを発行します。このレコードには、カスタムMAILFROMドメインを使用して から E メールを送信することを許可するサーバーのリストが含まれています。E メールプロバイダーは、カスタムMAILFROMドメインからメッセージを受信すると、そのドメインのDNSレコードをチェックして、E メールが承認されたサーバーから送信されたことを確認します。

に準拠する方法としてこのSPFレコードを使用する場合はDMARC、送信元アドレスのドメインがMAILFROMドメインと一致する必要があります。「the section called “SPF を介した DMARC への準拠”」を参照してください。

次のセクション ではthe section called “カスタムMAILFROMドメインの設定”、カスタムMAILFROMドメインSPF用に を設定する方法について説明します。

カスタムMAILFROMドメインの設定

カスタムMAILFROMドメインを設定するプロセスでは、ドメインDNSの設定にレコードを追加する必要があります。 SESでは、MX レコードを発行して、ドメインが E メールプロバイダーから送信されるバウンスおよび苦情の通知を受信できるようにする必要があります。また、Amazon がドメインから E メールを送信する権限があることを証明するために、 SPF (タイプ SES TXT) レコードを発行する必要があります。

ドメイン全体またはサブMAILFROMドメイン、および個々の E メールアドレスにカスタムドメインを設定できます。次の手順は、Amazon SESコンソールを使用してカスタムMAILFROMドメインを設定する方法を示しています。SetIdentityMailFromDomain API オペレーションを使用してカスタムMAILFROMドメインを設定することもできます。

検証済みMAILFROMドメインのカスタムドメインの設定

以下の手順では、MAILFROMドメイン全体またはサブドメインのカスタムドメインを設定して、そのドメインのアドレスから送信されるすべてのメッセージがこのカスタムMAILFROMドメインを使用するようにする方法を示します。

指定されたカスタムドメインを使用するように検証済みMAILFROMドメインを設定するには

1. で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. 左側のナビゲーションペインの [設定] の下にある [ID] を選択します。
3. ID のリストで、設定する ID を選択します。その際、ID のタイプを [Domain] (ドメイン)、ステータスを[Verified] (検証済み) にします。

   - ステータスが [Unverified] (未検証) の場合、DNS プロバイダーとのDKIMドメイン ID の検証の手順を完了させてから E メールアドレスのドメインを検証します。
4. カスタムMAILFROMドメインペインの画面の下部で、編集 を選択します。
5. [General details] (一般的な詳細) ペインで、次の操作を行います。

   a. 「カスタムMAILFROMドメインを使用する」チェックボックスを選択します。

   b. MAIL FROM ドメインには、ドメインとして使用するサブMAILFROMドメインを入力します。

   c. [Behavior on MX failure] (MX 障害時の動作) で、次のオプションのいずれかを選択します。

      - デフォルトMAILFROMドメインを使用する – カスタムMAILFROMドメインの MX レコードが正しく設定されていない場合、Amazon は のサブドメインSESを使用しますamazonses.com。サブドメインは、Amazon AWS リージョン を使用する SES によって異なります。
      - メッセージ拒否 – カスタムMAILFROMドメインの MX レコードが正しく設定されていない場合、Amazon はMailFromDomainNotVerified工ラーSESを返します。このドメインから送信しようとした E メールは自動的に拒否されます。

   d. [変更を保存する] を選択すると、前の画面に戻ります。
6. カスタムMAILFROMドメインのDNSサーバーに MX レコードと SPF (タイプ TXT) レコードを発行します。

   カスタムMAILFROMドメインペインで、DNSレコードの発行テーブルに、ドメインDNSの設定に発行 (追加TXT) する必要がある の MX レコードと SPF (タイプ) レコードが表示されるようになりました。これらのレコードでは、次の表に示す形式を使用します。

名前	タイプ	値
subdomain .*domain*.*com*	MX	10 feedback-smtp.*region*.amazonses.com
subdomain .*domain*.*com*	TXT	「v=spf1 include:amazonses.com すべて」

上記のレコードでは、以下のようになります。

- *subdomain*.*domain*.*com* にMAILFROMサブドメインが入力されます
- *region* には、MAILFROMドメイン AWS リージョン を検証する の名前 (us-west-2、us-east-1、 eu-west-1など) が入力されます。
- MX 値とともにリストされている 10 番はメールサーバーの優先順であり、DNSプロバイダーの で指定された別の値フィールドに入力する必要があります。 GUI
- SPFのTXTレコード値には通常引用符を含める必要がありますが、DNSプロバイダーによってはそれらを必要としない場合があります。

Publish DNS records テーブルから、各値の横にあるコピーアイコンを選択して MX レコードと SPF (タイプ TXT) レコードをコピーし、DNSプロバイダーの の対応するフィールドに貼り付けますGUI。または、[レコードセットを CSV としてダウンロード] を選択してコンピューターにこのレコードのコピーを保存することもできます。

> **Important**
>
> - MX レコードと SPF (タイプ TXT) レコードを発行する具体的な手順は、 DNSまたはホスティングプロバイダーによって異なります。これらのレコードをドメインDNSの設定に追加する方法については、プロバイダーのドキュメントを参照するか、プロバイダーにお問い合わせください。
> - Amazon でカスタムMAILFROMドメインを正常にセットアップするには SES、MAILFROMドメインのDNSサーバーに MX レコードを 1 つだけ発行する必要

があります。MAIL FROM ドメインに複数の MX レコードがある場合、Amazon でのカスタムMAILFROMセットアップSESは失敗します。

Route 53 がMAILFROMドメインDNSのサービスを提供し、Route 53 に使用するのと同じアカウント AWS Management Console で にサインインしている場合は、Route 53 を使用してレコードを発行を選択します。DNS レコードはドメインDNSの設定に自動的に適用されます。

別のDNSプロバイダーを使用する場合は、MAILFROMドメインのDNSサーバーにDNSレコードを手動で発行する必要があります。ドメインのDNSサーバーにDNSレコードを追加する手順は、ウェブホスティングサービスまたはDNSプロバイダーによって異なります。

ドメインのDNSレコードを発行する手順は、使用するDNSプロバイダーによって異なります。次の表には、広く使用されているいくつかのDNSプロバイダーのドキュメントへのリンクが含まれています。このリストは網羅的なものではなく、承認を意味するものでもありません。同様に、DNSプロバイダーがリストされていない場合、MAILFROMドメイン設定をサポートしていないことを意味するものではありません。

DNS/ホスティングプロバイダー名	ドキュメントのリンク
GoDaddy	• MX: MX レコードを追加する (外部リンク) • TXT: TXTレコードを追加する (外部リンク)
DreamHost	• MX: MX レコードを変更する方法 (外部リンク) • TXT: カスタムDNSレコードを追加する方法 (外部リンク)
Cloudflare	• MX: メールまたは MX レコードを追加または編集する方法 (外部リンク) • TXT: Cloudflare でのDNSレコードの管理 (外部リンク)

DNS/ホスティングプロバイダー名	ドキュメントのリンク
HostGator	• MX: MX メールレコードのセットアップ (外部リンク) • TXT: HostGator/ を使用してDNSレコードを管理するeNom (外部リンク)
Namecheap	• MX: メールサービスに必要な MX レコードを設定する方法 (外部リンク) • TXT: ドメインのTXT/SPF/DKIM/DMARCレコードを追加する方法 (外部リンク)
Names.co.uk	• MX: ドメインDNSの設定の変更 (外部リンク) • TXT: ドメインDNS設定の変更 (外部リンク)
Wix	• MX: Wix アカウントの MX レコードの追加または更新 (外部リンク) • TXT: Wix アカウントのTXTレコードの追加または更新 (外部リンク)

Amazon がレコードが設定されていることSESを検出すると、カスタムMAILFROMドメインが正常に設定されたことを通知する E メールが届きます。DNS プロバイダーによっては、Amazon が MX レコードSESを検出するまでに最大 72 時間の遅延が発生する場合があります。

検証済み E メールアドレスのカスタムMAILFROMドメインの設定

特定の E メールアドレスのカスタムMAILFROMドメインを設定することもできます。E メールアドレスのカスタムMAILFROMドメインを設定するには、E メールアドレスが関連付けられているドメインのDNSレコードを変更する必要があります。

> **Note**
>
> 所有していないMAILFROMドメインのアドレスにカスタムドメインを設定することはできません (たとえば、ドメインに必要なDNSレコードを追加できないため、gmail.comドメインのMAILFROMアドレスにカスタムドメインを作成することはできません)。

指定されたMAILFROMドメインを使用するように検証済み E メールアドレスを設定するには

1. で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. 左側のナビゲーションペインの [設定] の下にある [ID] を選択します。
3. ID のリストで、設定する ID を選択します。その際、ID のタイプを [Email address] (E メールアドレス)、ステータスを[Verified] (検証済み) にします。
   - ステータスが [Unverified] (未検証) の場合、E メールアドレス ID の検証 の手順を完了させてから E メールアドレスのドメインを検証します。
4. MAIL FROM ドメイン タブで、カスタムMAILFROMドメインペインで編集 を選択します。
5. [General details] (一般的な詳細) ペインで、次の操作を行います。
   a. 「カスタムMAILFROMドメインを使用する」チェックボックスを選択します。
   b. MAIL FROM ドメインには、ドメインとして使用するサブMAILFROMドメインを入力します。
   c. [Behavior on MX failure] (MX 障害時の動作) で、次のオプションのいずれかを選択します。
      - デフォルトMAILFROMドメインを使用する – カスタムMAILFROMドメインの MX レコードが正しく設定されていない場合、Amazon は のサブドメインSESを使用しますamazonses.com。サブドメインは、Amazon AWS リージョン を使用する SES によって異なります。
      - メッセージ拒否 – カスタムMAILFROMドメインの MX レコードが正しく設定されていない場合、Amazon はMailFromDomainNotVerifiedエラーSESを返します。この E メールアドレスから送信しようとした E メールは自動的に拒否されます。
   d. [変更を保存する] を選択すると、前の画面に戻ります。
6. カスタムMAILFROMドメインのDNSサーバーに MX レコードと SPF (タイプ TXT) レコードを発行します。

カスタムMAILFROMドメインペインで、DNSレコードの発行テーブルに、ドメインDNSの設定に発行 (追加TXT) する必要がある の MX レコードと SPF (タイプ) レコードが表示されるようになりました。これらのレコードでは、次の表に示す形式を使用します。

名前	タイプ	値
subdomain .*domain*.*com*	MX	10 feedback-smtp.*region*.amazonses.com
subdomain .*domain*.*com*	TXT	「v=spf1 include:amazonses.com すべて」

上記のレコードでは、以下のようになります。

- *subdomain*.*domain*.*com* にMAILFROMサブドメインが入力されます
- *region* には、MAILFROMドメイン AWS リージョン を検証する の名前 (us-west-2、us-east-1、 eu-west-1など) が入力されます。
- MX 値とともにリストされている 10 番はメールサーバーの優先順であり、DNSプロバイダーの で指定された別の値フィールドに入力する必要があります。 GUI
- SPFのTXTレコード値には引用符を含める必要があります

Publish DNS records テーブルから、各値の横にあるコピーアイコンを選択して MX レコードと SPF (タイプ TXT) レコードをコピーし、DNSプロバイダーの の対応するフィールドに貼り付けますGUI。または、[レコードセットを CSV としてダウンロード] を選択してコンピューターにこのレコードのコピーを保存することもできます。

> **Important**
>
> Amazon でカスタムMAILFROMドメインを正常にセットアップするには SES、MAILFROMドメインのDNSサーバーに MX レコードを 1 つだけ発行する必要があります。MAIL FROM ドメインに複数の MX レコードがある場合、Amazon でのカスタムMAILFROMセットアップSESは失敗します。

Route 53 がMAILFROMドメインDNSのサービスを提供し、Route 53 に使用するのと同じアカウント AWS Management Console で にサインインしている場合は、Route 53 を使用してレコードを発行を選択します。DNS レコードはドメインDNSの設定に自動的に適用されます。

別のDNSプロバイダーを使用する場合は、MAILFROMドメインのDNSサーバーにDNSレコードを手動で発行する必要があります。ドメインのDNSサーバーにDNSレコードを追加する手順は、ウェブホスティングサービスまたはDNSプロバイダーによって異なります。

ドメインのDNSレコードを発行する手順は、使用するDNSプロバイダーによって異なります。次の表には、広く使用されているいくつかのDNSプロバイダーのドキュメントへのリンクが含まれています。このリストは網羅的なものではなく、承認を意味するものでもありません。同様に、DNSプロバイダーがリストされていない場合、MAILFROMドメイン設定をサポートしていないことを意味するものではありません。

DNS/ホスティングプロバイダー名	ドキュメントのリンク
GoDaddy	• MX: MX レコードを追加する (外部リンク) • TXT: TXTレコードを追加する (外部リンク)
DreamHost	• MX: MX レコードを変更する方法 (外部リンク) • TXT: カスタムDNSレコードを追加する方法 (外部リンク)
Cloudflare	• MX: メールまたは MX レコードを追加または編集する方法 (外部リンク) • TXT: Cloudflare でのDNSレコードの管理 (外部リンク)
HostGator	• MX: MX レコードの変更 - Windows (外部リンク) • TXT: HostGator/ を使用してDNSレコードを管理するeNom (外部リンク)

DNS/ホスティングプロバイダー名	ドキュメントのリンク
Namecheap	• MX: メールサービスに必要な MX レコードを設定する方法 (外部リンク) • TXT: ドメインのTXT/SPF/DKIM/DMARCレコードを追加する方法 (外部リンク)
Names.co.uk	• MX: ドメインDNSの設定の変更 (外部リンク) • TXT: ドメインの設定の変更 DNS (外部リンク)
Wix	• MX: Wix アカウントの MX レコードの追加または更新 (外部リンク) • TXT: Wix アカウントのTXTレコードの追加または更新 (外部リンク)

Amazon がレコードが設定されていることSESを検出すると、カスタムMAILFROMドメインが正常に設定されたことを通知する E メールが届きます。DNS プロバイダーによっては、Amazon が MX レコードSESを検出するまでに最大 72 時間の遅延が発生する場合があります。

Amazon でのカスタムMAILFROMドメイン設定の状態 SES

カスタムMAILFROMドメインを使用するように ID を設定した後、Amazon がDNS設定で必要な MX レコードを検出SESしようとしている間、セットアップの状態は「保留中」になります。その後、状態は Amazon が MX レコードSESを検出するかどうかに応じて変わります。次の表は、E メール送信の動作と、各状態に関連付けられた Amazon SESアクションを示しています。状態が変わるたびに、Amazon は に関連付けられた E メールアドレスに通知SESを送信します AWS アカウント。

状態	E メール送信動作	Amazon SESアクション
保留中	カスタムMAILFROMフォールバック設定を使用します	Amazon は、必要な MX レコードを 72 時間検出

状態	E メール送信動作	Amazon SESアクション
		SESしようとします。検出できない場合、状態は "Failed" に変化します。
Success (成功)	カスタムMAILFROMドメインを使用する	Amazon は、必要な MX レコードが設定されていることSESを継続的に確認します。
Temporary Failure	カスタムMAILFROMフォールバック設定を使用します	Amazon は、必要な MX レコードを 72 時間検出SESしようとします。検出できない場合、状態は "Failed" に変化します。検出できた場合、状態は "Success" に変化します。

状態	E メール送信動作	Amazon SESアクション
失敗	カスタムMAILFROMフォールバック設定を使用します	Amazon SES は、必要な MX レコードを検出しようとしなくなりました。カスタムMAILFROMドメインを使用するには、 のセットアッププロセスを再起動する必要がありますカスタムMAILFROMドメインの設定。

Amazon SES の DMARC 認証プロトコルへの準拠

DMARC (Domain-based Message Authentication, Reporting and Conformance) は、SPF (Sender Policy Framework) および DKIM (DomainKeys Identified Mail) を使用して、E メールのなりすましやフィッシングを検出する、E メール認証プロトコルです。DMARC に準拠するには、メッセージを SPF または DKIM で認証する必要があります。理想的には、両方を DMARC で使用すると、E メール送信に関して可能な限り高いレベルの保護を確保できます。

SPF と DKIM の機能と、DMARC がこの 2 つを連携する方法を簡単に説明します。

- SPF – DNS が使用する DNS TXT レコードを介して、カスタム MAIL FROM ドメインに代わってメールを送信できるメールサーバーを特定します。受信者のメールシステムは、SPF TXT レコードを参照して、カスタムドメインからのメッセージが承認済みのメッセージングサーバーから送信されたかを判断します。基本的に、SPF はなりすまし防止目的で設計されているとはいえ、実際には SPF が影響を受けやすいなりすまし手法もあるため、DMARC とともに DKIM も使用する必要があります。
- DKIM – E メールヘッダーのアウトバウンドメッセージにデジタル署名を追加します。受信 E メールシステムは、このデジタル署名を使用して、受信する E メールがドメインが所有するキーに

よって署名されているかの検証に役立てます。ただし、受信 E メールシステムがメッセージを転送した場合、メッセージのエンベロープは SPF 認証を無効にするような方法で変更されます。デジタル署名は E メールヘッダーの一部であるため、E メールメッセージに保持されます。このため、メッセージがメールサーバー間で転送された場合でも、(メッセージコンテンツが変更されない限り) DKIM は機能します。

- DMARC – SPF と DKIM の少なくともいずれかでドメインのアライメントがとれていることを確認します。SPF と DKIM を単独で使用しても、送信元アドレスが認証されるかは保証されません (これは受信者が E メールクライアントで目にするメールアドレスです)。SPF は、MAIL FROM アドレスで指定されたドメインのみをチェックします (受信者には表示されません)。DKIM は、DKIM 署名で指定されたドメインのみをチェックします (これも受信者には表示されません)。DMARC は、SPF または DKIM のいずれかでドメインのアライメントが適切であることを求めることで、このような 2 つの問題に対処します。
  - SPF が DMARC アライメントで適格になるには、送信元アドレスのドメインが MAIL FROM アドレス (Return-Path やエンベロープ送信元アドレスとも呼ばれます) のドメインと一致する必要があります。Return-Path (MAIL FROM) は、プロバイダー (SES) が所有するアドレスを使用して追跡するバウンスや苦情に使用されます。このため、転送されたメールの場合には削除され、ほぼ不可能です。またはサードパーティーの一括 E メールプロバイダー経由でメールを送信する場合にもほとんど不可能です。
  - DKIM が DMARC アライメントに適格になるには、DKIM 署名で指定されたドメインが、送信元アドレスのドメインと一致する必要があります。お客様の代理でメールを送信するサードパーティーの送信者やサービスを使用する場合、サードパーティーの送信者が DKIM 署名向けに適切に設定され、お客様のドメイン内に適切な DNS レコードが追加されていることを確認することで、これを実現できます。その後、受信側メールサーバーは、サードパーティーから送信されたメールをドメイン内でアドレスを使用する権限が付与されたユーザーから送信されたメールであるかのように検証を行うことができます。

DMARC を使用した仕組みの構築

上記の DMARC アライメントチェックは、SPF、DKIM、DMARC すべてが連携し、ドメインの信頼性と受信トレイへの E メールの配信を向上させる方法です。DMARC は、受信者に表示される送信元アドレスが SPF または DKIM のいずれかによって認証されることを必須とすることでこれを実現します。

- 説明したとおり SPF または DKIM チェックのいずれかまたは両方で適格性が認められた場合、メッセージは DMARC で適格性が認められます。

- 説明したとおり SPF または DKIM チェックの両方で適格性が認められないと、メッセージは DMARC で不適格と見なされます。

したがって、DMARC が送信メールを認証する可能性を最大限に向上するには、SPF と DKIM の両方が必要であり、これら 3 つをすべて活用することで、送信ドメインが完全に保護されることが保証されます。

DMARC を使用する場合、ユーザー設定のポリシーを使用して、DMARC 認証に失敗した E メールの処理方法を E メールサーバーに指示することもできます。これについては、次のセクション「the section called “ドメインでの DMARC ポリシーのセットアップ”」で説明します。このセクションには、送信する E メールが SPF と DKIM の両方を介して DMARC 認証プロトコルに準拠するように SES ドメインを設定する方法に関する情報が提供されています。

ドメインでの DMARC ポリシーのセットアップ

DMARC をセットアップするには、ドメインの DNS 設定を変更する必要があります。ドメインの DNS 設定に、ドメインの DMARC 設定を指定する TXT レコードが含まれている必要があります。DNS 設定に TXT レコードを追加する手順は、DNS またはホスティングプロバイダーによって異なります。Route 53 を使用する場合、Amazon Route 53 デベロッパーガイドの「レコードで作業」を参照してください。別のプロバイダーを使用する場合、DNS 設定についてはプロバイダーのドキュメントを参照してください。

作成する TXT レコードの名前は、_dmarc.*example.com*の必要があります。ここで、*example.com*はお客様のドメインです。TXT レコードの値には、ドメインに適用される DMARC ポリシーが含まれています。以下に、DMARC ポリシーを含む TXT レコードの例を示します。

名前	タイプ	値
_dmarc.example.com	TXT	"v=DMARC1;p=quarantine;rua=mailto:my_dmarc_report@example.com"

上記の DMARC ポリシー例では、このポリシーは E メールプロバイダーに以下を実行するように指示します。

- 認証に失敗したメッセージについてはすべて、ポリシーパラメータ p=quarantine で指定されているとおり、スパムフォルダに送信します。その他のオプションには、p=none を使用して何もしない、または p=reject を使用してメッセージを完全に拒否する、などがあります。
  - 次のセクションでは、これら 3 つのポリシー設定の使用方法と、使用すべきケースについて説明します。適切でない設定を適切でないタイミングで使用すると、E メールが配信されなくなる可能性があります。「the section called “DMARC の実装”」を参照してください。
- レポートパラメータ rua=mailto:my_dmarc_report@example.com (rua は集約レポートのレポート URI の略) で指定されるとおり、認証に失敗したすべての E メールに関するレポートをダイジェスト (つまり、イベントごとに個別のレポートを送信するのではなく、特定の期間のデータを集約したレポート) で送信します。ポリシーはプロバイダーごとに異なりますが、通常、E メールプロバイダーは 1 日 1 回レポートを送信します。

ドメインの DMARC 設定の詳細については、DMARC ウェブサイトの「概要」を参照してください。

DMARC システムの完全な仕様については、「Internet Engineering Task Force (IETF) DMARC Draft」を参照してください。

DMARC の実装のベストプラクティス

メールフローのその他の部分が中断しないように、DMARC ポリシーの実施は段階的なアプローチで実装することをお勧めします。以下のステップに沿ったロールアウトプランを作成して実装します。まず各サブドメインでこれらの手順を実行し、最後に組織内のトップレベルのドメインで実行してから、次の手順に進みます。

1. DMARC (p=none) の実装の影響をモニタリングします。
   - まず、メール受信組織に、そのドメインを使用して受信したメッセージに関する統計を送信するように要求するサブドメインまたはドメインのシンプルなモニタリングモードレコードから始めます。モニタリングモードレコードとは、ポリシーがなし p=none に設定されている DMARC TXT レコードです。
   - DMARC を介して生成されたレポートには、これらのチェックに合格したメッセージ数とメッセージの送信元が記載されます。正当なトラフィックがどの程度カバーされているか、またはカバーされていないかを簡単に確認できます。コンテンツが変更されると、転送されたメッセージは SPF と DKIM に失敗するため、転送のマークが表示されます。送信された不正なメッセージの数と送信元も表示されます。

- このステップの目的は、次の 2 つのステップのいずれかを実装するとどのような E メールが影響を受けるかを把握し、サードパーティーまたは承認済みの送信者の SPF ポリシーまたは DKIM ポリシーへのアライメントを確保することにあります。
- 既存のドメインに最適です。

2. 外部メールシステムに、DMARC (p=quarantine) に失敗したメールを隔離するようリクエストします。
   - 正当なトラフィックのすべてまたはほとんどが SPF または DKIM のいずれかでドメインアライメントが確認されて送信されていることが確実であり、DMARC を実装することの影響を把握している場合に、隔離ポリシーを実装できます。隔離ポリシーとは、ポリシーが隔離 `p=quarantine` に設定されている DMARC TXT レコードです。これを実施することで、DMARC 受信者に、DMARC に失敗したドメインからのメッセージを、顧客の受信トレイではなく、スパムフォルダと同等のローカルの場所に配置するように要求することになります。
   - ステップ 1 で DMARC レポートを分析したドメインの移行に最適です。
3. 外部メールシステムに、DMARC (p=reject) に失敗したメールを受け取らないようリクエストします。
   - 通常、拒否ポリシーの実装が最後のステップとなります。拒否ポリシーとは、ポリシーが拒否 `p=reject` に設定されている DMARC TXT レコードです。これを実施すると、DMARC 受信者に DMARC チェックに失敗したメッセージを受け入れないように依頼することになります。つまり、これらのメッセージは、スパムフォルダや迷惑メールフォルダに隔離されることもなく、完全に拒否されます。
   - 拒否ポリシーを使用すると、拒否によって SMTP バウンスが発生するため、DMARC ポリシーに失敗したメッセージを正確に把握できます。隔離の場合、集約データは、SPF チェック、DKIM チェック、DMARC チェックで適格または不適格となった E メールの割合に関する情報を提供します。
   - 以前の 2 つのステップを実施した後の新しいドメインまたは既存のドメインに最適です。

SPF を介した DMARC への準拠

E メールが SPF に基づいて DMARC に準拠するためには、次の両方の条件を満たすことが求められています。

- メッセージは、カスタム MAIL FROM ドメインの DNS 設定に発行した有効な SPF (タイプは TXT) レコードに基づいて SPF チェックで適格性が認められる必要があります。

- E メールヘッダーの送信元アドレスのドメインは、MAIL FROM アドレスで指定されているドメイン、またはサブドメインとのアライメントが認められる (一致する) 必要があります。SES との SPF アラインメントを確保するために、ドメインの DMARC ポリシーで strict の SPF ポリシー (aspf=s) を指定することは避けます。

これらの要件に準拠するためには、次のステップを実行します。

- the section called “カスタムMAILFROMドメインの使用”の手順を実行して、カスタム MAIL FROM ドメインを設定します。
- 送信元ドメインが SPF に relaxed ポリシーを使用していることを確認します。ドメインのポリシーアラインメントを変更していない場合は、デフォルトで SES と同様に relaxed のポリシーが使用されます。

> **Note**
>
> コマンドラインで以下のコマンドを入力して、*example.com*をドメインで置き換えることで、SPF での DMARC アラインメントを選択できます。
>
> ```
> dig TXT _dmarc.example.com
> ```
>
> このコマンドの出力の [Non-authoritative answer] から、v=DMARC1で始まるレコードを探します。このレコードに文字列aspf=rが含まれるか、またはaspf文字列がまったく存在しない場合、ドメインは SPF に relaxed アラインメントを使用します。レコードに文字列aspf=sが含まれる場合、ドメインは SPF に strict アラインメントを使用します。システム管理者は、ドメインの DNS 設定の DMARC TXT レコードからこのタグを削除する必要があります。
> 別の方法として、dmarcian ウェブサイトの DMARC Inspector や MxToolBox ウェブサイトの DMARC Check ツールなどのウェブベースの DMARC ルックアップツールを使用して、ドメインのポリシーの SPF へのアライメントを判断することもできます。

DKIM を介した DMARC への準拠

E メールが DKIM に基づいて DMARC に準拠するためには、次の両方の条件を満たすことが求められています。

- メッセージには有効な DKIM 署名が必要であり、DKIM チェックで適格性が認められる必要があります。

- DKIM 署名で指定されたドメインが、送信元アドレスのドメインと一致する必要があります。ドメインの DMARC ポリシーで DKIM に strict アラインメントが指定されている場合は、これらのドメインは完全に一致する必要があります (SES はデフォルトで strict の DKIM ポリシーを使用します)。

これらの要件に準拠するためには、次のステップを実行します。

- the section called “簡単 DKIM”の手順を実行して Easy DKIM を設定します。Easy DKIM を使用すると、Amazon SES は自動的に E メールに署名します。

> **Note**
>
> Easy DKIM を使用せずに、メッセージに手動で署名することもできます。ただし、Amazon SES は構築した DKIM 署名を検証しないため、この選択は慎重に行ってください。そのため、Easy DKIM を使用することを強くお勧めします。

- DKIM 署名で指定されたドメインが、送信元アドレスのドメインと一致していることを確認してください。または、送信元アドレスのドメインのサブドメインから送信する場合は、DMARC ポリシーが relaxed アライメントに設定されていることを確認します。

> **Note**
>
> コマンドラインで以下のコマンドを入力して、*example.com*をドメインで置き換えることで、DKIM での DMARC アラインメントを選択できます。
>
> ```
> dig TXT _dmarc.example.com
> ```
>
> このコマンドの出力の [Non-authoritative answer] から、v=DMARC1で始まるレコードを探します。このレコードに文字列adkim=rが含まれるか、またはadkim文字列がまったく存在しない場合、ドメインは DKIM に relaxed アラインメントを使用します。レコードに文字列adkim=sが含まれる場合、ドメインは DKIM に strict アラインメントを使用します。システム管理者は、ドメインの DNS 設定の DMARC TXT レコードからこのタグを削除する必要があります。
> 別の方法として、dmarcian ウェブサイトの DMARC Inspector や MxToolBox ウェブサイトの DMARC Check ツールなどのウェブベースの DMARC ルックアップツールを使用して、ドメインのポリシーの DKIM へのアライメントを判断することもできます。

Amazon SES での BIMI の使用

Brand Indicators for Message Identification (BIMI) は、対応しているメールクライアントの E メールの受信トレイにおいて、ブランドの認証済み E メールメッセージの横にブランドのロゴを表示できるようにするメール仕様です。

BIMI は認証に直接関係するメール仕様ですが、すべての E メールに対して DMARC 認証への準拠を要求するため、スタンドアロンの E メール認証プロトコルではありません。

BIMI には DMARC が必要であり、DMARC にはドメインの SPF レコードまたは DKIM レコードのアラインメントが必要ですが、SPF レコードと DKIM レコードの両方を含めるのが最善です。セキュリティの強化になるとともに、E メールサービスプロバイダー (ESP) によっては BIMI を使用する際に両方を要求する場合があるためです。次のセクションでは、Amazon SES に BIMI を実装する手順を示します。

SES での BIMI の設定

所有するメールドメイン (SES ではカスタム MAIL FROM ドメインと呼ばれます) に BIMI を設定できます。設定すると、BIMI をサポートする E メールクライアントにおいて、そのドメインから送信されたすべてのメッセージに BIMI ロゴが表示されます。

E メールに BIMI ロゴを表示するには、SES 内でいくつかの前提条件を満たす必要があります。次の手順では、これらの前提条件を一般化するとともに、これらのトピックを詳細に説明する専用のセクションを参照として示します。ここでは、BIMI に固有の手順と、SES で BIMI を設定するために必要な事項について詳しく説明します。

カスタム MAIL FROM ドメインに BIMI を設定するには

1. SES でカスタム MAIL FROM ドメインを設定し、このドメインに SPF (タイプ TXT) レコードと MX レコードの両方を発行する必要があります。カスタム MAIL FROM ドメインを持っていないか、これを BIMI ロゴ用に新しく作成する場合は、「the section called “カスタムMAILFROMドメインの使用”」を参照してください。
2. Easy DKIM を使用してドメインを設定します。「the section called “簡単 DKIM”」を参照してください。
3. 以下の 2 つの例のいずれかと同様に、BIMI に必要な以下の強制ポリシーの詳細を含む TXT レコードを DNS プロバイダーに発行して、ドメインに DMARC を設定します。

名前	タイプ	値
_dmarc.example.com	TXT	v=DMARC1;p=quarantine;pct=100;rua=mailto:dmarcreports@example.com
_dmarc.example.com	TXT	v=DMARC1;p=reject;rua=mailto:dmarcreports@example.com

上の BIMI に必要な DMARC ポリシーの例では、以下のとおりとします。

- *example.com* は、ドメイン名またはサブドメイン名に置き換える必要があります。
- p= の値は以下のいずれかです。
  - 表示されているとおりに pct 値を 100 に設定した quarantine
  - 表示されているとおりに reject。
- サブドメインから送信する場合、BIMI は親ドメインにもこの強制ポリシーを持つことを要求します。サブドメインは親ドメインのポリシーに従います。ただし、親ドメインに発行した内容に加えてサブドメインにも DMARC レコードを追加する場合、BIMI の対象となるには、サブドメインにも同じ強制ポリシーが必要です。
- ドメインに DMARC ポリシーを設定したことがない場合は、「the section called “DMARC による E メールの認証”」を参照し、表示されているとおりに BIMI 固有の DMARC ポリシー値のみを使用してください。

4. BIMI ロゴをスケーラブルベクターグラフィックス (SVG) `.svg` ファイルとして作成します。BIMI に必要な特定の SVG プロファイルは、SVG Portable/Secure (SVG P/S) として定義します。ロゴを E メールクライアントに表示するには、これらの仕様に正確に準拠する必要があります。BIMI Group の SVG ロゴファイルの作成に関するガイダンスと、推奨される SVG 変換ツールを参照してください。
5. (オプション) Verified Mark Certificate (VMC) を取得します。Gmail や Apple などの一部の ESP では、BIMI ロゴの商標とコンテンツを所有していることの証拠として VMC を要求します。これはドメインに BIMI を実装するための要件ではありませんが、メール送信先の ESP が VMC 準拠を強制している場合、BIMI ロゴは E メールクライアントに表示されません。ロゴの VMC を取得するには、BIMI Group の加盟認証機関に関するリファレンスを参照してください。
6. BIMI ロゴの SVG ファイルを、ユーザーがアクセスできるサーバーでホストし、HTTPS 経由で一般に公開します。例えば、Amazon S3 バケットにファイルをアップロードできます。

7. ロゴの URL を含む BIMI DNS レコードを作成して発行します。BIMI をサポートする ESP は、DMARC レコードをチェックするときに、ロゴの .svg ファイルの URL が含まれている BIMI レコードと、VMC の .pem ファイルの URL (設定されている場合) も検索します。レコードが一致すると、BIMI ロゴが表示されます。

   ドメインに BIMI を設定します。そのために、以下に示す値を使用して DNS プロバイダーに TXT レコードを発行します。最初の例はドメインからの送信、2番目の例はサブドメインからの送信を示しています。

名前	タイプ	値
default._bimi.example.com	TXT	v=BIMI1;l=https://myhostingserver.com/images/logo.svg;a=https://myhostingserver.com/certificate/vmc_2023-01-01.pem
default._bimi.marketing.example.com		

   上の BIMI レコードの例では、以下のとおりとします。

   - 名前の値は、default._bimi. を *example.com* または *marketing.example.com* のサブドメインとして文字どおり指定する必要があり、自分のドメイン名またはサブドメイン名に置き換えます。
   - v= 値は BIMI レコードのバージョンです。
   - l= 値は、ロゴの URL であり、画像の .svg ファイルを指します。
   - a= 値は、機関の URL であり、証明書の .pem ファイルを指します。

   BIMI Group の BIMI Inspector などのツールを使用して BIMI レコードを検証できます。

このプロセスの最後のステップとして、BIMI ロゴの掲載をサポートする ESP への定期的な送信パターンを設定します。ドメインは、定期的な配信頻度を持ち、送信先の ESP から高い評価を得ている必要があります。評価や配信頻度の実績がない場合、BIMI ロゴが ESP に掲載されるまでに時間がかかることがあります。

BIMI に関する詳細情報やリソースについては、BIMI Group 組織で参照できます。

Amazon SES のイベント通知の設定

Amazon SES を使用して電子メールを送信するには、バウンスと苦情を管理するためのシステムが必要です。Amazon SES は、通知メールを送信する、Amazon SNS トピックを通知する、送信イベントを公開する、の 3 つのいずれかの方法でバウンスや苦情イベントを通知します。このセクションでは、E メールまたは Amazon SNS トピックに通知することによって、特定の種類の通知を送信するように Amazon SES を設定する方法について説明します。送信イベントを公開する方法の詳細については、「Amazon SES イベント発行を使用して E メール送信をモニタリングする」を参照してください。

Amazon SES コンソールまたは Amazon SES API を使用して通知をセットアップできます。

トピック

重要な考慮事項

通知を送信するように Amazon SES を設定する際に考慮すべきいくつかの重要な点があります。

- E メールと Amazon SNS 通知は、個々の ID (E メールの送信に使用する検証済みの電子メールアドレスまたはドメイン) に適用されます。ID の通知を有効にすると、Amazon SES はその ID から送信された E メールにのみ通知を送信し、通知を設定したAWS地域でのみ通知を送信します。
- バウンスや苦情の通知を受け取る方法を有効にする必要があります。バウンスや苦情を生成したドメインや電子メールアドレス、または Amazon SNS トピックに通知を送信できます。イベント発行を使用すると、いくつかの異なるタイプのイベントに関する通知 (バウンス、苦情、配信) を Amazon SNS トピックまたは Firehose ストリームに送信することもできます。

  バウンスや苦情の通知を受け取るこれらの方法の 1 つを設定しない場合は、E メールのフィードバック転送を無効にしていても、Amazon SES はバウンスや苦情のイベントの原因となった E メールの Return-Path アドレス (または Return-Path アドレスを指定しなかった場合はソースアドレス) にバウンスや苦情の通知を自動的に転送します。

  E メールのフィードバック転送を無効にし、イベント発行を有効にする場合は、送信するすべての E メールにイベント発行ルールを含む設定セットを適用する必要があります。この状況で、設定セットを使用しない場合は、Amazon SES はバウンスや苦情のイベントの原因となった E メールのリターンパス アドレスまたは出典アドレスにバウンスや苦情の通知を自動的に転送します。

- 1 つ以上の方法 (E メール通知を送信する、イベントの送信を使用するなど) を使用してバウンスや苦情を送信するように Amazon SES を設定する場合は、同じイベントに対して 1 つ以上の通知を受け取る場合があります。

E メールを介したAmazon SES に関する通知の受信

Amazon SES は、ユーザーがバウンスや苦情を受信すると、E メールのフィードバック転送というプロセスを使用して、ユーザーに E メールを送信します。

Amazon SES を使用して E メールを送信するには、次のいずれかの方法を使用して、バウンスや苦情の通知を送信するように設定する必要があります。

- E メールのフィードバック転送を有効にする。このタイプの通知を設定する手順は、このセクションに含まれています。
- Amazon SNS トピックに通知を送信する。詳細については、「Amazon を使用した Amazon SES 通知の受信 SNS」を参照してください。
- イベント通知を発行する。詳細については、「Amazon SES イベント発行を使用して E メール送信をモニタリングする」を参照してください。

> **Important**
>
> 通知についてのいくつかの重要なポイントについては、「Amazon SES のイベント通知の設定」を参照してください。

トピック

- E メールのフィードバック転送を有効にする
- E メールのフィードバック転送を無効にする
- E メールのフィードバック転送先

E メールのフィードバック転送を有効にする

E メールのフィードバック転送はデフォルトで有効です。以前に無効にしている場合、このセクションの以下の手順に従って有効にできます。

Amazon SES コンソールを使用して E メールによるバウンスや苦情の転送を有効にする

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインの [設定] で、[検証済み ID] を選択します。
3. 確認済みの E メールアドレスまたはドメインで、バウンスと苦情の通知を設定する E メールアドレスまたはドメインを選択します。
4. 詳細ペインで、[Notifications] セクションを展開します。
5. [Edit Configuration] を選択します。
6. [E メール Feedback Forwarding] で、[Enabled] を選択します。

> **Note**
>
> このページで行った変更は、反映されるまでに数分かかる場合があります。

また、SetIdentityFeedbackForwardingEnabled API オペレーションを使用して、バウンスや苦情の通知を有効にできます。

E メールのフィードバック転送を無効にする

バウンスや苦情の通知を提供する別の方法を設定する場合は、バウンスや苦情のイベントが発生したときに複数の通知を受け取らないように、E メールのフィードバック転送を無効にすることができます。

Amazon SES コンソールを使用して E メールを介したバウンスや苦情の転送を無効にする

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインの [設定] で、[検証済み ID] を選択します。
3. 確認済みの E メールアドレスまたはドメインで、バウンスと苦情の通知を設定する E メールアドレスまたはドメインを選択します。
4. 詳細ペインで、[Notifications] セクションを展開します。
5. [Edit Configuration] を選択します。
6. [E メール Feedback Forwarding] で、[Disabled] を選択します。

> **Note**
>
> Amazon SES を介して E メールを送信するには、バウンスや苦情の通知を受け取る 1 つの方法を設定する必要があります。E メールのフィードバック転送を無効にする場合は、Amazon SNS が送信する通知を有効にするか、イベント発行を使用して、バウンスイベントと苦情イベントを Amazon SNS トピックまたは Firehose ストリームに発行する必要があります。イベント発行を使用する場合は、送信する各 E メールにイベント発行ルールを含む設定セットも適用する必要があります。バウンスや苦情の通知を受け取る方法を設定しない場合は、Amazon SES はバウンスや苦情のイベントの原因となったメッセージのリターンパス フィールド (またはリターンパス アドレスを指定しなかった場合は出典フィールド) のアドレスに、E メールによるフィードバック通知を自動的に転送します。この場合、E メールのフィードバック通知を無効にしても、Amazon SES はバウンス通知や苦情の通知を転送します。

7. [Save Config] を選択して通知設定を保存します。

> **Note**
>
> このページで行った変更は、反映されるまでに数分かかる場合があります。

また、SetIdentityFeedbackForwardingEnabled API オペレーションを使用して、バウンスや苦情の通知を無効にできます。

E メールのフィードバック転送先

E メールで通知を受け取る場合、Amazon SES は`From`ヘッダーを書き換えて通知を送信します。Amazon SES が通知を転送するアドレスは、元のメッセージの送信方法によって異なります。

SMTP インターフェイスを使用してメッセージを送信すると、通知は以下の規則に従って配信されます。

- SMTP DATA セクションで `Return-Path` ヘッダーを指定すると、通知はそのアドレスに送信されます。
- その他の場合は、MAIL FROM コマンドを発行したときに指定したアドレスに通知が送信されます。

SendEmailAPI オペレーションを使用してメッセージを送信すると、通知は以下の規則に従って配信されます。

- SendEmailAPI を呼び出す際にオプションのReturnPath パラメータを指定すると、通知はそのアドレスに送信されます。
- 指定しない場合、Sourceの必須SendEmailパラメータであるで指定されたアドレスに通知が送信されます。

SendRawEmailAPI オペレーションを使用してメッセージを送信すると、通知は以下の規則に従って配信されます。

- raw メッセージで Return-Path ヘッダーを指定すると、通知はそのアドレスに送信されます。
- その他の場合は、SendRawEmail API を呼び出す際に Source パラメータを指定すると、通知はそのアドレスに送信されます。
- その他の場合、raw メッセージのFromヘッダーにおいて指定されたアドレスに通知が送信されます。

> **Note**
>
> E メールでReturn-Pathアドレスを指定すると、通知はそのアドレスに送信されます。ただし、受信者が受信するメッセージのバージョンには、 匿名化されたEメールアドレス (a0b1c2d3e4f5a6b7-c8d9e0f1-a2b3-c4d5-e6f7-a8b9c0d1e2f3-000000@amazonses.com) を含むReturn-Pathヘッダー含まれています。この匿名化は、E メールの送信方法にかかわらず実行されます。

Amazon を使用した Amazon SES通知の受信 SNS

バウンスや苦情を受け取ったとき、または E メールが配信されたときに Amazon SNSトピックに通知するSESように Amazon を設定できます。Amazon SNS通知は JavaScript Object Notation (JSON) 形式で、プログラムで処理できます。

Amazon を使用して E メールを送信するにはSES、次のいずれかの方法を使用してバウンスや苦情の通知を送信するように設定する必要があります。

- Amazon SNSトピックに通知を送信する。このタイプの通知を設定する手順は、このセクションに含まれています。

- E メールのフィードバック転送を有効にする。詳細については、「E メールを介したAmazon SESに関する通知の受信」を参照してください。
- イベント通知を発行する。詳細については、「Amazon SES イベント発行を使用して E メール送信をモニタリングする」を参照してください。

> **Important**
>
> 通知に関する重要な情報については、「Amazon SES のイベント通知の設定」を参照してください。

トピック

- Amazon の Amazon SNS通知の設定 SES
- Amazon の Amazon SNS通知コンテンツ SES
- Amazon の Amazon SNS通知の例 SES

Amazon の Amazon SNS通知の設定 SES

Amazon SESは、Amazon Simple Notification Service (Amazon SNS) を通じてバウンス、苦情、配信を通知できます。

通知は、Amazon SESコンソールで、または Amazon SES を使用して設定できますAPI。

このセクションのトピック:

- 前提条件
- Amazon SESコンソールを使用した通知の設定
- Amazon を使用した通知の設定 SES API
- フィードバック通知のトラブルシューティング

前提条件

Amazon で Amazon SNS通知を設定する前に、次のステップを実行しますSES。

1. Amazon でトピックを作成しますSNS。詳細については、Amazon Simple Notification Service デベロッパーガイドの「トピックの作成」を参照してください。

> **Important**
>
> Amazon を使用してトピックを作成する場合SNS、タイプ では、標準 のみを選択します (SES はFIFOタイプトピックをサポートしていません)。

新しいSNSトピックを作成するか、既存のトピックを選択するかにかかわらず、トピックに通知を発行SESするには へのアクセス権を に付与する必要があります。

トピックに通知を発行するアクセスSES許可を Amazon に付与するには、SNSコンソールのトピックの編集画面でアクセスポリシーを展開し、JSONエディタで次のアクセス許可ポリシーを追加します。

```
{
  "Version": "2012-10-17",
  "Id": "notification-policy",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ses.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:topic_region:111122223333:topic_name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "111122223333",
          "AWS:SourceArn":
 "arn:aws:ses:topic_region:111122223333:identity/identity_name"
        }
      }
    }
  ]
}
```

上のポリシー例に、以下の変更を加えます。

- を、SNSトピックを作成した AWS リージョン*topic_region*に置き換えます。
- *111122223333* を AWS アカウント ID に置き換えます。

- をSNSトピックの名前*topic_name*に置き換えます。
- を、SNSトピックにサブスクライブしている検証済み ID (E メールアドレスまたはドメイン) *identity_name*に置き換えます。

2. 少なくとも 1 つのエンドポイントをトピックにサブスクライブします。たとえば、テキストメッセージで通知を受信する場合は、 トピックにSMSエンドポイント (つまり、携帯電話番号) をサブスクライブします。E メールで通知を受信するには、E メールエンドポイント (E メールアドレス) をトピックにサブスクライブします。

   詳細については、Amazon Simple Notification Service デベロッパーガイドの「作業スタート」を参照してください。

3. (オプション) Amazon SNSトピックがサーバー側の暗号化に AWS Key Management Service (AWS KMS) を使用する場合は、 AWS KMS キーポリシーにアクセス許可を追加する必要があります。アクセス許可を追加するには、次のポリシーを AWS KMS キーポリシーにアタッチします。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSESToUseKMSKey",
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon SESコンソールを使用した通知の設定

Amazon SESコンソールを使用して通知を設定するには

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

2. ナビゲーションペインの [設定] で、[ID] を選択します。
3. [Identities] (ID) コンテナで、この ID の結果から送信されたメッセージがバウンス、苦情、または配信になったときのフィードバック通知を受信する検証済み ID を選択します。

> **Important**
>
> 確認済みドメイン通知設定は、同様に確認済みの E メールアドレスを除き、そのドメインの E メールアドレスから送信されるすべてのメールに適用されます。

4. 選択した検証済み ID の詳細画面で、[Notifications] (通知) タブを選択し、[Feedback notifications] (フィードバック通知) コンテナの [Edit] (編集) を選択します。
5. 通知を受信する各フィードバックタイプのSNSトピックリストボックスを展開し、所有するSNSトピック、トピックなしSNS、またはSNS所有しないトピックを選択します。

   - SNS 所有していないトピックを選択した場合、SNSトピックARNフィールドが表示され、代理送信者がARN共有したSNSトピックを入力する必要があります。(代理送信者のみがSNSトピックを所有しているため、これらの通知を受け取ります。 代理送信の詳細については、「」を参照してください送信承認の概要。)

> **Important**
>
> バウンス、苦情、配信の通知に使用する Amazon SNSトピックは、Amazon を使用する AWS リージョン トピックと同じ にある必要がありますSES。
> さらに、通知を受け取るには、1 つ以上のエンドポイントをトピックにサブスクライブしている必要があります。たとえば、E メールアドレスに通知を送信する場合は、E メールエンドポイントをそのトピックにサブスクライブ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「作業スタート」を参照してください。

6. (オプション) トピック通知に元の E メールのヘッダーを含める場合は、各フィードバックタイプのSNSトピック名のすぐ下にある「元の E メールヘッダーを含める」チェックボックスをオンにします。このオプションは、関連付けられた通知タイプに Amazon SNSトピックを割り当てた場合にのみ使用できます。元の E メールヘッダーの内容については、通知の内容の `mail` オブジェクトを参照してください。
7. [Save changes] (変更の保存) をクリックします。通知設定に対する変更は、反映されるまでに数分かかる場合があります。

8. （オプション) バウンスと苦情の両方に対して Amazon SNSトピック通知を選択した場合は、E メール通知を完全に無効にして、E メールと 通知を介して二重SNS通知を受信しないようにできます。バウンスや苦情に関する E メール通知を無効にするには、[Email Feedback Forwarding] (E メールのフィードバック転送) コンテナの検証済み ID の詳細画面にある [Notifications] (通知) タブで、[Edit] (編集) を選択し、[Enabled] (有効) ボックスのチェックを外して、[Save changes] (変更を保存) を選択します。

設定を構成すると、Amazon SNSトピックへのバウンス、苦情、配信の通知を受信し始めます。これらの通知は JavaScript Object Notation (JSON) 形式で、「」で説明されている構造に従います通知の内容。

バウンス、苦情、配信の通知には、標準の Amazon SNS料金が課金されます。詳細については、Amazon のSNS料金ページを参照してください。

> **Note**
>
> トピックが削除されたか、 に発行するアクセス許可 AWS アカウント がなくなったためにAmazon SNSトピックへの発行が失敗した場合、Amazon はバウンスまたは苦情 (配信ではなく、配信通知の場合、 はトピック設定を削除SESしません) 用に設定されている場合、そのSNSトピックの設定SESを削除します。さらに、Amazon はアイデンティティのバウンスと苦情に関する E メール通知SESを再度有効にし、変更の通知を E メールで受け取ります。トピックを使用するように複数の ID が設定されている場合、各 ID でトピックへの発行に失敗すると、各 ID のトピック設定が変更されます。

Amazon を使用した通知の設定 SES API

Amazon を使用して、バウンス、苦情、配信の通知を設定することもできますSESAPI。次のオペレーションを使用して通知を設定します。

- SetIdentityNotificationTopic
- SetIdentityFeedbackForwardingEnabled
- GetIdentityNotificationAttributes
- SetIdentityHeadersInNotificationsEnabled

これらのAPIアクションを使用して、通知用にカスタマイズされたフロントエンドアプリケーションを記述できます。通知に関連するAPIアクションの詳細については、Amazon Simple Email Service APIリファレンスを参照してください。

フィードバック通知のトラブルシューティング

通知が送られてこない

通知を受け取っていない場合は、通知が送信されるトピックにエンドポイントをサブスクライブしていることを確認します。E メールエンドポイントをトピックにサブスクライブすると、サブスクリプションの確認を求める E メールが届きます。E メール通知の受信を開始するには、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「作業スタート」を参照してください。

`InvalidParameterValue` トピックを選択する際にエラーが発生する

エラーが発生したことを示すInvalidParameterValueエラーが表示された場合は、Amazon SNS トピックで を使用して暗号化されているかどうかを確認します AWS KMS。その場合は、 AWS KMS キーのポリシーを変更する必要があります。サンプルポリシーについては、「前提条件」を参照してください。

Amazon の Amazon SNS通知コンテンツ SES

バウンス、苦情、配信の通知は、Amazon Simple Notification Service (Amazon SNS) トピックに JavaScript Object Notation (JSON) 形式で発行されます。最上位JSONオブジェクトには、`notificationType`文字列、`mail`オブジェクト、および`bounce`オブジェクト、`complaint`オブジェクト、または `delivery` オブジェクトが含まれます。

オブジェクトのタイプごとの詳細については以下のセクションを参照してください。

- 最上位JSONオブジェクト
- `mail` オブジェクト
- `bounce` オブジェクト
- `complaint` オブジェクト
- `delivery` オブジェクト

以下は、Amazon の Amazon SNS通知の内容に関する重要な注意事項ですSES。

- 特定の通知タイプでは、複数の受信者に対して 1 つの Amazon SNS 通知を受け取ることも、受信者ごとに 1 つの Amazon SNS通知を受け取ることもできます。コードは Amazon SNS通知を解析

し、両方のケースを処理できる必要があります。Amazon SES は、Amazon を介して送信される通知の順序付けやバッチ処理を保証しませんSNS。ただし、さまざまな Amazon SNS通知タイプ (バウンスや苦情など) は 1 つの通知に結合されません。

- 1 人の受信者に対して複数のタイプの Amazon SNS通知を受け取る場合があります。たとえば、受信メールサーバーは、E メールを受理した場合でも (配信の通知をトリガーします)、そのメールの処理後に、そのメールは実際にはバウンスであると判定する場合があります (バウンスの通知をトリガーします)。ただし、通知のタイプが異なるため、これらは常に個別に通知されます。
- Amazon SESは、通知にフィールドを追加する権利を留保します。そのため、これらの通知を解析するアプリケーションには、不明なフィールドを処理できるだけの十分な柔軟性が必要です。
- Amazon は、E メールを送信するときにメッセージのヘッダーをSES上書きします。`mail` オブジェクトの `headers` および `commonHeaders` フィールドから元のメッセージのヘッダーを取得できます。

最上位JSONオブジェクト

Amazon SES通知の最上位JSONオブジェクトには、次のフィールドが含まれます。

フィールド名	説明
`notificationType`	JSON オブジェクトによって表される通知のタイプを保持する文字列。想定される値は 、`Bounce`、`Complaint` 、および `Delivery` です。 イベント発行をセットアップする場合、このフィールドの名前は `eventType` です。
`mail`	通知が関係する元のメールに関する情報を含むJSONオブジェクト。詳細については、「Mailオブジェクト」を参照してください。
`bounce`	このフィールドは、`notificationType` が Bounceであり、バウンスに関する情報を保持するJSONオブジェクトが含まれている場合にのみ存在します。詳細については、「Bounceオブジェクト」を参照してください。

フィールド名	説明
complaint	このフィールドは、 notificationType が Complaint であり、苦情に関する情報を保持するJSONオブジェクトが含まれている場合にのみ存在します。詳細については、「苦情のオブジェクト」を参照してください。
delivery	このフィールドは、 notificationType が Deliveryであり、配信に関する情報を保持するJSONオブジェクトが含まれている場合にのみ存在します。詳細については、「配信オブジェクト」を参照してください。

Mail オブジェクト

バウンス、苦情、または配信の通知にはそれぞれ、mail オブジェクト内の元の E メールについての情報が含まれます。JSON オブジェクトに関する情報を含む mail オブジェクトには、次のフィールドがあります。

フィールド名	説明
timestamp	元のメッセージが送信された時刻 (ISO8601 形式) 。
messageId	Amazon がメッセージにSES割り当てた一意の ID。Amazon は、メッセージを送信したときにこの値をSES返しました。 **Note** このメッセージ ID は Amazon によって割り当てられましたSES。元の E メールのメッセージ ID は、mail オブジェクトの headers フィールドにあります。

フィールド名	説明
source	元のメッセージが送信された E メールアドレス (エンベロープMAILFROMアドレス) 。
sourceArn	E メールの送信に使用された ID の Amazon リソースネーム (ARN) 。送信承認の場合、sourceArn は、ID 所有者が代理送信者が E メールの送信に使用することを承認した ID ARNの です。送信承認の詳細については、「E メールの認証方法」を参照してください。
sourceIp	Amazon への E メール送信リクエストを実行したクライアントの発信元パブリック IP アドレスSES。
sendingAccountId	E メールの送信に使用されたアカウントの AWS アカウント ID。送信承認の場合、sendingAccountId は代理送信者のアカウント ID です。
callerIdentity	E メールを送信した Amazon SESユーザーの IAM ID。
destination	元のメールの受取人の E メールアドレスのリスト。
headersTruncated	元の E メールからヘッダーを含めるように通知設定を構成した場合にのみ、このオブジェクトが表示されます。 ヘッダーが通知で切り詰められるかどうかを示します。元のメッセージのヘッダーのサイズが SES 10 KB 以上の場合、Amazon は通知のヘッダーを切り捨てます。指定できる値は true および false です。

フィールド名	説明
`headers`	元の E メールからヘッダーを含めるように通知設定を構成した場合にのみ、このオブジェクトが表示されます。 E メールの元のヘッダーの一覧。リスト内の各ヘッダーには、`name` フィールドと `value` フィールドがあります。 **Note** `headers` オブジェクト内のメッセージ ID は、Amazon に渡した元のメッセージからのものですSES。AmazonがSESその後メッセージに割り当てたメッセージ ID は、 `mail` オブジェクトの `messageId` フィールドにあります。

フィールド名	説明
commonHeaders	元の E メールからヘッダーを含めるように通知設定を構成した場合にのみ、このオブジェクトが表示されます。 元の E メールからの一般的な E メールヘッダーに関する情報 (送信元、宛先、件名フィールドなど) が含まれます。このオブジェクト内では、各ヘッダーはキーとなります。送信元フィールドと宛先フィールドは、複数の値を含むことができる配列で表されます。 **Note** イベントの場合、 commonHeaders フィールド内のメッセージ ID は、Amazon がSESその後メールオブジェクトの messageId フィールドのメッセージに割り当てたメッセージ ID です。通知には、元の E メールのメッセージ ID が含まれます。

以下は、元の E メールヘッダーを含む mail オブジェクトの例です。この通知タイプが元の E メールヘッダーを含めるように設定されていない場合は、mail オブジェクトに headersTruncated、headers および commonHeaders フィールドが含まれません。

```
{
   "timestamp":"2018-10-08T14:05:45 +0000",
   "messageId":"000001378603177f-7a5433e7-8edb-42ae-af10-f0181f34d6ee-000000",
   "source":"sender@example.com",
   "sourceArn": "arn:aws:ses:us-east-1:888888888888:identity/example.com",
   "sourceIp": "127.0.3.0",
   "sendingAccountId":"123456789012",
   "destination":[
      "recipient@example.com"
   ],
   "headersTruncated":false,
```

```
   "headers":[
      {
         "name":"From",
         "value":"\"Sender Name\" <sender@example.com>"
      },
      {
         "name":"To",
         "value":"\"Recipient Name\" <recipient@example.com>"
      },
      {
         "name":"Message-ID",
         "value":"custom-message-ID"
      },
      {
         "name":"Subject",
         "value":"Hello"
      },
      {
         "name":"Content-Type",
         "value":"text/plain; charset=\"UTF-8\""
      },
      {
         "name":"Content-Transfer-Encoding",
         "value":"base64"
      },
      {
         "name":"Date",
         "value":"Mon, 08 Oct 2018 14:05:45 +0000"
      }
   ],
   "commonHeaders":{
      "from":[
         "Sender Name <sender@example.com>"
      ],
      "date":"Mon, 08 Oct 2018 14:05:45 +0000",
      "to":[
         "Recipient Name <recipient@example.com>"
      ],
      "messageId":" custom-message-ID",
      "subject":"Message sent using Amazon SES"
   }
}
```

Bounce オブジェクト

バウンスに関する情報を含む JSON オブジェクトには、次のフィールドが含まれます。

フィールド名	説明
`bounceType`	Amazon によって決定されるバウンスのタイプSES。詳細については、「バウンスのタイプ」を参照してください。
`bounceSubType`	Amazon によって決定されるバウンスのサブタイプSES。詳細については、「バウンスのタイプ」を参照してください。
`bouncedRecipients`	バウンスとなった元のメールの受取人についての情報を含むリスト。詳細については、「バウンスとなった受取人」を参照してください。
`timestamp`	バウンスが送信された日時 (ISO8601 形式) 。これは、通知が によって送信された時刻でありISP、Amazon によって受信された時刻ではないことに注意してくださいSES。
`feedbackId`	バウンスの一意の ID。

Amazon SESがリモート Message Transfer Authority (MTA) に連絡できた場合は、次のフィールドも表示されます。

フィールド名	説明
`remoteMtaIp`	Amazon MTAが E メールの配信SESを試みたの IP アドレス。

配信ステータス通知 (DSN) がバウンスにアタッチされている場合、次のフィールドも表示されます。

フィールド名	説明
reportingMTA	からの Reporting-MTA フィールドの値DSN。これは、 で説明MTAされている配信、リレー、またはゲートウェイオペレーションを実行しようとした の値ですDSN。

以下は、bounce オブジェクトの例です。

```
{
   "bounceType":"Permanent",
   "bounceSubType": "General",
   "bouncedRecipients":[
      {
         "status":"5.0.0",
         "action":"failed",
         "diagnosticCode":"smtp; 550 user unknown",
         "emailAddress":"recipient1@example.com"
      },
      {
         "status":"4.0.0",
         "action":"delayed",
         "emailAddress":"recipient2@example.com"
      }
   ],
   "reportingMTA": "example.com",
   "timestamp":"2012-05-25T14:59:38.605Z",
   "feedbackId":"000001378603176d-5a4b5ad9-6f30-4198-a8c3-b1eb0c270a1d-000000",
   "remoteMtaIp":"127.0.2.0"
}
```

バウンスとなった受取人

バウンスの通知には、1 人の受信者に関するものと複数の受信者に関するものがあります。bouncedRecipients フィールドはオブジェクトのリスト (バウンスの通知が関係する受取人ごとに 1 つのオブジェクト) を保持し、常に次のフィールドが含まれます。

フィールド名	説明
emailAddress	受取人の E メールアドレス。DSN が使用可能な場合、これは の Final-Recipient フィールドの値ですDSN。

オプションで、 DSNがバウンスにアタッチされている場合、次のフィールドが存在する場合もあります。

フィールド名	説明
action	からの Actionフィールドの値DSN。これは、この受信者にメッセージを配信しようとした結果、 Reporting-MTA によって実行されたアクションを示します。
status	からの Statusフィールドの値DSN。これは、メッセージの配信状態を示す、受取人ごとに個別の、トランスポート独立型ステータスコードです。
diagnosticCode	レポート によって発行されたステータスコードMTA。これは、 の Diagnostic-Codeフィールドの値ですDSN。このフィールドは、 には存在しない可能性があります DSN (したがって、 にも存在しない) JSON。

以下は、bouncedRecipients のリストに示されるオブジェクトの例です。

```
{
    "emailAddress": "recipient@example.com",
    "action": "failed",
    "status": "5.0.0",
    "diagnosticCode": "X-Postfix; unknown user"
}
```

バウンスのタイプ

バウンスオブジェクトには、バウンスタイプとして Undetermined、Permanent、または Transient が含まれます。Permanent バウンスタイプと Transient バランスタイプには、複数のバウンスサブタイプのいずれかも含まれます。

バウンスタイプが Transient のバウンス通知を受信した場合は、メッセージのバウンスを起こした問題が解決されたときに、この受取人に対して将来 E メールを送信できる可能性があります。

バウンスタイプが Permanent のバウンス通知を受信した場合、この受取人に将来 E メールを送信できる可能性はありません。このため、バウンスを生じたアドレスを持つ受取人はメーリングリストから即座に削除してください。

> **Note**
>
> ソフトバウンス (受信者の受信トレイがいっぱいになっているなど、一時的な問題に関連するバウンス) が発生すると、Amazon は一定期間にわたって E メールの再配信SESを試みます。その期間の終了時に、Amazon SESが E メールをまだ配信できない場合、Amazon は試行を停止します。
> Amazon SESは、ハードバウンス、および配信を停止したソフトバウンスの通知を提供します。ソフトバウンスが発生するたびに通知を受信する場合は、イベントの公開を有効にし、配信遅延イベントが発生したときに通知を送信するように設定します。

bounceType	bounceSubType	説明
Undetermined	Undetermined	受取人の E メールプロバイダーはバウンスメッセージを送信しました。バウンスメッセージには、Amazon がバウンスの理由を判断するSESのに十分な情報が含まれていませんでした。バウンスを生じた E メールのリターンパスヘッダーのアドレスに送信されたバウンス E メールには、E メールのバウンスを起こした問題について追加情報が含まれている可能性があります。
Permanent	General	受取人の E メールプロバイダーはハードバウンスメッセージを送信しました。

bounceType	bounceSubType	説明
		Important このタイプのバウンス通知を受信した場合は、受取人の E メールアドレスをメーリングリストから即座に削除してください。ハードバウンスを生じたアドレスにメッセージを送信すると、送信者としての評価に悪影響を及ぼす可能性があります。ハードバウンスを生じたアドレスに E メールを送信し続けると、追加の E メールの送信機能が一時停止される場合があります。「the section called “アカウントレベルのサプレッションリストの使用”」を参照してください。
Permanent	NoEmail	バウンスメッセージから受信者のメールアドレスを取得できませんでした。
Permanent	Suppressed	受信者の E メールアドレスは、最近のハードバウンスの発生履歴があるため、Amazon SES プレッションリストに記載されています。グローバルサプレッションリストを上書きするには、「Amazon SESアカウントレベルのサプレッションリストの使用」を参照してください。
Permanent	OnAccountSuppressionList	Amazon SESは、アカウントレベルのサプレッションリストに含まれているため、このアドレスへの送信を抑制しました。これは、バウンス率のメトリクスに対してはカウントされません。

bounceType	bounceSubType	説明
Transient	General	受取人の E メールプロバイダーは一般的なバウンスメッセージを送信しました。メッセージのバウンスを生じた問題が解決された場合、将来、同じ受取人にメッセージを送信できる可能性があります。 **Note** アクティブな自動応答ルール (不在メッセージなど) が設定されている受取人に E メールを送信すると、このタイプの通知を受け取る場合があります。レスポンスの通知タイプが であってもBounce、Amazon SESはアカウントのバウンス率を計算するときに自動レスポンスをカウントしません。
Transient	MailboxFull	受取人の E メールプロバイダーは、受取人の受信トレイが満杯であるために、バウンスメッセージを送信しました。メールボックスが満杯でなくなった場合、将来、同じ受取人に送信できる可能性があります。
Transient	MessageTooLarge	受取人の E メールプロバイダーは、受信したメッセージが大きすぎるために、バウンスメッセージを送信しました。メッセージのサイズを小さくすることで、同じ受取人にメッセージを送信できる可能性があります。
Transient	ContentRejected	受取人の E メールプロバイダーは、受信したメッセージにプロバイダーが許可しないコンテンツが含まれていたために、バウンスメッセージを送信しました。メッセージのコンテンツを変更することで、同じ受取人にメッセージを送信できる可能性があります。

bounceType	bounceSubType	説明
Transient	AttachmentRejected	受取人の E メールプロバイダーは、メッセージ内に許容されないコンテンツが含まれていたために、バウンスメッセージを送信しました。たとえば、一部の E メールプロバイダーは特定のファイルタイプのファイルが添付されたメッセージや、非常に大きなファイルが添付されたメッセージを拒否する場合があります。添付ファイルのコンテンツを削除または変更することで、同じ受取人にメッセージを送信できる可能性があります。

苦情のオブジェクト

苦情に関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
complainedRecipients	苦情の原因である可能性がある受取人についての情報を含むリスト。詳細については、「苦情を申告した受取人」を参照してください。
timestamp	が苦情通知を 8601 ISO 形式でISP送信した日時。このフィールドの日時は、Amazon が通知をSES受信した日時と同じではない場合があります。
feedbackId	苦情に関連付けられた一意の ID。
complaintSubType	complaintSubType フィールドの値は、null または OnAccountSuppressionList のいずれかになります。値が の場合OnAccount SuppressionList 、Amazon はメッセージSESを受け入れましたが、アカウントレベルのサプレッションリストに含まれていたため、送信を試みませんでした。

また、フィードバックレポートが苦情に添付されている場合、以下のフィールドが示される場合があります。

フィールド名	説明
userAgent	フィードバックレポートの User-Agent フィールドの値。これは、レポートを生成したシステムの名前とバージョンを示します。
complaintFeedbackType	から受け取ったフィードバックレポートの Feedback-Type フィールドの値ISP。これには、フィードバックのタイプが含まれます。
arrivalDate	フィードバックレポートの Arrival-Date または Received-Date フィールドの値 (ISO8601 形式) 。このフィールドはレポートにない場合があります (したがって、 にもない JSON) 。

以下は、complaint オブジェクトの例です。

```
{
   "userAgent":"ExampleCorp Feedback Loop (V0.01)",
   "complainedRecipients":[
      {
         "emailAddress":"recipient1@example.com"
      }
   ],
   "complaintFeedbackType":"abuse",
   "arrivalDate":"2009-12-03T04:24:21.000-05:00",
   "timestamp":"2012-05-25T14:59:38.623Z",
   "feedbackId":"000001378603177f-18c07c78-fa81-4a58-9dd1-fedc3cb8f49a-000000"
}
```

苦情を申告した受取人

complainedRecipients フィールドには、苦情の送信元と思われる受信者のリストが含まれます。この情報を使用して、苦情を送信した受信者を特定し、その受信者をメーリングリストからすぐに削除する必要があります。

> **Important**
>
> ほとんどの ISPs は、苦情を送信した受信者の E メールアドレスを苦情通知から削除します。このため、このリストには、元のメッセージの受信者と苦情を受け取った ISP に基づいて、苦情を送信した可能性のある受信者に関する情報が含まれています。Amazon は元のメッセージに対して検索SESを実行して、この受信者リストを決定します。

JSON このリストの オブジェクトには、次のフィールドが含まれます。

フィールド名	説明
emailAddress	受取人の E メールアドレス。

以下は、苦情を申告した受取人のオブジェクトの例です。

```
{ "emailAddress": "recipient1@example.com" }
```

> **Note**
>
> 受信者 1 人につき 1 つのメッセージが送信されるように制限している (BCC 行に 30 個の異なる E メールアドレスを指定して送信しない) 場合は、この動作を利用して、メッセージに関する苦情を伝えている E メールアドレスをより確実に特定することできます。

苦情のタイプ

Internet Assigned Numbers Authority のウェブサイトによるとISP、レポート によって割り当てられた complaintFeedbackTypeフィールドに次の苦情タイプが表示されることがあります。

- abuse — 未承諾またはその他種類の不正使用メールを示します。
- auth-failure — E メールの認証の障害を示します。

- fraud — なんらかの詐欺またはフィッシング行為を示します。
- not-spam — レポートの提供者がこのメッセージをスパムではないと見なしていることを示します。このタイプは、誤ってスパムとしてタグ付けまたは分類されたメッセージを修正するために使用される場合があります。
- other — その他の登録されたタイプに該当しないフィードバックを示します。
- virus — 元のメッセージでウイルスが見つかったことを示します。

配信オブジェクト

配信に関する情報を含む JSON オブジェクトには、常に次のフィールドがあります。

フィールド名	説明
timestamp	Amazon が受信者のメールサーバーに E メールをSES配信した時刻 (ISO8601 形式) 。
processingTimeMillis	Amazon が送信者からのリクエストSESを受け入れてから受信者のメールサーバーにメッセージを渡すまでのミリ秒単位の時間。
recipients	配信通知を適用する E メールの対象となる受信者のリスト。
smtpResponse	Amazon からの E メールを受け入れISPたリモート のSMTPレスポンスメッセージSES。このメッセージは、E メール、メールサーバー、および の受信によって異なりますISP。
reportingMTA	SES メールを送信した Amazon メールサーバーのホスト名。
remoteMtaIp	Amazon が E メールをSES配信MTAした の IP アドレス。

以下は、delivery オブジェクトの例です。

```
{
```

```
    "timestamp":"2014-05-28T22:41:01.184Z",
    "processingTimeMillis":546,
    "recipients":["success@simulator.amazonses.com"],
    "smtpResponse":"250 ok:  Message 64111812 accepted",
    "reportingMTA":"a8-70.smtp-out.amazonses.com",
    "remoteMtaIp":"127.0.2.0"
}
```

Amazon の Amazon SNS通知の例 SES

以下のセクションでは、3 種類の通知の例を紹介します。

- バウンス通知の例については、「Amazon SNS バウンス通知の例」を参照してください。
- 苦情通知の例については、「Amazon SNS 苦情通知の例」を参照してください。
- 配信通知の例については、「Amazon SNS配信通知の例」を参照してください。

Amazon SNS バウンス通知の例

このセクションでは、フィードバックを送信した E メール受信者から提供された配信ステータス通知 (DSN) の有無にかかわらず、バウンス通知の例を示します。

によるバウンス通知 DSN

以下は、 DSNと元の E メールヘッダーを含むバウンス通知の例です。バウンス通知が元の E メールヘッダーを含めるように設定されていない場合は、通知内の `mail` オブジェクトに `headersTruncated`、`headers` および `commonHeaders` フィールドが含まれません。

```
   {
       "notificationType":"Bounce",
       "bounce":{
          "bounceType":"Permanent",
          "reportingMTA":"dns; email.example.com",
          "bouncedRecipients":[
             {
                "emailAddress":"jane@example.com",
                "status":"5.1.1",
                "action":"failed",
                "diagnosticCode":"smtp; 550 5.1.1 <jane@example.com>... User"
             }
          ],
          "bounceSubType":"General",
          "timestamp":"2016-01-27T14:59:38.237Z",
```

```
         "feedbackId":"00000138111222aa-33322211-cccc-cccc-cccc-ddddaaaa068a-000000",
         "remoteMtaIp":"127.0.2.0"
      },
      "mail":{
         "timestamp":"2016-01-27T14:59:38.237Z",
         "source":"john@example.com",
         "sourceArn": "arn:aws:ses:us-east-1:888888888888:identity/example.com",
         "sourceIp": "127.0.3.0",
         "sendingAccountId":"123456789012",
         "callerIdentity": "IAM_user_or_role_name",
         "messageId":"00000138111222aa-33322211-cccc-cccc-cccc-ddddaaaa0680-000000",
         "destination":[
           "jane@example.com",
           "mary@example.com",
           "richard@example.com"],
         "headersTruncated":false,
         "headers":[
          {
            "name":"From",
            "value":"\"John Doe\" <john@example.com>"
          },
          {
            "name":"To",
            "value":"\"Jane Doe\" <jane@example.com>, \"Mary Doe\" <mary@example.com>,
\"Richard Doe\" <richard@example.com>"
          },
          {
            "name":"Message-ID",
            "value":"custom-message-ID"
          },
          {
            "name":"Subject",
            "value":"Hello"
          },
          {
            "name":"Content-Type",
            "value":"text/plain; charset=\"UTF-8\""
          },
          {
            "name":"Content-Transfer-Encoding",
            "value":"base64"
          },
          {
            "name":"Date",
```

```
            "value":"Wed, 27 Jan 2016 14:05:45 +0000"
         }
        ],
        "commonHeaders":{
           "from":[
              "John Doe <john@example.com>"
           ],
           "date":"Wed, 27 Jan 2016 14:05:45 +0000",
           "to":[
              "Jane Doe <jane@example.com>, Mary Doe <mary@example.com>, Richard Doe
<richard@example.com>"
           ],
           "messageId":"custom-message-ID",
           "subject":"Hello"
         }
      }
   }
```

を使用しないバウンス通知 DSN

以下は、元の E メールヘッダーを含むが、 を含まないバウンス通知の例ですDSN。バウンス通知が元の E メールヘッダーを含めるように設定されていない場合は、通知内の `mail` オブジェクトに `headersTruncated`、`headers` および `commonHeaders` フィールドが含まれません。

```
  {
     "notificationType":"Bounce",
     "bounce":{
        "bounceType":"Permanent",
        "bounceSubType": "General",
        "bouncedRecipients":[
           {
              "emailAddress":"jane@example.com"
           },
           {
              "emailAddress":"richard@example.com"
           }
        ],
        "timestamp":"2016-01-27T14:59:38.237Z",
        "feedbackId":"00000137860315fd-869464a4-8680-4114-98d3-716fe35851f9-000000",
        "remoteMtaIp":"127.0.2.0"
     },
     "mail":{
        "timestamp":"2016-01-27T14:59:38.237Z",
```

```
        "messageId":"00000137860315fd-34208509-5b74-41f3-95c5-22c1edc3c924-000000",
        "source":"john@example.com",
        "sourceArn": "arn:aws:ses:us-east-1:888888888888:identity/example.com",
        "sourceIp": "127.0.3.0",
        "sendingAccountId":"123456789012",
        "callerIdentity": "IAM_user_or_role_name",
        "destination":[
           "jane@example.com",
           "mary@example.com",
           "richard@example.com"
        ],
       "headersTruncated":false,
       "headers":[
        {
           "name":"From",
           "value":"\"John Doe\" <john@example.com>"
        },
        {
           "name":"To",
           "value":"\"Jane Doe\" <jane@example.com>, \"Mary Doe\" <mary@example.com>,
\"Richard Doe\" <richard@example.com>"
        },
        {
           "name":"Message-ID",
           "value":"custom-message-ID"
        },
        {
           "name":"Subject",
           "value":"Hello"
        },
        {
           "name":"Content-Type",
           "value":"text/plain; charset=\"UTF-8\""
        },
        {
           "name":"Content-Transfer-Encoding",
           "value":"base64"
        },
        {
           "name":"Date",
           "value":"Wed, 27 Jan 2016 14:05:45 +0000"
         }
        ],
        "commonHeaders":{
```

```
          "from":[
             "John Doe <john@example.com>"
          ],
          "date":"Wed, 27 Jan 2016 14:05:45 +0000",
          "to":[
             "Jane Doe <jane@example.com>, Mary Doe <mary@example.com>, Richard Doe
 <richard@example.com>"
          ],
          "messageId":"custom-message-ID",
          "subject":"Hello"
       }
    }
 }
```

Amazon SNS 苦情通知の例

このセクションには、フィードバックを送信した E メール受取人によって提供されるフィードバックレポートを含む苦情通知と含まない苦情通知の例が記載されています。

フィードバックレポートを含む苦情通知

以下は、フィードバックレポートおよび元の E メールヘッダーを含む苦情通知の例です。苦情通知が元の E メールヘッダーを含めるように設定されていない場合は、通知内の `mail` オブジェクトに `headersTruncated`、`headers` および `commonHeaders` フィールドが含まれません。

```
  {
     "notificationType":"Complaint",
     "complaint":{
        "userAgent":"AnyCompany Feedback Loop (V0.01)",
        "complainedRecipients":[
           {
              "emailAddress":"richard@example.com"
           }
        ],
        "complaintFeedbackType":"abuse",
        "arrivalDate":"2016-01-27T14:59:38.237Z",
        "timestamp":"2016-01-27T14:59:38.237Z",
        "feedbackId":"000001378603177f-18c07c78-fa81-4a58-9dd1-fedc3cb8f49a-000000"
     },
     "mail":{
        "timestamp":"2016-01-27T14:59:38.237Z",
        "messageId":"000001378603177f-7a5433e7-8edb-42ae-af10-f0181f34d6ee-000000",
        "source":"john@example.com",
```

```
        "sourceArn": "arn:aws:ses:us-east-1:888888888888:identity/example.com",
        "sourceIp": "127.0.3.0",
        "sendingAccountId":"123456789012",
        "callerIdentity": "IAM_user_or_role_name",
        "destination":[
           "jane@example.com",
           "mary@example.com",
           "richard@example.com"
        ],
         "headersTruncated":false,
         "headers":[
          {
            "name":"From",
            "value":"\"John Doe\" <john@example.com>"
          },
          {
            "name":"To",
            "value":"\"Jane Doe\" <jane@example.com>, \"Mary Doe\" <mary@example.com>,
\"Richard Doe\" <richard@example.com>"
          },
          {
            "name":"Message-ID",
            "value":"custom-message-ID"
          },
          {
            "name":"Subject",
            "value":"Hello"
          },
          {
            "name":"Content-Type",
            "value":"text/plain; charset=\"UTF-8\""
          },
          {
            "name":"Content-Transfer-Encoding",
            "value":"base64"
          },
          {
            "name":"Date",
            "value":"Wed, 27 Jan 2016 14:05:45 +0000"
          }
        ],
        "commonHeaders":{
          "from":[
             "John Doe <john@example.com>"
```

```
          ],
          "date":"Wed, 27 Jan 2016 14:05:45 +0000",
          "to":[
             "Jane Doe <jane@example.com>, Mary Doe <mary@example.com>, Richard Doe
<richard@example.com>"
          ],
          "messageId":"custom-message-ID",
          "subject":"Hello"
       }
    }
  }
```

フィードバックレポートを含まない苦情通知

以下は、元の E メールヘッダーを含むがフィードバックレポートを含まない苦情通知の例です。苦情通知が元の E メールヘッダーを含めるように設定されていない場合は、通知内の `mail` オブジェクトに `headersTruncated`、`headers` および `commonHeaders` フィールドが含まれません。

```
  {
     "notificationType":"Complaint",
     "complaint":{
        "complainedRecipients":[
           {
              "emailAddress":"richard@example.com"
           }
        ],
        "timestamp":"2016-01-27T14:59:38.237Z",
        "feedbackId":"0000013786031775-fea503bc-7497-49e1-881b-a0379bb037d3-000000"
     },
     "mail":{
        "timestamp":"2016-01-27T14:59:38.237Z",
        "messageId":"0000013786031775-163e3910-53eb-4c8e-a04a-f29debf88a84-000000",
        "source":"john@example.com",
        "sourceArn": "arn:aws:ses:us-east-1:888888888888:identity/example.com",
        "sourceIp": "127.0.3.0",
        "sendingAccountId":"123456789012",
        "callerIdentity": "IAM_user_or_role_name",
        "destination":[
           "jane@example.com",
           "mary@example.com",
           "richard@example.com"
        ],
        "headersTruncated":false,
```

```
        "headers":[
         {
           "name":"From",
           "value":"\"John Doe\" <john@example.com>"
         },
         {
           "name":"To",
           "value":"\"Jane Doe\" <jane@example.com>, \"Mary Doe\" <mary@example.com>,
\"Richard Doe\" <richard@example.com>"
         },
         {
           "name":"Message-ID",
           "value":"custom-message-ID"
         },
         {
           "name":"Subject",
           "value":"Hello"
         },
         {
           "name":"Content-Type",
           "value":"text/plain; charset=\"UTF-8\""
         },
         {
           "name":"Content-Transfer-Encoding",
           "value":"base64"
         },
         {
           "name":"Date",
           "value":"Wed, 27 Jan 2016 14:05:45 +0000"
         }
         ],
         "commonHeaders":{
            "from":[
               "John Doe <john@example.com>"
            ],
            "date":"Wed, 27 Jan 2016 14:05:45 +0000",
            "to":[
               "Jane Doe <jane@example.com>, Mary Doe <mary@example.com>, Richard Doe
<richard@example.com>"
            ],
            "messageId":"custom-message-ID",
            "subject":"Hello"
         }
      }
```

```
}
```

Amazon SNS配信通知の例

以下は、元の E メールヘッダーを含む配信通知の例です。配信通知が元の E メールヘッダーを含めるように設定されていない場合は、通知内の `mail` オブジェクトに `headersTruncated`、`headers` および `commonHeaders` フィールドが含まれません。

```
{
   "notificationType":"Delivery",
   "mail":{
      "timestamp":"2016-01-27T14:59:38.237Z",
      "messageId":"0000014644fe5ef6-9a483358-9170-4cb4-a269-f5dcdf415321-000000",
      "source":"john@example.com",
      "sourceArn": "arn:aws:ses:us-east-1:888888888888:identity/example.com",
      "sourceIp": "127.0.3.0",
      "sendingAccountId":"123456789012",
      "callerIdentity": "IAM_user_or_role_name",
      "destination":[
         "jane@example.com"
      ],
       "headersTruncated":false,
       "headers":[
        {
           "name":"From",
           "value":"\"John Doe\" <john@example.com>"
        },
        {
           "name":"To",
           "value":"\"Jane Doe\" <jane@example.com>"
        },
        {
           "name":"Message-ID",
           "value":"custom-message-ID"
        },
        {
           "name":"Subject",
           "value":"Hello"
        },
        {
           "name":"Content-Type",
           "value":"text/plain; charset=\"UTF-8\""
        },
```

```
          {
             "name":"Content-Transfer-Encoding",
             "value":"base64"
          },
          {
             "name":"Date",
             "value":"Wed, 27 Jan 2016 14:58:45 +0000"
          }
        ],
        "commonHeaders":{
          "from":[
             "John Doe <john@example.com>"
          ],
          "date":"Wed, 27 Jan 2016 14:58:45 +0000",
          "to":[
             "Jane Doe <jane@example.com>"
          ],
          "messageId":"custom-message-ID",
          "subject":"Hello"
        }
     },
    "delivery":{
       "timestamp":"2016-01-27T14:59:38.237Z",
       "recipients":["jane@example.com"],
       "processingTimeMillis":546,
       "reportingMTA":"a8-70.smtp-out.amazonses.com",
       "smtpResponse":"250 ok:  Message 64111812 accepted",
       "remoteMtaIp":"127.0.2.0"
    }
}
```

Amazon SES での ID 承認の使用

ID 承認ポリシーでは、その ID に対して許可または拒否される SES API アクションと条件を指定することで、個々の検証済み ID が Amazon SES をどのように使用できるかを定義します。

これらの承認ポリシーを使用することにより、いつでもアクセス許可を変更または取り消して、ID に対する制御を維持できます。他のユーザーが、そのユーザーの SES アカウントで、自分が所有する (ドメインまたはメールアドレス) ID を使用することを承認することもできます。

トピック

- Amazon SES ポリシーの構造分析

- Amazon SES の ID 承認ポリシーの作成
- Amazon SES の ID ポリシーの例
- Amazon SES の ID 承認ポリシーの管理

Amazon SES ポリシーの構造分析

ポリシーは特定の構造に準拠し、要素を含み、特定の要件を満たす必要があります。

ポリシーの構造

各承認ポリシーは、ID にアタッチされる JSON ドキュメントです。各ポリシーには以下のセクションがあります。

- ポリシー全体に関する情報 (ドキュメントの最上部)。
- 1 つ以上の個別のステートメント (それぞれが 1 セットのアクセス許可を説明)。

次の例では、AWS アカウント ID 123456789012 に、検証済みドメイン example.com についてアクションセクションで指定されたアクセス許可を付与します。

```
{
  "Id":"ExampleAuthorizationPolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeAccount",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:123456789012:identity/example.com",
      "Principal":{
        "AWS":[
          "123456789012"
        ]
      },
      "Action":[
        "ses:GetEmailIdentity",
        "ses:UpdateEmailIdentityPolicy",
        "ses:ListRecommendations",
        "ses:CreateEmailIdentityPolicy",
        "ses:DeleteEmailIdentity"
      ]
    }
```

```
  ]
}
```

他の承認ポリシーの例については、「ID ポリシーの例」を参照してください。

ポリシーの要素

このセクションでは、ID 承認ポリシーに含まれる要素について説明します。まず、ポリシー全体の要素について説明した後、その要素が含まれるステートメントにのみ適用される要素について説明します。その後、ステートメントに条件を追加する方法について説明します。

エレメントの構文の詳細については、IAM ユーザーガイドの「IAM ポリシー言語の文法」を参照してください。

ポリシー全体に関する情報

ポリシー全体の要素として、Id と Version の 2 つがあります。次の表に、これらの要素についての情報を示します。

名前	説明	必須	有効な値
Id	ポリシーを一意に識別します。	いいえ	任意の文字列
Version	ポリシーアクセス言語のバージョンを指定します。	いいえ	任意の文字列。ベストプラクティスとして、このフィールドに値 "2012-10-17" を含めることをお勧めします。

個別のポリシーに関するステートメント

ID 承認ポリシーには、少なくとも 1 つのステートメントが必要です。各ステートメントには、次の表に示す要素を含めることができます。

名前	説明	必須	有効な値
Sid	ステートメントを一意に識別します。	いいえ	任意の文字列。

名前	説明	必須	有効な値
`Effect`	評価時にポリシーのステートメントによって返される結果を指定します。	はい	"Allow" または "Deny"。
`Resource`	ポリシーが適用されるアイデンティティを指定します。 (送信承認の場合、これは ID 所有者が代理送信者に使用を承認するメールアドレスまたはドメインです)。	可能	ID の Amazon リソースネーム (ARN)。

名前	説明	必須	有効な値
Principal	ステートメントで、アクセス許可を受け取る AWS アカウント、ユーザー、または AWS サービスを指定します。	可能	有効な AWS アカウント ID、ユーザー ARN、または AWS サービス。AWS アカウントID とユーザー ARN は、"AWS" (例えば、"AWS": ["123456789012"] または "AWS": ["arn:aws:iam::123456789012:root"]) を使用して指定され、AWS サービス名は、"Service" (例えば、"Service": ["cognito-idp.amazonaws.com"]) を使用して指定されます。 ユーザー ARN の形式の例については、「AWS 全般のリファレンス」を参照してください。

名前	説明	必須	有効な値
Action	ステートメントが適用されるアクションを指定します。	可能	"ses:BatchGetMetricData"、"ses:CancelExportJob"、"ses:CreateDeliverabilityTestReport"、"ses:CreateEmailIdentityPolicy"、"ses:CreateExportJob"、"ses:DeleteEmailIdentity"、"ses:DeleteEmailIdentityPolicy"、"ses:GetDomainStatisticsReport"、"ses:GetEmailIdentity"、"ses:GetEmailIdentityPolicies"、"ses:GetExportJob"、"ses:ListExportJobs"、"ses:ListRecommendations"、"ses:PutEmailIdentityConfigurationSetAttributes"、"ses:PutEmailIdentityDkimAttributes"、"ses:PutEmailIdentityDkimSigningAttributes"、"ses:PutEmailIdentityFeedbackAttributes"、"ses:PutEmailIdentityMailFromAttributes"、"ses:TagResource"、"ses:UntagResour

名前	説明	必須	有効な値
			ce"、"ses:UpdateEmailIdentityPolicy" (送信承認アクション: "ses:SendEmail"、"ses:SendRawEmail"、"ses:SendTemplatedEmail"、"ses:SendBulkTemplatedEmail") これらのオペレーションは 1 つ以上指定できます。
`Condition`	アクセス許可についての制限や詳細を指定します。	いいえ	この表の後の条件についての情報を参照してください。

条件

条件は、ステートメントのアクセス権限に関する制限のことです。ステートメントの中でも、記述が最も詳細になるのが、この条件部分です。キーは、リクエストの日時など、アクセス制限に使用される基本項目です。

制限は、条件とキーの両方を使用して定義します。たとえば、代理送信者が 2019 年 7 月 30 日以降はお客様の代わりに Amazon SES にリクエストを行うことができないように制限する場合、`DateLessThan` という条件を使用します。キーは `aws:CurrentTime` を使用し、値を `2019-07-30T00:00:00Z` に設定します。

SES は、以下の AWS 全体のポリシーキーのみを実装しています。

- `aws:CurrentTime`
- `aws:EpochTime`
- `aws:SecureTransport`
- `aws:SourceIp`
- `aws:SourceVpc`

- aws:SourceVpce
- aws:UserAgent
- aws:VpcSourceIp

これらのキーについては、「IAM ユーザーガイド」を参照してください。

ポリシーの要件

ポリシーは、以下の要件をすべて満たしている必要があります。

- 各ポリシーには、少なくとも 1 つのステートメントが含まれている必要があります。
- 各ポリシーには、少なくとも 1 つの有効なプリンシパルが含まれている必要があります。
- 各ポリシーには、1 つのリソースを指定する必要があります。またそのリソースは、ポリシーがアタッチされている ID の ARN である必要があります。
- アイデンティティ所有者は、最大 20 のポリシーにそれぞれ一意のアイデンティティを関連付けることができます。
- ポリシーのサイズが 4 キロバイト (KB) を超えることはできません。
- ポリシー名が 64 文字を超えることはできません。また、英数字、ダッシュ、アンダースコアのみを含めることができます。

Amazon SES の ID 承認ポリシーの作成

ID 承認ポリシーは、特定の ID に対して許可または拒否される API アクションとその条件を指定するステートメントで構成されます。

所有する Amazon SES ドメインまたはメールアドレスを承認するには、承認ポリシーを作成し、そのポリシーを ID にアタッチします。ID には、0、1、または多くのポリシーを含めることができます。ただし、1 つのポリシーは 1 つの ID にのみ関連付けることができます。

ID 承認ポリシーで使用できる API アクションのリストについては、the section called “個別のポリシーに関するステートメント” テーブルの「アクション」行を参照してください。

ID 承認ポリシーは、次のような方法で作成できます。

- Policy Generator を使用して – SES コンソールで Policy Generator を使用することによって、シンプルなポリシーを作成できます。SES API アクションに対するアクセス許可を許可または拒否

するだけでなく、アクションを条件で制約できます。さらに、Policy Generator を使用して、ポリシーの基本構造をすばやく作成した後、ポリシーを編集することでカスタマイズすることもできます。

- カスタムポリシーを作成して – 詳細な条件を追加したり、AWS サービスをプリンシパルとして使用したりする場合は、SES コンソールまたは SES API を使用してカスタムポリシーを作成し、それを ID にアタッチできます。

トピック

Policy Generatorの使用

Policy Generator を使用し、次の手順に従ってシンプルな送信承認ポリシーを作成できます。

Policy Generatorを用してポリシーを作成するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインの設定で、検証済み ID を選択します。
3. [Verified identities] (検証済み ID) 画面の [Identities] (ID) コンテナで、承認ポリシーを作成する検証済み ID を選択します。
4. 前のステップで選択した検証済み ID の詳細画面で、[Authorization] (承認) タブを選択します。
5. [Authorization policies] (承認ポリシー) ペインで、[Create policy] (ポリシーの作成) を選択し、ドロップダウンから [Use policy generator] (Policy Generator の使用) を選択します。
6. [Create statement] (ステートメントの作成) ペインで、効果フィールドの [Allow] (許可) を選択します。(この ID を制限するポリシーを作成する場合は、代わりに [Deny] (拒否) を選択します)。
7. [Principals] (プリンシパル) フィールドに、AWS アカウント ID、IAM ユーザー ARN、またはこの ID に対して承認するアクセス許可を受け取る AWS サービスを入力し、[Add] (追加) を選択します (複数を承認する場合は、それぞれについて、この手順を繰り返します)。
8. [Actions] (アクション) フィールドで、プリンシパルに対して承認する各アクションのチェックボックスをオンにします。
9. (オプション) アクセス許可に限定的なステートメントを追加する場合は、[Specify conditions] (条件を指定) を展開します。

a. [Operator] (演算子) ドロップダウンから演算子を選択します。

b. [Key] (キー) ドロップダウンからタイプを選択します。

c. 選択したキータイプに応じて、その値を値フィールドに入力します。(条件をさらに追加する場合は、[Add new condition] (新しい条件を追加) を選択します。条件を追加するたびに、この手順を繰り返します)

10. [Save statement] (ステートメントを保存) を選択します。

11. (オプション) ポリシーにステートメントを追加する場合は、[Create another statement] (別のステートメントを作成) を展開して、ステップ 6 ~ 10 を繰り返します。

12. [Next] (次へ) を選択すると、[Customize policy] (ポリシーのカスタマイズ) 画面で、[Edit policy details] (ポリシーの詳細の編集) コンテナには、ポリシーの [Name] (名前) と [Policy document] (ポリシードキュメント) 自体を変更またはカスタマイズするためのフィールドが表示されます。

13. [Next] (次へ) を選択すると、[Review and apply] (確認して適用) 画面の [Overview] (概要) コンテナに、承認する検証済み ID と、このポリシーの名前が表示されます。[Policy document] (ポリシードキュメント) ペインには、追加した条件とともに、作成した実際のポリシーが表示されます。ポリシーを確認し、内容が正しい場合は、[Apply policy] (ポリシーの適用) をクリックします。(変更や修正の必要がある場合は、[Previous] (戻る) をクリックして、[Edit policy details] (ポリシーの詳細の編集) コンテナで作業を行います)

カスタムポリシーの作成

カスタムポリシーを作成してアイデンティティにアタッチする場合、次のオプションがあります。

- Amazon SES API の使用 – Amazon Simple Email Service API リファレンス で説明されている `PutIdentityPolicy` API を使用し、テキストエディタでポリシーを作成してアイデンティティにアタッチします。
- Amazon SES コンソールの使用 – テキストエディタを使用してポリシーを作成し、それを Amazon SES コンソールのカスタムポリシーエディタに貼り付けることでアイデンティティにアタッチします。以下の手順では、この方法について説明します。

カスタムポリシーエディタを使用してカスタムポリシーを作成するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインの設定で、検証済み ID を選択します。

3. [Verified identities] (検証済み ID) 画面の [Identities] (ID) コンテナで、承認ポリシーを作成する検証済み ID を選択します。
4. 前のステップで選択した検証済み ID の詳細画面で、[Authorization] (承認) タブを選択します。
5. [Authorization policies] (承認ポリシー) ペインで、[Create policy] (ポリシーの作成) を選択して、ドロップダウンから [Create custom policy] (カスタムポリシーの作成) を選択します。
6. [Policy document] (ポリシードキュメント) ペインで、JSON 形式のポリシーのテキストを入力または貼り付けます。Policy Generator を使用すると、基本的な構造のポリシーをすばやく作成でき、ここでカスタマイズすることもできます。
7. [ポリシーを適用] を選びます。(カスタムポリシーを変更する必要がある場合は、[Authorization] (承認) タブのチェックボックスをオンにし、[Edit] (編集) を選択して、[Policy document] (ポリシードキュメント) ペインで変更を行ってから、[Save changes] (変更の保存) を選択します)

Amazon SES の ID ポリシーの例

ID 承認を使用すると、ID の API アクションを許可または拒否する条件をきめ細かく指定することができます。

以下の例は、API アクションのさまざまな側面を制御するポリシーを記述する方法を示しています。

- プリンシパルの指定
- アクションの制限
- 複数のステートメントの使用

プリンシパルの指定

プリンシパルは、アクセス許可を付与するエンティティであり、AWS アカウント、AWS Identity and Access Management (IAM) ユーザー、または同じアカウントに属する AWS サービスです。

次の例は、AWS ID 123456789012 が、AWS アカウント 123456789012 によっても所有されている検証済み ID example.com を制御することを許可するシンプルなポリシーを示しています。

```
{
  "Id":"SampleAuthorizationPolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeMarketer",
      "Effect":"Allow",
```

```
      "Resource":"arn:aws:ses:us-east-1:123456789012:identity/example.com",
      "Principal":{
        "AWS":[
          "123456789012"
        ]
      },
      "Action":[
        "ses:DeleteEmailIdentity",
        "ses:PutEmailIdentityDkimSigningAttributes"
      ]
    }
  ]
}
```

次のポリシー例では、検証済み ID example.com を制御するアクセス許可を 2 人のユーザーに付与します。ユーザーは、Amazon リソースネーム (ARN) によって指定されます。

```
{
  "Id":"ExampleAuthorizationPolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeIAMUser",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:123456789012:identity/example.com",
      "Principal":{
        "AWS":[
          "arn:aws:iam::123456789012:user/John",
          "arn:aws:iam::123456789012:user/Jane"
        ]
      },
      "Action":[
        "ses:DeleteEmailIdentity",
        "ses:PutEmailIdentityDkimSigningAttributes"
      ]
    }
  ]
}
```

アクションの制限

承認する制御のレベルに応じて、ID 承認ポリシーで指定できる複数のアクションがあります。

```
"BatchGetMetricData",
"ListRecommendations",
"CreateDeliverabilityTestReport",
"CreateEmailIdentityPolicy",
"DeleteEmailIdentity",
"DeleteEmailIdentityPolicy",
"GetDomainStatisticsReport",
"GetEmailIdentity",
"GetEmailIdentityPolicies",
"PutEmailIdentityConfigurationSetAttributes",
"PutEmailIdentityDkimAttributes",
"PutEmailIdentityDkimSigningAttributes",
"PutEmailIdentityFeedbackAttributes",
"PutEmailIdentityMailFromAttributes",
"TagResource",
"UntagResource",
"UpdateEmailIdentityPolicy"
```

ID 承認ポリシーでは、プリンシパルをそれらのアクションの 1 つだけに制限することもできます。

```
{
  "Id":"ExamplePolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"ControlAction",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:123456789012:identity/example.com",
      "Principal":{
        "AWS":[
          "123456789012"
        ]
      },
      "Action":[
        "ses:PutEmailIdentityMailFromAttributes
      ]
    }
  ]
}
```

複数のステートメントの使用

ID 承認ポリシーには複数のステートメントを含めることができます。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では、2 人のユーザーが同じアカウント 123456789012 内の sender@example.com から getemailidentity にアクセスすることを拒否します。2 つ目のステートメントでは、同じアカウント 123456789012 内のプリンシパル Jack に対して UpdateEmailIdentityPolicy を拒否します。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"DenyGet",
      "Effect":"Deny",
      "Resource":"arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
      "Principal":{
        "AWS":[
          "arn:aws:iam::123456789012:user/John",
          "arn:aws:iam::123456789012:user/Jane"
        ]
      },
      "Action":[
        "ses:GetEmailIdentity"
      ]
    },
    {
      "Sid":"DenyUpdate",
      "Effect":"Deny",
      "Resource":"arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
      "Principal":{
        "AWS":"arn:aws:iam::123456789012:user/Jack"
      },
      "Action":[
        "ses:UpdateEmailIdentityPolicy"
      ]
    }
  ]
}
```

Amazon SES の ID 承認ポリシーの管理

ポリシーの作成と ID へのアタッチに加えて、以下のセクションで説明するように、ID のポリシーを編集、削除、リスト、取得することができます。

Amazon SES コンソールを使用してポリシーを管理する

Amazon SES ポリシーの管理では、Amazon SES コンソールを使用して、ID にアタッチされたポリシーを表示、編集、または削除する必要があります。

Amazon SES コンソールを使用してポリシーを管理するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左のナビゲーションペインで、[ID の検証] を選択します。
3. ID のリストで、管理する ID を選択します。
4. ID の詳細ページで、[Authorization] (承認) タブに移動します。ここでは、この ID に添付されているすべてのポリシーの一覧が表示されます。
5. 管理するポリシーのチェックボックスをオンにします。
6. 求める管理タスクに応じて、次のようにそれぞれのボタンを選択します。

   a. ポリシーを表示する場合は、[View policy] (ポリシーを表示) を選択します。コピーが必要な場合は、[Copy] (コピー) ボタンを選択すると、クリップボードにコピーされます。

   b. ポリシーを編集する場合は、[Edit] (編集) を選択します。[Policy document] (ポリシードキュメント) ペインで、ポリシーを編集してから、[Save changes] (変更の保存) を選択します。

   > **Note**
   >
   > アクセス許可を取り消すには、ポリシーを編集または削除します。

   c. ポリシーを削除する場合は、[Delete] (削除) を選択します。

   > **Important**
   >
   > ポリシーの削除は永続的です。ポリシーを削除する前に、テキストファイルにポリシーをコピーアンドペーストして、ポリシーをバックアップすることをお勧めします。

Amazon SES API を使用してポリシーを管理する

Amazon SES ポリシーの管理では、Amazon SES API を使用して、ID にアタッチされたポリシーを表示、編集、または削除する必要があります。

Amazon SES API を使用してポリシーを一覧表示と表示するには

- ListIdentityPolicies API オペレーションを使用して、ID にアタッチされたポリシーをリストできます。GetIdentityPolicies API オペレーションを使用することでポリシー自体を取得することもできます。

Amazon SES API を使用してポリシーを編集するには

- PutIdentityPolicy API 操作を使用して、ID にアタッチされたポリシーを編集できます。

Amazon SES API を使用してポリシーを削除するには

- DeleteIdentityPolicy API 操作を使用して、ID にアタッチされたポリシーを削除できます。

Amazon SES での送信承認の使用

Amazon SES を設定することで、別のユーザーが、そのユーザーの Amazon SES アカウントを使用して、自分が所有する (ドメインまたは E メールアドレス) IDからEメールを送信することを許可できます。送信承認機能では、ID のコントロールは維持されるため、いつでもアクセス許可を変更または取り消すことができます。たとえば、ビジネスの所有者は、送信承認を使用することで、サードパーティー (E メールマーケティング会社など) の E メールを、自分が所有するドメインから送信できるようにすることができます。

この章では、従来のクロスアカウント通知機能に代わる送信承認の詳細について説明します。まず、承認ポリシーの構造分析やポリシーの管理方法などの重要なトピックに関する Amazon SES での ID 承認の使用 で説明されているように、承認ポリシーを使用した ID ベースの承認の基本を理解する必要があります。

クロスアカウント通知のレガシーサポート

ID 所有者によって検証済み ID の 1 つから送信することを代理送信者が許可され、送信する E メールに関連付けられたバウンス、苦情、および配信に関するフィードバック通知は、従来、クロスアカウント通知を使用して設定されていました。この場合、代理送信者が所有していない ID にトピック

を関連付けていました (これがクロスアカウントです)。ただし、クロスアカウント通知は、代理送信に関連して設定セットと検証済み ID を使用して置き換えられました。この場合、代理送信者は、ID 所有者によって、検証済み ID の 1 つを使用して E メールを送信することを承認されています。この新しい方法では、代理送信者であるか、検証済み ID の所有者であるかに応じて、バウンス、苦情、配信、およびその他のイベント通知を次の 2 つの構成で柔軟に設定できます。

- 設定セット – 代理送信者は、自分の所有ではないが、承認ポリシーによって ID 所有者から送信を許可された検証済み ID から E メールを送信するとき、指定可能な独自の設定セットでイベント発行をセットアップできます。イベント発行を使用すると、バウンス通知、苦情通知、配信通知、およびその他のイベント通知を Amazon CloudWatch、Amazon Data Firehose、Amazon Pinpoint、Amazon SNS に発行できます。「イベント送信先の作成」を参照してください。
- 検証済み ID – ID 所有者は、代理送信者が検証済み ID のいずれかを使用して E メールを送信することを許可するだけでなく、代理送信者のリクエストに応じて、共有 ID でフィードバック通知を設定して、代理送信者が所有する SNS トピックを使用することもできます。代理送信者が SNS トピックを所有するため、代理送信者だけがこれらの通知を受け取ります。「所有していない SNS トピック」を設定する方法については、承認ポリシーの手順にあるステップ 14 を参照してください。

> **Note**
>
> アカウント内で現在使用されているレガシークロスアカウント通知の互換性を保つため、クロスアカウント通知がサポートされています。このサポートは、Amazon SES クラシックコンソールで作成した現在のクロスアカウントの変更と使用に限られています。ただし、新しいクロスアカウント通知を作成することはできなくなっています。Amazon SES の新しいコンソールで新しいクロスアカウント通知を作成するには、イベント発行を使用した設定セット、または独自の SNS トピックで設定された検証済み ID を利用して代理送信という新しい方法を適用します。

トピック

- Amazon SESの送信承認の概要
- Amazon SES送信承認を行うためのID所有者のタスク
- Amazon SESの送信承認を行うための代理送信者のタスク

Amazon SESの送信承認の概要

このトピックでは、送信承認プロセスの概要を示し、送信承認と Amazon SES の E メール送信機能 (送信クォータや通知など) がどのように関係するかについて説明します。

このセクションでは以下の用語を使用します。

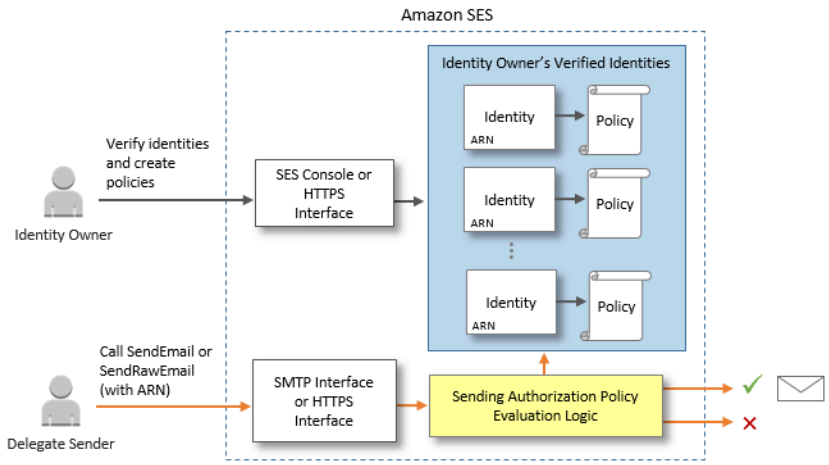
- アイデンティティ – Amazon SES ユーザーが E メールの送信に使用する E メールアドレスまたはドメイン。
- アイデンティティ所有者 – 「検証済みID」の手順を使用して、E メールアドレスまたはドメインの所有権を検証した Amazon SES ユーザー。
- 代理送信者 – AWS アカウント、AWS Identity and Access Management (IAM) ユーザー、または ID 所有者の代わりに承認ポリシーによって E メールを送信することを許可された AWS のサービスです。
- 送信承認ポリシー – アイデンティティにアタッチすることで、そのアイデンティティを使用して送信できるユーザーとその条件を指定できるドキュメント。
- Amazon リソースネーム (ARN) – すべての AWS サービス間で AWS リソースを一意に識別する標準化された方法。送信承認の場合、リソースは、ID 所有者が代理送信者に使用を許可した ID です。ARN の例は arn:aws:ses:us-east-1:123456789012:identity/example.com などです。

送信承認プロセス

送信承認は、送信承認ポリシーに基づいています。代理送信者が代わりに送信できるようにする場合、Amazon SES コンソールまたは Amazon SES API を使用して、送信承認ポリシーを作成してポリシーを ID に関連付けます。代理送信者がお客様の代わりに Amazon SES を通じて E メールを送信するとき、代理送信者はリクエストまたは E メールのヘッダーで ID の ARN を渡します。

Amazon SES は、E メール送信リクエストを受け取ると、ID のポリシーを確認し (存在する場合)、代理送信者がその ID の代わりに送信することが承認されているかどうかを判断します。代理送信者が承認されている場合、Amazon SES は E メールを受け入れます。承認されていない場合、Amazon SES はエラーメッセージを返します。

次の図は、送信承認の各概念の間の全体的な関係を示しています。



送信承認のプロセスは、以下のステップで構成されています。

1. ID 所有者は、代理送信者が使用する検証済み ID を選択します。ID が検証済みでない場合は、「検証済みID」を参照してください。

   > **Note**
   >
   > 代理送信者のために選択した検証済み ID には、デフォルト設定セットを割り当てることができません。

2. 代理送信者は、送信に使用する AWS アカウント ID または IAM ユーザーの ARN を ID 所有者に知らせます。
3. 代理送信者が ID 所有者のいずれかのアカウントから送信することを許可するよう ID 所有者が同意した場合、ID 所有者は Amazon SES コンソールまたは Amazon SES API を使用して送信承認ポリシーを作成し、そのポリシーを選択した ID にアタッチします。
4. ID 所有者が代理送信者に承認済み ID の ARN を付与し、代理送信者が E メールの送信時に Amazon SES に ARN を提供できるようにします。
5. 代理送信者は、代理送信時に指定された設定セットで有効になっているイベント発行を使用して、バウンスや苦情の通知を設定できます。ID 所有者は、バウンスや苦情のイベントに関する E メールフィードバック通知を代理送信者の Amazon SNS トピックに送信するように設定することもできます。

> **Note**
>
> ID 所有者が送信イベント通知を無効にする場合、代理送信者は、バウンスイベントと苦情イベントを Amazon SNS トピックまたは Firehose ストリームに公開するようにイベント発行を設定する必要があります。送信者は、送信する各 E メールにイベント発行ルールを含む設定セットも適用する必要があります。ID 所有者および代理送信者のいずれも、バウンスイベントと苦情イベントの通知を送信する方法を設定していない場合、または送信者がイベント公開ルールを使用する設定セットを適用しない場合は、ID 所有者が E メールのフィードバック転送を無効にしていても、Amazon SES は、E メールの Return-Path フィールドのアドレス (または Return-Path アドレスを指定しなかった場合はソースフィールド) に自動的にイベント通知を E メールで送信します。

6. 代理送信者は、リクエストまたは E メールのヘッダーで ID 所有者の ID の ARN を渡すことで、ID 所有者の代わりに Amazon SES を通じて E メールを送信します。代理送信者は、Amazon SES SMTP インターフェイスまたは Amazon SES API を使用して E メールを送信できます。Amazon SES は、リクエストを受け取ると、ID にアタッチされたポリシーを調べ、代理送信者が指定された 「From」アドレスと「Return Path」アドレスの使用を承認されている場合は E メールを受け入れます。承認されていない場合、Amazon SES はエラーを返し、メッセージを受け入れません。

> **Important**
>
> 代理送信者の AWS アカウントは、検証されていないアドレスに E メールを送信する前に、サンドボックスから削除する必要があります。

7. ID 所有者が代理送信者の承認を解除する場合は、ID 所有者が送信承認ポリシーを編集するか、ポリシーを完全に削除します。ID 所有者は、Amazon SES コンソールまたは Amazon SES API を使用してどちらのアクションも実行できます。

ID所有者または代理所有者がこれらのタスクを実行する方法の詳細については、それぞれ「ID所有者のタスク」または「代理送信者のタスク」を参照してください。

Eメール送信機能の属性

日次送信クォータ、返送と苦情、DKIM 署名、フィードバック転送などの Amazon SES E メール送信機能に関して、代理送信者とID所有者の役割を理解しておくことが重要です。属性は次のとおりです。

- 送信クォータ – ID所有者のIDから送信される E メールは、代理送信者のクォータにカウントされます。
- バウンスと苦情 – バウンスおよび苦情イベントは、代理送信者の Amazon SES アカウントに記録されます。したがって、代理送信者の評価に影響を及ぼします。
- DKIM 署名 – アイデンティティ所有者がアイデンティティの Easy DKIM 署名を有効にした場合、そのアイデンティティから送信されるすべての E メール (代理送信者が送信した E メールを含む) が DKIM 署名されます。E メールが DKIM 署名されるかどうかをコントロールできるのはアイデンティティ所有者だけです。
- 通知 – アイデンティティ所有者と代理送信者のいずれも、バウンスと苦情に関する通知を設定できます。E メールのアイデンティティ所有者は、E メールのフィードバック転送を有効にすることもできます。通知のセットアップについては、「Amazon SES 送信アクティビティのモニタリング」を参照してください。
- 検証 – アイデンティティ所有者には、「検証済みID」の手順に従って、代理送信者に試用を許可する E メールアドレスとドメインを自分が所有していることを検証する責任があります。代理送信者は、送信許可用に指定されたEメールアドレスまたはドメインを検証する必要はありません。

  > **Important**
  >
  > 代理送信者の AWS アカウントは、検証されていないアドレスに E メールを送信する前に、サンドボックスから削除する必要があります。

- AWS リージョン – 代理送信者は、アイデンティティ所有者のアイデンティティが検証された AWS リージョンから E メールを送信する必要があります。代理送信者にアクセス許可を付与する承認の送信ポリシーは、その地域のアイデンティティにアタッチする必要があります。
- 請求 – 代理送信者のアカウントから送信されるすべてのメッセージ (代理送信者がアイデンティティ所有者のアドレスを使用して送信する E メールを含む) の料金は、代理送信者に請求されます。

Amazon SES送信承認を行うためのID所有者のタスク

このセクションでは、送信承認の設定時にアイデンティティ所有者が実行する必要がある手順について説明します。

トピック

- Amazon SESの送信承認を行うためのIDの検証
- Amazon SESの送信承認に使用するID所有者通知の設定

- Amazon SES送信承認の代理送信者からの情報の取得
- Amazon での送信承認ポリシーの作成 SES
- 送信ポリシーの例
- Amazon SESの送信承認のための ID 情報を代理送信者に提供する

Amazon SESの送信承認を行うためのIDの検証

送信承認を設定するための最初の手順は、代理送信者が E メールを送信する E メールアドレスまたはドメインを自分が所有していることを証明することです。検証手順については、「検証済みID」を参照してください。

E メールアドレスまたはドメインが検証済みであるかどうかは、https://console.aws.amazon.com/ses/ の検証済み ID セクションでステータスをチェックするか、`GetIdentityVerificationAttributes` API 操作を使用して確認できます。

ユーザーまたは代理送信者が、検証されていない E メールアドレスに E メールを送信するには、アカウントを Amazon SES サンドボックスから削除するリクエストを送信する必要があります。詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。

> **Important**
>
> - 代理送信者の AWS アカウントは、未検証のアドレスとの間で E メールを送受信する前に、サンドボックスから削除する必要があります。
> - アカウントがサンドボックス内にある場合、ドメインやメールアドレスが ID アカウントで検証済みであっても、アカウントで検証されていないメールアドレスに送信することはできません。

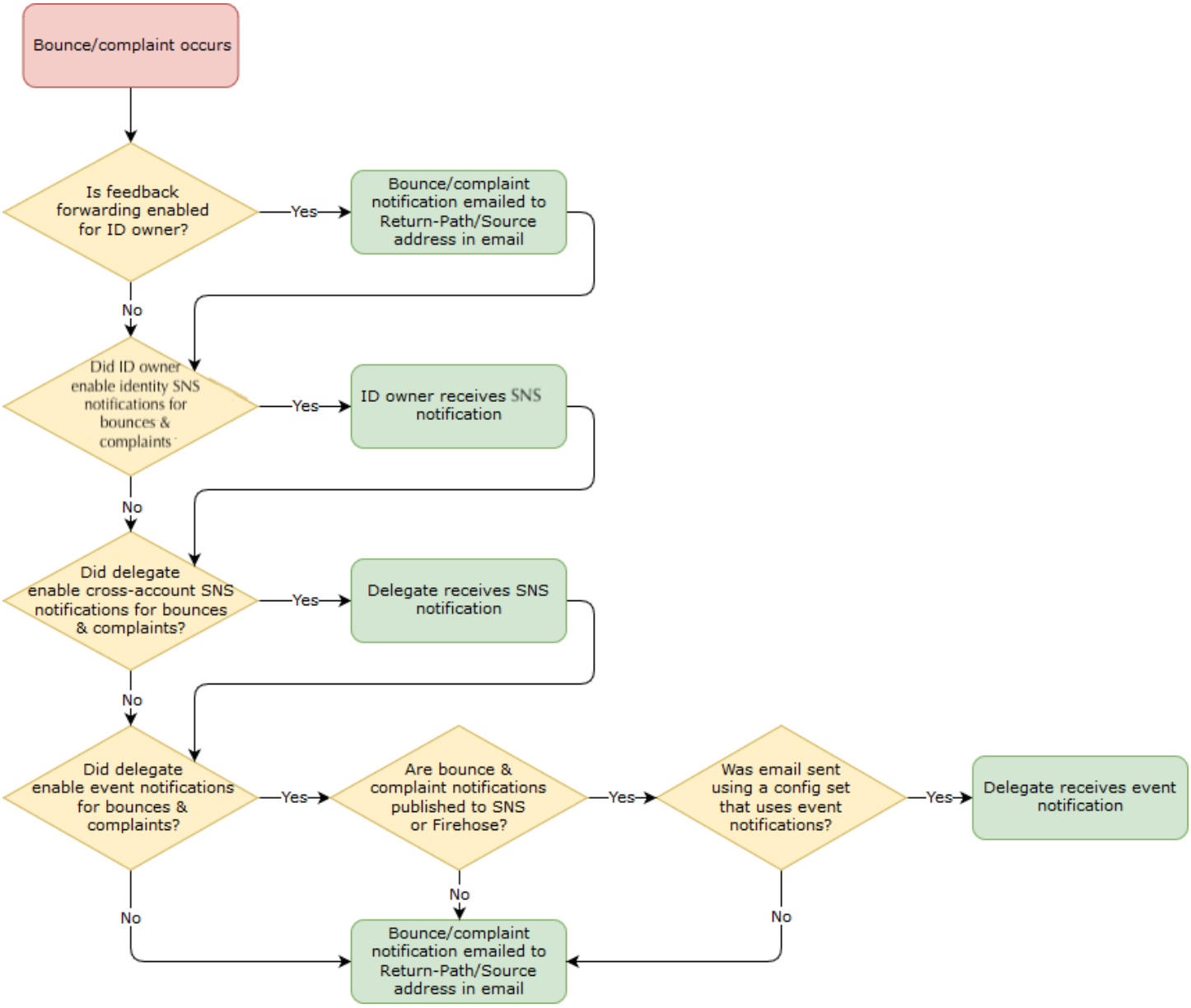
Amazon SESの送信承認に使用するID所有者通知の設定

代理送信者がお客様の代わりに E メールを送信することを承認する場合、Amazon SES は、それらの E メールが生成するすべてのバウンスと苦情は、お客様ではなく代理送信者のバウンスと苦情の制限に対してカウントします。ただし、代理送信者から送信されたメッセージの結果、IP アドレスがサードパーティーのスパム対策 DNS ベースのブラックホールリスト (DNSBL) に表示される場合、ID の評価が損なわれる可能性があります。このため、お客様が ID 所有者である場合は、代理送信を許可した ID を含め、すべての ID に対して E メールフィードバック転送を設定する必要があり

ます。詳細については、「E メールを介したAmazon SES に関する通知の受信」を参照してください。

代理送信者は、お客様が使用を許可した ID に対して、独自のバウンスおよび苦情の通知を設定する必要があります。イベント発行は、Amazon SNS トピックまたは Firehose ストリームにバウンスイベントや苦情イベントを発行するように設定できます。

ID 所有者および代理送信者のいずれも、バウンスイベントと苦情イベントの通知を送信する方法を設定していない場合、または送信者がイベント公開ルールを使用する設定セットを適用しない場合は、E メールのフィードバック転送を無効にしていても、Amazon SES は、E メールの Return-Path フィールドのアドレス (または Return-Path アドレスを指定しなかった場合はソースフィールド) に自動的にイベント通知を E メールで送信します。次のイメージは、このプロセスを示したものです。



Amazon SES送信承認の代理送信者からの情報の取得

送信承認ポリシーでは、少なくとも 1 つのプリンシパルを指定する必要があります。プリンシパルとは、検証済み ID のいずれかの代わりに送信できるようにアクセスを許可している代理送信者のエンティティです。Amazon SESの送信承認ポリシーの場合、プリンシパルは代理送信者の AWS アカウント、 AWS Identity and Access Management （ IAM) ユーザーARN、または AWS サービスのいずれかになります。

簡単に言うと、プリンシパル (代理送信者) は被付与者であり、お客様 (ID 所有者) は承認ポリシーの付与者になります。お客様 (ID 所有者) が、E メール、raw E メール、テンプレートに基づく E メー

ル、またはテンプレートに基づくバルク E メールの任意の組み合わせを、所有するリソース (検証済み ID) から送信するための Allowアクセス許可をプリンシパルに付与します。

最高のきめ細かな制御が必要な場合は、代理送信者の AWS アカウントのIAMユーザーではなく、1 人の代理送信者だけが送信できるように ユーザーを設定するよう代理送信者に依頼します。代理送信者は、IAM「 IAMユーザーガイド」の「 AWS アカウントでの IAM ユーザーの作成」で ユーザーの設定に関する情報を確認できます。

送信承認ポリシーに含めることができるように、代理送信者に AWS アカウント ID またはIAMユーザーの Amazon リソースネーム (ARN) を依頼します。代理送信者には、「ID所有者への情報提供」にあるこの情報を調べるための手順を参照してもらいます。代理送信者が AWS サービスの場合は、そのサービスのドキュメントを参照してサービス名を確認してください。

次のポリシー例は、ID 所有者のリソースからの送信を代理送信者に許可するために、ID 所有者が作成するポリシーで必要とされる基本的要素を示しています。ID 所有者は検証済み ID ワークフローに移動し、[Authorization] (認可) で Policy Generator を使用して、ID 所有者が所有するリソースの代わりに代理送信者が送信できるようにする次の基本ポリシーを最もシンプルな形式で作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSESSendEmail",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": [
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Resource": [
          "arn:aws:ses:us-east-1:444455556666:identity/bob@example.com"
      ],
      "Condition": {}
    }
  ]
}
```

上記のポリシーについては、次の凡例で主な要素とその所有者を説明します。

- プリンシパル – このフィールドには、代理送信者のIAMユーザー が入力されますARN。

- アクション – このフィールドには、ID 所有者が代理送信者に ID 所有者のリソースからの実行を許可している 2 つのSESアクション (`SendEmail` と `SendRawEmail`) が入力されます。
- リソース – このフィールドには、代理送信者に送信を許可している ID 所有者の検証済みリソースが入力されます。

Amazon での送信承認ポリシーの作成 SES

Amazon での承認ポリシーの作成と同様にSES、「」で説明されているようにID 承認ポリシーの作成、代理送信者が所有している E メールアドレスまたはドメイン (アイデンティティ) を使用して E メールを送信することを許可するには、SES送信APIアクションを指定してポリシーを作成し、そのポリシーをアイデンティティにアタッチします。

送信承認ポリシーで指定できるAPIアクションのリストについては、the section called “個別のポリシーに関するステートメント”表の「アクション」行を参照してください。

送信承認ポリシーは、Policy Generator を使用するか、カスタムポリシーを作成することで作成できます。いずれの方法でも、送信承認ポリシーの作成に固有の手順が提供されています。

> **Note**
>
> - E メールアドレス ID にアタッチする送信承認ポリシーは、対応するドメイン ID にアタッチするポリシーよりも優先されます。例えば、example.com に対して代理送信者を許可しないポリシーを作成し、sender@example.com に対して代理送信者を許可するポリシーを作成した場合、代理送信者は sender@example.com からメールを送信できますが、example.com ドメイン内の他のアドレスからは送信できません。
> - 代理送信者を許可する example.com のポリシーを作成し、代理送信者を拒否する sender@example.com のポリシーを作成した場合、代理送信者は example.com ドメインの任意のアドレスからメールを送信できますが、 sender@example.com ドメインの他のアドレスから送信することはできません。
> - SES 認可ポリシーの構造に慣れていない場合は、「」を参照してくださいポリシー分析。
> - 承認する ID がグローバルエンドポイント機能の一部としてセカンダリリージョンで複製されている場合は、プライマリリージョンとセカンダリリージョンの両方で ID の送信承認ポリシーを作成して、代理送信者が両方のリージョンで送信するためにこの ID を使用するアクセス許可を持つようにする必要があります。

Policy Generator を使用して送信承認ポリシーを作成する

Policy Generator を使用し、次の手順に従って送信承認ポリシーを作成できます。

Policy Generator を使用して送信承認ポリシーを作成するには

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [設定] で、[ID] を選択します。
3. [Verified identities] (検証済み ID) 画面の [Identities] (ID) コンテナで、代理送信者がお客様の代わりに送信することを許可する検証済み ID を選択します。
4. 検証済み ID の [承認] タブをクリックします。
5. [Authorization policies] (承認ポリシー) ペインで、[Create policy] (ポリシーの作成) を選択し、ドロップダウンから [Use policy generator] (Policy Generator の使用) を選択します。
6. [Create statement] (ステートメントの作成) ペインで、効果フィールドの [Allow] (許可) を選択します。(代理送信者を制限するポリシーを作成する場合は、代わりに [Deny] (拒否) を選択します)
7. プリンシパル フィールドに、代理送信者が共有した AWS アカウント ID またはIAMユーザー ARNを入力して、この ID のアカウントに代わって E メールを送信することを承認し、追加を選択します。(複数の代理送信者を承認する場合は、各代理送信者に対してこの手順を繰り返します)
8. アクションフィールドで、代理送信者に許可する各送信タイプのチェックボックスをオンにします。
9. (オプション) 代理送信者アクセス許可に限定的なステートメントを追加する場合は、[Specify conditions] (条件を指定) を展開します。
   a. [Operator] (演算子) ドロップダウンから演算子を選択します。
   b. [Key] (キー) ドロップダウンからタイプを選択します。
   c. 選択したキータイプに応じて、その値を値フィールドに入力します。(条件をさらに追加する場合は、[Add new condition] (新しい条件を追加) を選択します。条件を追加するたびに、この手順を繰り返します)
10. [Save statement] (ステートメントを保存) を選択します。
11. (オプション) ポリシーにステートメントを追加する場合は、[Create another statement] (別のステートメントを作成) を展開して、ステップ 6 ~ 10 を繰り返します。

12. [Next] (次へ) を選択すると、[Customize policy] (ポリシーのカスタマイズ) 画面で、[Edit policy details] (ポリシーの詳細の編集) コンテナには、ポリシーの [Name] (名前) と [Policy document] (ポリシードキュメント) 自体を変更またはカスタマイズするためのフィールドが表示されます。

13. [Next] (次へ) をクリックすると、[Review and apply] (確認して適用) 画面の [Overview] (概要) コンテナには、代理送信者に対して許可している検証済み ID と、このポリシーの名前が表示されます。[Policy document] (ポリシードキュメント) ペインには、追加した条件とともに、作成した実際のポリシーが表示されます。ポリシーを確認し、内容が正しい場合は、[Apply policy] (ポリシーの適用) をクリックします。(変更や修正の必要がある場合は、[Previous] (戻る) をクリックして、[Edit policy details] (ポリシーの詳細の編集) コンテナで作業を行います) 作成したポリシーにより、代理送信者がお客様の代わりに送信できるようになります。

14. (オプション) 代理送信者が所有している SNSトピックを使用したり、バウンスや苦情を受け取ったとき、または E メールが配信されたときにフィードバック通知を受け取ったりする場合は、この検証済み ID でSNSトピックを設定する必要があります。(代理送信者はトピック SNS を共有する必要があります) ARN。[Notifications] (通知) タブを選択し、[Feedback notifications] (フィードバック通知) コンテナの [Edit] (編集) を選択します。

    a. SNS トピックの設定ペインのフィードバックフィールド (バウンス、苦情、配信) のいずれかで、SNS所有していないトピックを選択し、代理送信者が所有および共有しているSNSトピックARNを入力します。(代理送信者のみがこれらの通知を受け取ります。これは、代理送信者がSNSトピックを所有しているためです。ID 所有者は、この通知を受け取りません。)

    b. (オプション) トピック通知に元の E メールのヘッダーを含める場合は、各フィードバックタイプのSNSトピック名の直下にある「元の E メールヘッダーを含める」チェックボックスをオンにします。このオプションは、関連付けられた通知タイプに Amazon SNSトピックを割り当てた場合にのみ使用できます。元の E メールヘッダーの内容については、通知の内容の `mail` オブジェクトを参照してください。

    c. [Save changes] (変更の保存) をクリックします。通知設定に対する変更は、反映されるまでに数分かかる場合があります。

    d. (オプション) 代理送信者はバウンスや苦情に関する Amazon SNSトピック通知を受け取るため、この ID の送信に関するフィードバックを受け取りたくない場合は、E メール通知を完全に無効にできます。バウンスや苦情に関する E メールフィードバックを無効にするには、[Notifications] (通知) タブの [Email Feedback Forwarding] (E メールのフィードバック転送) コンテナで、[Edit] (編集) を選択し、[Enabled] (有効) ボックスのチェックを外して、[Save changes] (変更の保存) を選択します。配信ステータス通知は、代理送信者が所有するSNSトピックにのみ送信されるようになりました。

カスタム送信承認ポリシーの作成

カスタム送信承認ポリシーを作成して、アイデンティティにアタッチする場合、次のオプションがあります。

- Amazon の使用 SES API – テキストエディタでポリシーを作成し、Amazon Simple Email Service APIリファレンスでPutIdentityPolicyAPI説明されている を使用してポリシーをアイデンティティにアタッチします。
- Amazon SESコンソールの使用 – テキストエディタでポリシーを作成し、Amazon SESコンソールのカスタムポリシーエディタに貼り付けてアイデンティティにアタッチします。以下の手順では、この方法について説明します。

カスタムポリシーエディタを使用してカスタム送信承認ポリシーを作成するには

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [設定] で、[ID] を選択します。
3. [Verified identities] (検証済み ID) 画面の [Identities] (ID) コンテナで、代理送信者がお客様の代わりに送信することを許可する検証済み ID を選択します。
4. 前のステップで選択した検証済み ID の詳細画面で、[Authorization] (承認) タブを選択します。
5. [Authorization policies] (承認ポリシー) ペインで、[Create policy] (ポリシーの作成) を選択して、ドロップダウンから [Create custom policy] (カスタムポリシーの作成) を選択します。
6. ポリシードキュメントペインで、ポリシーのテキストを JSON形式で入力または貼り付けます。Policy Generator を使用すると、基本的な構造のポリシーをすばやく作成でき、ここでカスタマイズすることもできます。
7. [ポリシーを適用] を選びます。(カスタムポリシーを変更する必要がある場合は、[Authorization] (承認) タブのチェックボックスをオンにし、[Edit] (編集) を選択して、[Policy document] (ポリシードキュメント) ペインで変更を行ってから、[Save changes] (変更の保存) を選択します)
8. (オプション) 代理送信者が所有している SNSトピックを使用したり、バウンスや苦情を受け取ったとき、または E メールが配信されたときにフィードバック通知を受け取ったりする場合は、この検証済み ID でSNSトピックを設定する必要があります。(代理送信者はトピック SNS を共有する必要があります) ARN。[Notifications] (通知) タブを選択し、[Feedback notifications] (フィードバック通知) コンテナの [Edit] (編集) を選択します。

a. SNS トピックの設定ペインのフィードバックフィールド (バウンス、苦情、配信) のいずれかで、SNS所有していないトピックを選択し、代理送信者が所有および共有しているSNSトピックARNを入力します。(代理送信者のみがこれらの通知を受け取ります。これは、代理送信者がSNSトピックを所有しているためです。ID 所有者は、この通知を受け取りません。)

b. (オプション) トピック通知に元の E メールのヘッダーを含める場合は、各フィードバックタイプのSNSトピック名の直下にある「元の E メールヘッダーを含める」チェックボックスをオンにします。このオプションは、関連付けられた通知タイプに Amazon SNSトピックを割り当てた場合にのみ使用できます。元の E メールヘッダーの内容については、通知の内容の `mail` オブジェクトを参照してください。

c. [Save changes] (変更の保存) をクリックします。通知設定に対する変更は、反映されるまでに数分かかる場合があります。

d. (オプション) 代理送信者はバウンスや苦情に関する Amazon SNSトピック通知を受け取るため、この ID の送信に関するフィードバックを受け取りたくない場合は、E メール通知を完全に無効にできます。バウンスや苦情に関する E メールフィードバックを無効にするには、[Notifications] (通知) タブの [Email Feedback Forwarding] (E メールのフィードバック転送) コンテナで、[Edit] (編集) を選択し、[Enabled] (有効) ボックスのチェックを外して、[Save changes] (変更の保存) を選択します。配信ステータス通知は、代理送信者が所有するSNSトピックにのみ送信されるようになりました。

送信ポリシーの例

送信承認では、代理送信者に代理送信を許可する際の細かい条件を指定することができます。

次の条件と例は、送信のさまざまな側面をコントロールするポリシーの作成方法を示しています。

- 送信承認に固有の条件
- 代理送信者の指定
- 「From」アドレスの制限
- 代理送信者による E メール送信時間に対する制限
- E メール送信アクションの制限
- E メール送信者の表示名の制限
- 複数のステートメントの使用

送信承認に固有の条件

条件は、ステートメントのアクセス権限に関する制限のことです。ステートメントの中でも、記述が最も詳細になるのが、この条件部分です。キーは、リクエストの日時など、アクセス制限に使用される基本項目です。

制限は、条件とキーの両方を使用して定義します。たとえば、代理送信者が 2019 年 7 月 30 日以降はお客様の代わりに Amazon SES にリクエストを行うことができないように制限する場合、`DateLessThan` という条件を使用します。キーは `aws:CurrentTime` を使用し、値を `2019-07-30T00:00:00Z` に設定します。

IAM ユーザーガイドの「使用可能なキー」にリストされている AWS 全体のキーを使用するか、送信承認ポリシーで役立つ SES に固有の次のいずれかのキーを使用できます。

条件キー	説明
`ses:Recipients`	受取人アドレスを制限します。To:、"CC"、"BCC" アドレスが含まれます。
`ses:FromAddress`	「From」アドレスを制限します。
`ses:FromDisplayName`	「From」表示名として使用される文字列の内容を制限します ("フレンドリ名" と呼ばれることもあります)。たとえば、"John Doe <johndoe@example.com>" の表示名は John Doe です。
`ses:FeedbackAddress`	「Return Path」アドレス (E メールのフィードバック転送によりバウンスや苦情を送信できるアドレス) を制限します。E メールのフィードバック転送の詳細については、「E メールを介したAmazon SES に関する通知の受信」を参照してください。

Amazon SES キーで `StringEquals` 条件および `StringLike` 条件を使用することができます。これらの条件は、大文字小文字を区別する文字列の一致を指定するために使用します。`StringLike` の場合、値には、複数文字一致のワイルドカード (*) または 1 文字一致のワイルドカード (?) を指定できます。文字列のどこにでも含めることができます。たとえば、次の条件は、先頭に「invoicing」、末尾に「@example.com」が付いた「From」アドレスからのみ代理送信者が送信できることを指定します。

```
"Condition": {
    "StringLike": {
      "ses:FromAddress": "invoicing*@example.com"
    }
}
```

また、StringNotLike 条件を使用して、代理送信者が特定の E メールアドレスから E メールを送信できないようにすることもできます。例えば、admin@example.com や同様の E メールアドレス ("admin"@example.com、admin+1@example.com、sender@admin.example.com) からの送信を禁止するには、ポリシーステートメントに次の条件を含めます :

```
"Condition": {
    "StringNotLike": {
      "ses:FromAddress": "*admin*example.com"
    }
 }
```

条件の指定方法の詳細については、IAM ユーザーガイド の「IAM JSON ポリシーエレメント: 条件」を参照してください。

代理送信者の指定

プリンシパル 、アクセス許可を付与するエンティティ は AWS アカウント 、AWS Identity and Access Management (IAM) ユーザー、またはAWSのサービスです。

以下の例は、検証済みAWSID example.com (888888888888AWS アカウントが所有) からのメール送信を ID 123456789012に許可する、シンプルなポリシーを示しています。このポリシーの Condition ステートメントにより、marketing+.*@example.com アドレスからメールを送信できるのが代理送信者 (AWS ID 123456789012) に限定されています。.* は、送信者が marketing+ の後に追加する任意の文字列を示しています。

```
{
  "Id":"SampleAuthorizationPolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeMarketer",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:888888888888:identity/example.com",
      "Principal":{
```

```
        "AWS":[
          "123456789012"
        ]
      },
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Condition":{
        "StringLike":{
          "ses:FromAddress":"marketing+.*@example.com"
        }
      }
    }
  ]
}
```

次のポリシー例では、IAM ユーザー 2人に、アイデンティティ example.com から送信するアクセス許可を付与します。IAM ユーザーは、自分の Amazon リソースネーム (ARN) によって指定されます。

```
{
  "Id":"ExampleAuthorizationPolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeIAMUser",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:888888888888:identity/example.com",
      "Principal":{
        "AWS":[
          "arn:aws:iam::111122223333:user/John",
          "arn:aws:iam::444455556666:user/Jane"
        ]
      },
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ]
    }
  ]
}
```

次のポリシー例では、 Amazon Cognito に、アイデンティティ example.com から送信するアクセス許可を付与します。

```
{
  "Id":"ExampleAuthorizationPolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeService",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:888888888888:identity/example.com",
      "Principal":{
        "Service":[
          "cognito-idp.amazonaws.com"
        ]
      },
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "888888888888",
          "aws:SourceArn": "arn:aws:cognito-idp:us-east-1:888888888888:userpool/your-
user-pool-id-goes-here"
        }
      }
    }
  ]
}
```

次の例のポリシーでは、AWSOrganization 内のすべてのアカウントに、identity example.com から送信するアクセス許可が付与されます。AWSOrganization は、PrincipalOrgIDグローバル条件キーにより指定されます。

```
{
  "Id":"ExampleAuthorizationPolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeOrg",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:888888888888:identity/example.com",
```

```
      "Principal":"*",
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Condition":{
        "StringEquals":{
          "aws:PrincipalOrgID":"o-xxxxxxxxxx"
        }
      }
    }
  ]
}
```

「From」アドレスの制限

検証済みドメインを使用する場合、代理送信者だけが特定の E メールアドレスから送信できるようにするポリシーを作成することをお勧めします。"From" アドレスを制限するには、キーで ses:FromAddress と呼ばれる条件を設定します。次のポリシーでは、AWS アカウントID 123456789012 がIDexample.com から送信することを許可すると同時に、送信元の E メールアドレスをsender@example.com に限定しています。

```
{
  "Id":"ExamplePolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeFromAddress",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:888888888888:identity/example.com",
      "Principal":{
        "AWS":[
          "123456789012"
        ]
      },
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Condition":{
        "StringEquals":{
          "ses:FromAddress":"sender@example.com"
        }
```

```
      }
    }
  ]
}
```

代理送信者による E メール送信時間に対する制限

送信者の承認ポリシーを設定することで、代理送信者が E メールを送信できる日付と時間や、日付の範囲を限定することもできます。例えば、2021 年 9 月内で E メールキャンペーンの送信を予定している場合、次のポリシーを使用することで、代理送信者による E メール送信を、その月に限定することができます。

```
{
  "Id":"ExamplePolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"ControlTimePeriod",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:888888888888:identity/example.com",
      "Principal":{
        "AWS":[
          "123456789012"
        ]
      },
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Condition":{
        "DateGreaterThan":{
          "aws:CurrentTime":"2021-08-31T12:00Z"
        },
        "DateLessThan":{
          "aws:CurrentTime":"2021-10-01T12:00Z"
        }
      }
    }
  ]
}
```

E メール送信アクションの制限

送信者が Amazon SES で E メールを送信するために使用できるアクションは、SendEmail と SendRawEmail の 2 つです。送信者が E メールの形式をどの程度コントロールするかに応じて決定します。送信承認ポリシーでは、代理送信者をこれらの 2 つのアクションのいずれかに制限できます。ただし、多くのアイデンティティ所有者は、ポリシーで両方のアクションを有効にすることで、E メール送信呼び出しの詳細を代理送信者に任せています。

> **Note**
>
> 代理送信者が SMTP インターフェイスを介して Amazon SES にアクセスできるようにする場合、少なくとも SendRawEmail を選択する必要があります。

アクションを制限する必要がある場合、送信承認ポリシーにどちらかのアクションのみ含めることで制限することができます。次の例は、アクションを SendRawEmail に制限する方法を示しています。

```
{
  "Id":"ExamplePolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"ControlAction",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:888888888888:identity/example.com",
      "Principal":{
        "AWS":[
          "123456789012"
        ]
      },
      "Action":[
        "ses:SendRawEmail"
      ]
    }
  ]
}
```

E メール送信者の表示名の制限

E メールクライアントによっては、実際の「From」アドレスではなく、E メール送信者の「フレンドリ」名 (E メールヘッダーで指定されている場合) が表示されます。たとえば、"John Doe <johndoe@example.com>" の表示名は John Doe です。例として、user@example.com から E メールを送信しますが、E メールが user@example.com ではなく「Marketing」から送信されたものとして受信者に表示したいとします。次のポリシーでは、「From」アドレスの表示名に「Marketing」が含まれている場合のみ、AWS アカウントID 123456789012 がIDexample.com から送信できるようにします。

```
{
  "Id":"ExamplePolicy",
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeFromAddress",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:888888888888:identity/example.com",
      "Principal":{
        "AWS":[
          "123456789012"
        ]
      },
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Condition":{
        "StringLike":{
          "ses:FromDisplayName":"Marketing"
        }
      }
    }
  ]
}
```

複数のステートメントの使用

送信承認ポリシーには複数のステートメントを含めることができます。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、「From」アドレスとフィードバックアドレスの両方がドメイン example.com を使用している場合に限り、2 つのAWS アカウントがsender@example.comから送信することを承認します。2 番目のステートメントは、

受信者の E メールアドレスが example.com ドメインに属している場合に限り、IAM ユーザーが sender@example.com から E メールを送信することを承認します。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AuthorizeAWS",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:999999999999:identity/sender@example.com",
      "Principal":{
        "AWS":[
          "111111111111",
          "222222222222"
        ]
      },
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Condition":{
        "StringLike":{
          "ses:FromAddress":"*@example.com",
          "ses:FeedbackAddress":"*@example.com"
        }
      }
    },
    {
      "Sid":"AuthorizeInternal",
      "Effect":"Allow",
      "Resource":"arn:aws:ses:us-east-1:999999999999:identity/sender@example.com",
      "Principal":{
        "AWS":"arn:aws:iam::333333333333:user/Jane"
      },
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Condition":{
        "ForAllValues:StringLike":{
          "ses:Recipients":"*@example.com"
        }
      }
```

```
    }
  ]
}
```

Amazon SESの送信承認のための ID 情報を代理送信者に提供する

送信承認ポリシーを作成してアイデンティティにアタッチしたら、代理送信者にアイデンティティの Amazon リソースネーム (ARN) を提供できます。代理送信者は、E SESメール送信オペレーションまたは E メールのヘッダーで Amazon ARNにこれを渡します。ID の を検索するにはARN、次の手順に従います。

ID ARNの を検索するには

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの設定で、検証済み ID を選択します。
3. ID のリストで、送信承認ポリシーをアタッチしたアイデンティティを選択します。
4. 概要ペインの 2 番目の列 Amazon リソースネーム (ARN) には、アイデンティティの が含まれますARN。arn:aws:ses:us-east-1:123456789012:identity/user@example.com のような内容です。全体をコピーARNし、代理送信者に渡します。

> **Important**
>
> 承認する ID がグローバルエンドポイント機能の一部としてセカンダリリージョンで複製されている場合は、 などのリージョンパラメータをus-east-1、次の例*のようにアスタリスク に置き換えますarn:aws:ses:*:123456789012:identity/user@example.com。

Amazon SESの送信承認を行うための代理送信者のタスク

代理送信者は、自分が所有していないが使用を許可された ID の代わりに E メールを送信します。ID 所有者の代わりに送信しますが、バウンスと苦情は自分のAWSアカウントのバウンスと苦情のメトリクスに対してカウントされ、メッセージ送信数は自分の送信クォータに対してカウントされます。ID所有者の E メールを送信するために送信クォータの増加が必要になった場合は、それをリクエストする責任も自分にあります。

代理送信者として、次のタスクを完了する必要があります。

- ID所有者への情報提供
- 代理送信者通知の使用
- ID 所有者の E メールの送信

Amazon SESの送信承認を行うためのID所有者への情報提供

代理送信者は ID 所有者の代わりに E メールを送信するため、自分の AWS アカウント ID または (IAM) ユーザーの Amazon リソース名 (ARN) を ID 所有者に提供する必要があります。ID 所有者は、検証済みの ID のいずれかから送信するためのアクセス許可を付与するポリシーを作成するために、ユーザーのアカウント情報を必要とします。

独自の SNS トピックを使用する場合は、バウンス、苦情、または配信に関するフィードバック通知が 1 つ以上の SNS トピックに送信されるように設定することを、ID 所有者にリクエストできます。これを行うには、SNS トピックを ID 所有者が送信を許可した検証済み ID で設定できるように、SNS トピック ARN を ID 所有者と共有する必要があります。

次の手順では、ID 所有者と共有するアカウント情報および SNS トピック ARN を検索する方法について説明します。

AWS アカウント ID を確認するには

1. AWS Management Console (https://console.aws.amazon.com) にサインインします。
2. コンソールの右上隅で名前/アカウント番号を展開し、ドロップダウンで [My Account] (マイアカウント) を選択します。
3. アカウント設定ページが開き、AWS アカウント ID を含むすべてのアカウント情報が表示されます。

IAM ユーザー ARN を検索するには

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで [Users] を選択します。
3. ユーザーのリストで、ユーザー名を選択します。[Summary] (概要) セクションに IAM ユーザー ARN が表示されます。ARN は次の例のようになります。arn:aws:iam::123456789012:user/John。

SNSトピック ARN を検索するには

1. Amazon SNS コンソール (https://console.aws.amazon.com/sns/v3/home) を開きます。
2. ナビゲーションペインで、[トピック] を選択します。
3. トピックのリストでは、SNS トピック ARN が ARN 列に表示されます。ARN は次の例のようになります。arn:aws:sns:us-east-1:444455556666:my-sns-topic。

Amazon SESの送信承認を行うための代理送信者通知の使用

代理送信者は、自分が所有していないが使用を許可された ID の代わりに E メールを送信します。ただし、バウンスと苦情は、ID 所有者ではなく代理送信者のバウンスと苦情のメトリクスに対してカウントされます。

アカウントのバウンス率や苦情率が高くなりすぎる場合、アカウントが確認対象となり、アカウントのEメール送信機能を一時停止されるおそれがあります。したがって、通知を設定したり、通知を監視するためのプロセスを備えることが重要です。バウンスや苦情が発生しているアドレスをメーリングリストから削除するためのプロセスを備えることも重要です。

したがって、代理送信者は、自分が所有していないが、ID 所有者によって使用が許可されている ID の代わりに送信した E メールに対してバウンスまたは苦情イベントが発生した場合に、通知を送信するように Amazon SES を設定できます。Amazon SNS または Firehose にバウンスや苦情の通知を発行するようにイベント発行をセットアップすることもできます。

> **Note**
>
> Amazon SNS を使用して通知を送信するように Amazon SES を設定すると、受け取る通知に対して Amazon SNS の標準レートが課金されます。詳細については、「Amazon SNS 料金」ページを参照してください。

新しい代理送信者通知を作成

代理送信通知は、イベントパブリッシングを使用する設定セットか、独自の SNS トピックで設定された検証済みの ID を使用して設定できます。

次のいずれかの方法を使用して、新しい代理送信通知を設定するための手順を以下に示します。

- 設定セットを使用したイベント発行
- 所有する SNS トピックへのフィードバック通知

代理送信用の設定セットを使用してイベント発行をセットアップするには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 「イベント送信先の作成」の手順に従います。
3. 設定セットでイベント発行をセットアップした後、ID 所有者が送信を許可した検証済み ID を使用して、代理送信者として E メールを送信するときに、設定セットの名前を指定します。「ID 所有者の E メールの送信」を参照してください。

代理送信用に所有する SNS トピックへのフィードバック通知を設定するには

1. フィードバック通知に使用する SNS トピックを決定したら、手順に従って、SNS トピック ARN を検索し、完全な ARN をコピーして、ID 所有者と共有します。
2. ID 所有者が送信を許可した共有 ID でフィードバック通知用の SNS トピックを設定するように、ID 所有者に依頼します。(ID 所有者は、承認ポリシー手順にある SNS トピックの設定用に指定された手順に従う必要があります)

Amazon の送信承認のための ID 所有者の E メールSESの送信

代理送信者は、他の Amazon SES送信者と同じ方法で E メールを送信します。ただし、ID 所有者が使用を許可した ID の Amazon リソースネーム (ARN) を指定する点が異なります。Amazon SESを呼び出して E メールを送信すると、Amazon は指定した ID に、送信を許可するポリシーがあるSESかどうかを確認します。

E メールを送信ARNするときに ID を指定する方法はさまざまです。使用する方法は、Amazon SESAPIオペレーションと Amazon SESSMTPインターフェイスのどちらを使用して E メールを送信するかによって異なります。

> **Important**
>
> - E メールを正常に送信するには、ID 所有者が ID を検証した AWS リージョンの Amazon SESエンドポイントに接続する必要があります。
> - さらに、いずれかの AWS アカウントが検証されていないアドレスに E メールを送信する前に、ID 所有者と代理送信者の両方のアカウントをサンドボックスから削除する必要があります。詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。

- 使用が許可されている ID がグローバルエンドポイント機能の一部としてセカンダリリージョンで複製されている場合：
  - ID 所有者は、次の例*のようにus-east-1、 などのリージョンパラメータをアスタリスクに置き換えARNた ID を指定している必要がありますarn:aws:ses:*:123456789012:identity/user@example.com。
  - ID 所有者は、プライマリリージョンとセカンダリリージョンの両方で送信承認ポリシーを作成しておく必要があります。

Amazon の使用 SES API

他の Amazon E SESメール送信者と同様に、Amazon SES経由で SES API (を通じて直接HTTPSまたは間接的に AWS SDK) Amazon にアクセスする場合SendEmail、、SendTemplatedEmail、 の 3 つの E メール送信アクションのいずれかを選択できますSendRawEmail。Amazon Simple Email Service APIリファレンスでは、これらの の詳細について説明していますがAPIs、ここでは送信承認パラメータの概要を示します。

SendRawEmail

SendRawEmail を使用して E メールの形式をコントロールできるようにする場合、次の 2 つの方法のいずれかを使用して委任された承認済み ID を指定できます。

- オプションパラメータを **SendRawEmail** に渡しますAPI。必須のパラメーターを次の表で説明します。

Parameter	説明
SourceArn	ARN の Sourceパラメータで指定された E メールアドレスに送信することを許可する送信承認ポリシーに関連付けられている ID の SendRawEmail 。 **Note** のみを指定した場合SourceArn 、Amazon は「From」アドレスと「Return Path」アドレスを、 で指定した ID SESに設定しますSourceArn 。

Parameter	説明
FromArn	raw E メールARNの ヘッダーで特定の「From」アドレスを指定することを許可する送信承認ポリシーに関連付けられている ID の 。
ReturnPathArn	ARN の ReturnPath パラメータで指定された E メールアドレスの使用を許可する送信承認ポリシーに関連付けられている ID の SendRawEmail 。

- E メールに X ヘッダーを含める。X ヘッダーは、標準的な E メールヘッダーに加えて使用できるカスタムヘッダーです (「From」、「Reply-To」、または「Subject」ヘッダーなど)。Amazon は、送信承認パラメータを指定するために使用できる 3 つの X ヘッダーSESを認識します。

> **Important**
>
> これらの X ヘッダーは、Amazon によって E メールを送信SESする前に削除されるため、DKIM署名に含めないでください。

X ヘッダー	説明
X-SES-SOURCE-ARN	SourceArn に対応します。
X-SES-FROM-ARN	FromArn に対応します。
X-SES-RETURN-PATH-ARN	ReturnPathArn に対応します。

Amazon は、送信する前に E メールからすべての X ヘッダーSESを削除します。X ヘッダーの複数のインスタンスを含めると、Amazon は最初のインスタンスのみSESを使用します。

次の例は、送信認可 X ヘッダーを含む E メールを示しています。

```
X-SES-SOURCE-ARN: arn:aws:ses:us-east-1:123456789012:identity/example.com
X-SES-FROM-ARN: arn:aws:ses:us-east-1:123456789012:identity/example.com
X-SES-RETURN-PATH-ARN: arn:aws:ses:us-east-1:123456789012:identity/example.com

From: sender@example.com
```

```
To: recipient@example.com
Return-Path: feedback@example.com
Subject: subject
Content-Type: multipart/alternative;
 boundary="----=_boundary"

------=_boundary
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 7bit

body
------=_boundary
Content-Type: text/html; charset=UTF-8
Content-Transfer-Encoding: 7bit

body
------=_boundary--
```

SendEmail and SendTemplatedEmail

SendEmail または SendTemplatedEmail オペレーションを使用する場合、次のオプションパラメータを渡すことで委任された承認済み ID を指定できます。SendEmail または SendTemplatedEmail オペレーションを使用する場合、X ヘッダーは使用できません。

パラメータ	説明
SourceArn	SendEmail または ARNの Sourceパラメータで指定された E メールアドレスの送信を許可する送信承認ポリシーに関連付けられている ID の SendTemplatedEmail 。
ReturnPathArn	SendEmail または ARNの ReturnPath パラメータで指定された E メールアドレスの使用を許可する送信承認ポリシーに関連付けられている ID の SendTemplatedEmail 。

次の例は、 SendEmailまたは SendTemplatedEmailオペレーションSourceArnと for Python を使用して、 属性と ReturnPathArn 属性を含む E メールを送信する方法を示しています。 SDK

```
import boto3
from botocore.exceptions import ClientError

# Create a new SES resource and specify a region.
client = boto3.client('ses',region_name="us-east-1")

# Try to send the email.
try:
    #Provide the contents of the email.
    response = client.send_email(
        Destination={
            'ToAddresses': [
                'recipient@example.com',
            ],
        },
        Message={
            'Body': {
                'Html': {
                    'Charset': 'UTF-8',
                    'Data': 'This email was sent with Amazon SES.',
                },
            },
            'Subject': {
                'Charset': 'UTF-8',
                'Data': 'Amazon SES Test',
            },
        },
        SourceArn='arn:aws:ses:us-east-1:123456789012:identity/example.com',
        ReturnPathArn='arn:aws:ses:us-east-1:123456789012:identity/example.com',
        Source='sender@example.com',
        ReturnPath='feedback@example.com'
    )
# Display an error if something goes wrong.
except ClientError as e:
    print(e.response['Error']['Message'])
else:
    print("Email sent! Message ID:"),
    print(response['ResponseMetadata']['RequestId'])
```

Amazon SESSMTPインターフェイスの使用

代理送信に Amazon SESSMTPインターフェイスを使用する場合は、、X-SES-FROM-ARN、および X-SES-SOURCE-ARNX-SES-RETURN-PATH-ARNヘッダーをメッセージに含める必要があります。SMTP 会話で DATA コマンドを発行した後、これらのヘッダーを渡します。

シミュレータSESーを使用して Amazon でテスト E メールを送信する

Amazon SESコンソールを使用して、Amazon でテストメールを送信することをお勧めしますSES。ただしコンソールでは手動で情報を入力する必要があるため、通常、テストメールの送信にのみ使用します。Amazon の使用を開始したらSES、Amazon SESSMTPインターフェイスまたは を使用して E メールを送信することがほとんどですAPI。ただし、コンソールは送信アクティビティを監視するのに役立ちます。

以下のトピックでは、コンソールからと手動で E メールを送信の両方でメールボックスシミュレーターを使用する方法を説明します。

- コンソールからメールボックスシミュレーターを使用する
- 手動でメールボックスシミュレーターを使用する

コンソールからメールボックスシミュレーターを使用する

> **Important**
>
> - このチュートリアルでは、受信を確認できるようにコンソールから自分宛に E メールを送信します。さらに詳しい実験や負荷テストについては、「手動でメールボックスシミュレーターを使用する」を参照してください。
> - メールボックスシミュレーターに送信される E メールは送信クォータに加算されず、バウンス率や苦情率の計算にも含まれません。また、Virtual Deliverability Manager のメトリクスにも影響しません。

以下のステップを実行する前に、「Amazon Simple Email Service を設定する」のタスクを完了してください

Amazon SESコンソールからテスト E メールメッセージを送信するには

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの「設定」で「アイデンティティ」を選択します。
3. [Identities] (ID) テーブルで、検証済み E メール ID を選択します (チェックボックスをオンにするのではなく、ID の名前を直接クリックします)。検証済みメールアドレス ID がない場合は、「Eメールアドレス ID の作成」を参照してください。
4. 選択した E メール ID の詳細ページで、[Send test email] (テストメールを送信する) を選択します。
5. メッセージの詳細で、E メールの形式を選択します。次のように、2 つの選択肢があります。

   - フォーマット済み – これは最も簡単な選択肢です。[Body] テキストボックスにメッセージのみを入力する場合は、このオプションを選択します。E メールを送信すると、Amazon はテキストを E SES メール形式にします。
   - Raw - を含むメッセージHTMLや添付ファイルなど、より複雑なメッセージを送信する場合は、このオプションを選択します。このような柔軟性により、Amazon SES API v2 を使用した raw E メールの送信に記載されているとおり、メッセージを各自でフォーマットし、次にフォーマット済みメッセージの全体 (ヘッダーも含む) を [Body] テキストボックスに貼り付けます。を含む次の例を使用して、Raw E メール形式を使用してHTMLテスト E メールを送信できます。このメッセージをコピーし、[Body] テキストボックスにすべてを貼り付けます。MIME-Version ヘッダーと Content-Typeヘッダーの間に空白行がないことを確認します。これらの 2 行の間に空白行があると、E メールは ではなくプレーンテキストとしてフォーマットされますHTML。

```
Subject: Amazon SES Raw Email Test
MIME-Version: 1.0
Content-Type: text/html

<!DOCTYPE html>
<html>
<body>
<h1>This text should be large, because it is formatted as a header in HTML.</h1>
<p>Here is a formatted link: <a href="https://docs.aws.amazon.com/ses/latest/
DeveloperGuide/Welcome.html">Amazon Simple Email Service Developer Guide</a>.</p>
</body>
</html>
```

6. テストするシミュレートされた E メールシナリオのタイプを、[Scenario] (シナリオ) リストボックスを展開して選択します。

   - カスタム を選択し、Amazon SESサンドボックスにまだいる場合は、カスタム受信者フィールドのアドレスが検証済みの E メールアドレスであることを確認してください。詳細については、「Eメールアドレス ID の作成」を参照してください。

7. 必要に応じて、残りのフィールドに入力します。
8. [Send test email] (テストメール送信) を選択します。
9. E メールの宛先の E メールクライアントにサインインします。送信した E メールメッセージを確認します。

手動でメールボックスシミュレーターを使用する

Amazon SES には、アプリケーションがさまざまな E メール送信シナリオをどのように処理するかをテストするために使用できるメールボックスシミュレーターが含まれています。メールボックスシミュレーターは、架空の E メールアドレスを作成せずに E メール送信アプリケーションをテストしたり、1 日あたりの送信クォータに影響を与えずにシステムの最大スループットを確認したりする場合に役立ちます。

重要な考慮事項

Amazon SESメールボックスシミュレーターを使用する場合は、次の機能と制限を考慮してください。

- アカウントが Amazon SESサンドボックスにある場合でも、メールボックスシミュレーターを使用できます。
- メールボックスシミュレーターに送信する E メールは、アカウントの最大送信レートのクォータを受けますが、毎日の送信クォータには影響しません。たとえば、アカウントが 24 時間あたり 10,000 件のメッセージ送信を許可されていて、100 件のメッセージをメールボックスシミュレーターに送信した場合、依然として最大 10,000 件のメッセージを通常の受取人に送信でき、送信クォータに達することはありません。
- メールボックスシミュレーターに送信する E メールは、E メールの配信性能や評価のメトリクスには影響しません。たとえば、大量のメッセージをメールボックスシミュレーターの返送アドレスに送信しても、返送率が高すぎることを警告するメッセージがレピュテーションメトリクスコンソールページに表示されることはありません。

- 請求目的上、Amazon SESメールボックスシミュレーターに送信する E メールは、Amazon を使用して送信する他の E メールと同じですSES。つまり、メールボックスシミュレーターに送信したメッセージに対しても、通常の受取人に送信したメッセージと同額が請求されます。
- メールボックスシミュレーターはラベル付けをサポートしています。これにより、複数の方法で同じメールボックスシミュレーターアドレスに E メールを送信したり、アプリケーションが可変エンベロープリターンパス () をどのように処理するかを確認したりできますVERP。たとえば、bounce+label1@simulator.amazonses.comとbounce+label2@simulator.amazonses.comにEメールを送信して、バウンスメッセージとバウンスを生じた E メールアドレスをアプリケーションが照合できるかどうかを確認できます。
- メールボックスシミュレーターを使用して、同じ送信リクエストからの複数のバウンスをシミュレートする場合、Amazon SESはバウンスレスポンスを 1 つのレスポンスに結合します。

メールボックスシミュレーターの使用

メールシミュレーターを使用するには、以下の表にあるシナリオを見つけ、対応する E メールアドレスに E メールを送信します。

> **Note**
>
> メールボックスシミュレーターアドレスに E メールを送信するときは、、、Amazon SESコンソールSES、 AWS CLI AWS SDKAmazon SESSMTPインターフェイス、または Amazon を使用して、Amazon 経由で E SES メールを送信する必要がありますAPI。メールボックスシミュレーターは、外部ソースから送信された E メールには応答しません。

シミュレートシナリオ	Eメールアドレス
正常な配信 – 受取人のEメールプロバイダーが、Eメールを受領します。「」の説明に従って配信通知を設定するとAmazon SES のイベント通知の設定、Amazon Simple Notification Service (Amazon) を通じて Amazon から配信通知SESが送信されますSNS。	success@simulator.amazonses.com
バウンス — 受信者の E メールプロバイダーは、550 SMTP 5.1.1 (「不明なユーザー」) レスポンスコードを使用して E メールを拒否し	bounce@simulator.amazonses.com

シミュレートシナリオ	Eメールアドレス
ます。Amazon はバウンス通知SESを生成し、アカウントの設定方法に応じて、E メールで送信するか、Amazon SNSトピックに通知を送信します。メールボックスシミュレーターの E メールアドレスは Amazon サSESプレッションリストに表示されません。これは通常、ハードバウンスが発生したときに発生します。メールボックスシミュレーターから受信したバウンスレスポンスは、RFC3464 に準拠しています。返送フィードバックの受け取り方法については、「Amazon SES のイベント通知の設定」を参照してください。	
自動応答 – 受取人の E メールプロバイダーが E メールを受領し、受取人の受信トレイに配信します。E メールプロバイダーは、「不在」(OOTO) メッセージなどの自動応答を E メールの Return-Path ヘッダーのアドレスに送信します。Return-Path ヘッダーが存在しない場合はエンベロープ送信者 (FROMMAIL「」) アドレスに送信します。メールボックスシミュレーターから受信した自動応答は、RFC3834 に準拠しています。	ooto@simulator.amazonses.com

シミュレートシナリオ	Eメールアドレス
苦情 – 受取人のEメールプロバイダーがEメールを受領し、受取人の受信トレイに配信します。受取人は、これを未承諾のメッセージであると判断し、E メールクライアントで [Mark as Spam (スパムとしてマーク)] をクリックします。SES その後、Amazon は、アカウントの設定方法に応じて、E メールまたは Amazon SNSトピックに通知することで、苦情通知を転送します。メールボックスシミュレーターから受け取った苦情レスポンスは、RFC5965 に準拠しています。苦情フィードバックの受け取り方法については、「Amazon SES のイベント通知の設定」を参照してください。	complaint@simulator.amazonses.com
サプレッションリストの受信者アドレス - Amazon は、受信者のアドレスがグローバルサプレッションリストにあるかのようにハードバウンスSESを生成します。	suppressionlist@simulator.amazonses.com

拒否イベントのテスト

Amazon 経由で送信するすべてのメッセージSESは、ウイルスについてスキャンされます。ウイルスを含むメッセージを送信すると、Amazon はメッセージSESを受け入れ、ウイルスを検出し、メッセージ全体を拒否します。Amazon がメッセージSESを拒否すると、メッセージの処理が停止し、受信者のメールサーバーへの配信は試行されません。次に、拒否イベントが生成されます。

Amazon SESメールボックスシミュレーターには、拒否イベントをテストするためのアドレスは含まれていません。ただし、欧州コンピュータウイルス対策研究所 (EICAR) テストファイルを使用して拒否イベントをテストできます。このファイルは、ウイルス対策ソフトウェアを安全な方法でテストする業界標準の手段です。EICAR テストファイルを作成するには、次のテキストをファイルに貼り付けます。

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

そのファイルを`sample.txt`として保存し、Eメールに添付して、そのメールを確認済みアドレスに送信します。E メールに他の問題がない場合、Amazon はメッセージSESを受け入れますが、実際のウイルスが含まれている場合と同様に拒否します。

> **Note**
>
> 拒否された E メール(上記の手順を使用して送信したものを含む) は、1 日あたりの送信クォータに対してカウントされます。送信するメッセージ (拒否されたメッセージを含む) ごとに料金が発生します。

EICAR テストファイルの詳細については、EICARWikipedia のテストファイルページを参照してください。

Amazon SES の設定セットの使用

設定セットは、検証済みの ID に適用できるルールのグループです。検証済みのID とは、Amazon SES 経由で E メールを送信するために使用するドメイン、サブドメインまたはE メールアドレスです。E メールに構成セットを適用すると、その構成セットに含まれるすべてのルールが E メールに適用されます。

設定セットを使用すると、送信する E メールに次のタイプのルールを適用できます。これらのいずれかまたは両方を含めることも、いずれも含めないこともできます。

- イベント送信先 – 送信する E メールごとに、送信数、配信数、開封数、クリック数、バウンス数、苦情数などの E メール送信メトリクスを他の AWS 製品に発行できます。例えば、E メールのメトリクスを Amazon Data Firehose の送信先に送ってから、Amazon Managed Service for Apache Flink を使用して分析できます。または、Amazon SNS にバウンスや苦情の情報を送信し、これらのイベントが発生したときに即座に通知を受け取ることができます。
- IP プールの管理 - 専用の IP アドレスをリースして Amazon SES で使用する場合、専用 IP プールと呼ばれるアドレスのグループを作成して、特定のタイプの E メールを送信するときに使用できます。例えば、専用 IP プールを設定セットに関連付けて、マーケティングコミュニケーションの送信や、取引の E メールの送信に使用できます。取引 E メールの送信者の評価はマーケティング E メールの送信者の評価とは切り離されています。

次の方法で、検証済み ID に設定セットを関連付けます。

- E メールのヘッダーに、設定セットへの参照を含めます。E メールの設定セットの指定の詳細については、メールの送信時に設定セットを指定する を参照してください。
- ID の作成中、または検証済み ID の編集中に使用される、デフォルトの設定セットを指定します。デフォルトの設定セットを理解する を参照してください。

内容

SES での設定セットの作成

SES コンソール、Amazon SES API v2 の `CreateConfigurationSet` アクション、または Amazon SES CLI v2 の `aws sesv2 create-configuration-set` コマンドを使用して、新しい設定セットを作成できます。このセクションでは、SES コンソールおよび Amazon SES CLI v2 を使用して設定セットを作成する方法を示します。

設定セットを作成する (コンソール)

SES コンソールを使用して設定セットを作成するには、次の手順に従います。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/ses/ で Amazon SES コンソールを開きます。
2. 設定のナビゲーションペインで、設定セットを選択します。
3. [Create set] を選択します。
4. 以下の詳細を一般的な詳細セクションに入力します。

   - 設定セット名 – 設定セットの名前です。最大 64 文字の英数字を含めることができます。これには、文字、数字、ハイフン (-)、アンダースコア (_) のみが含まれます。
   - IP プールを送信する – この設定セットを使用して E メールを送信すると、割り当て済みのプール内の専用 IP アドレスからメッセージが送信されます。リストから IP プールを選択します。

     > **Note**
     >
     > デフォルト (ses-default-dedicated-pool) には、他のプールに割り当てられていない専用 IP アドレスが含まれます。IP プールの管理の方法については、IP プールの割り当て を参照してください。

   - 追跡オプション
     - カスタムリダイレクトドメインの使用 – このチェックボックスをオンにすると、カスタムリダイレクトドメインを使用して、この設定セットで送信された E メールの開封とクリックの追跡を処理できます。
     - カスタムリダイレクトドメイン – [検証済みドメインの選択] リストからカスタムリダイレクトドメインとなる検証済みドメインを選択します。[サブドメインの入力] フィールドにサブドメインを入力することもできます。

> **Note**
>
> custom ドメインは、次のように指定できます。
>
> - まず、E メールを送信および追跡 AWS リージョン する でカスタムリダイレクトドメインを作成して検証し、コンテンツ配信ネットワーク (CDN) を設定する必要があります。これは「カスタムドメインを設定してオープンとクリックの追跡を処理します」で説明されています。
> - カスタムリダイレクトドメインを開封とクリックの追跡に使用するには、次に、このステップで設定セットを作成または編集する際に、カスタムリダイレクトドメインを指定する必要があります。
> - 最後に、カスタムリダイレクトドメインの指定後、[DNS レコードの表示] が設定セットの [全般的な詳細] コンテナに表示されます。展開すると、 で使用されている追跡ドメインを含む CNAME レコードが表示されます AWS リージョン。例えば、カスタムサブドメインが marketing.example.com という名前で `us-east-1` の AWS リージョン で作成された場合、[DNS レコードの表示] で展開すると、[名前] = marketing.example.com と [値] = r.us-east-1.awstrack.me の値を持つ CNAME レコードが表示されます。
>
> この情報は、「カスタムドメインを設定してオープンとクリックの追跡を処理します」で説明されているとおり、CDN をセットアップする際に、テーブルから適切な追跡ドメインを選択したことの単なる確認としてのみ使用できます。または、最初にこれを実行し、ここで表示された CNAME レコード値を CDN 設定で使用することもできます。

- HTTPS ポリシー – カスタムリダイレクトドメインの開封とクリックの追跡リンクのプロトコルに HTTPS ポリシーオプションを選択します。
  - オプション – (デフォルト動作) 開封の追跡リンクは HTTP を使用してラップされます。クリックの追跡リンクは、リンクの元のプロトコルを使用してラップされます。
  - 必須 – 開封とクリックの追跡リンクはどちらも HTTPS を使用してラップされます。
  - 開封に必要 – 開封の追跡リンクは HTTPS を使用してラップされます。クリックの追跡リンクは、リンクの元のプロトコルを使用してラップされます。
- 高度な配信オプション – 左側の矢印を選択すると、[高度な配信オプション] セクションを展開できます。

- Transport Layer Security (TLS) – SES が受信側のメールサーバーとの安全な接続を確立し、TLS プロトコルを使用して E メールを送信することを求める場合は、[必須] チェックボックスをオンにします。

> **Note**
>
> SES は TLS 1.2 をサポートし、TLS 1.3 を推奨しています。詳細については、「SES のインフラストラクチャセキュリティ」を参照してください。

- 最大配信期間 – SES がこの設定セットを使用して E メール配信を試行する時間制限を指定するには、300 ~ 50,400 の値を秒単位で入力します。

> **Note**
>
> カスタムの最大配信制限 (SES のデフォルトである 14 時間より短い期間) を設定すると、時間的制約のある E メール (ワンタイムパスワードを記載したメールなど)、トランザクション E メール、営業時間外に配信されないようにする E メールなどに役立ちます。
>
> > **Tip**
> >
> > - 分から秒の計算には、60 を掛けます。例えば、7 分 * 60 = 420 秒です。
> > - 時間から秒の計算には、3,600 を掛けます。例えば、2 時間 * 3,600 = 7,200 秒です。

5. 以下の詳細を評判のオプションセクションに入力します。

- 評価メトリクス – この設定セットを使用して送信されたメールについて、CloudWatch でバウンスと苦情のメトリクスを追跡するために使用されます。(追加料金が適用されます。「CloudWatch のメトリクスあたりの料金」を参照してください。)
- 有効 – 設定セットで評価メトリクスを有効にするには、このチェックボックスをオンにします。

6. [Suppression list options] (サプレッションリスト) のオプションセクションには、この設定セットによりアカウントレベルの抑制を上書きするオプションなどを含む決定セットが用意されています。設定セットレベルのサプレッションロジックマップによって、オーバーライドが組み合わ

さった結果が、どのように影響するのかが分かります。これらの多層オーバーライドを選択し組み合わせることで、3 つの異なるレベルの抑制を実装できます。

a. アカウントレベルのサプレッションを使用してください。アカウントレベルの抑制を上書きしたり、設定セットレベルの抑制を実装したりしないでください。基本的に、この設定セットを使用して送信される E メールは、アカウントレベルの抑制のみを使用します。これを実行するには:

   - [Suppression list settings] (サプレッションリストの設定) で、[Override account level settings] (アカウントレベルの設定を上書きする) ボックスのチェックを外します。

b. 抑制は使用しないでください。設定セットレベルの抑制は有効にせずに、アカウントレベルの抑制を上書きします。この設定セットを使用して送信される E メールは、アカウントレベルの抑制を使用しないことになります。言い換えると、すべての抑制がキャンセルされます。これを実行するには:

   i. [Suppression list settings] (サプレッションリストの設定) で、[Override account level settings] (アカウントレベルの設定を上書きする) ボックスにチェックを付けます。

   ii. [Suppression list] (サプレッションリスト) で、[Enabled] (有効) ボックスのチェックを外します。

c. 設定セットレベルの抑制を使用してください。この設定セットで定義されたカスタムのサプレッションリストの設定を使用して、アカウントレベルの抑制を上書きします。この設定セットを使用して送信される E メールは、独自の抑制設定のみを使用することになり、アカウントレベルの抑制設定は無視されます。これを実行するには:

   i. [Suppression list settings] (サプレッションリストの設定) で、[Override account level settings] (アカウントレベルの設定を上書きする) ボックスにチェックを付けます。

   ii. [Suppression list] (サプレッションリスト) で [Enabled] (有効) にチェックを入れます。

   iii. [Specify the reason(s)...] (理由を指定) で、この設定セットで使用する抑制の理由を 1 つ選択します。

7. [Virtual Deliverability Manager options] (Virtual Deliverability Manager のオプション) セクションでは、アカウントレベルで Virtual Deliverability Manager 設定で定義されている方法を上書きすることで、設定セットでエンゲージメントの追跡と最適な共有配信を使用する方法に関するカスタム設定を定義できます。

   a. この設定セットでエンゲージメントの追跡と最適な共有配信の両方を無効にするには:

i. [Override account level settings] (アカウントレベルの設定の上書き) ボックスをオンにします。

ii. [Engagement tracking] (エンゲージメントの追跡) と [Optimized shared delivery] (共有配信の最適化) の両方で [有効化] がオフになっていることを確認し、[変更の保存] を選択します。

b. この設定セットで、エンゲージメントの追跡と最適な共有配信のいずれかまたは両方を有効または無効にするには

i. [アカウントレベルの設定の上書き] ボックスをオンにします。

ii. [Engagement tracking] (エンゲージメントの追跡) と [Optimized shared delivery] (共有配信の最適化) のいずれかまたは両方で [有効化] をオンまたはオフにし、[変更の保存] を選択します。

c. この設定セットのエンゲージメントの追跡と最適な共有配信について、Virtual Deliverability Manager のアカウントレベルの設定に戻すには:

- [Override account level settings] (アカウントレベルの設定の上書き) ボックスをオフにして、[Save changes] (変更の保存) を選択します。

8. オプションで、1 つ以上のタグをタグセクションに追加することができます。設定セットに追加するタグごとに、上記の手順を繰り返します。

a. 新しいタグを追加を選択します。

b. タグキーを入力します。

c. タグ値(オプション) を入力します。

入力したタグを削除するには、そのタグの削除を選択します。最大 50 個のタグを入力できます。

9. セットを作成するを選択して、設定セットを作成します。

これで設定セットを作成したので、オプションで設定セットのイベント送信先をオプションで定義できます。これにより、イベントの送信先に対して指定したイベントタイプでトリガーされるイベント公開が有効になります。設定セットには、複数のイベントタイプが定義された複数のイベント送信先を含めることができます。Amazon SES でのイベント送信先の作成 を参照してください。

設定セット (AWS CLI) を作成します。

AWS CLIのaws sesv2 create-configuration-setコマンドへの入力として、JSON ファイルを使用して、設定セットを作成できます。

1. CLI 入力 JSON ファイルの作成

   任意のファイル編集ツールを使用して、以下のキーと、ご使用の環境に有効な値を持つ JSON ファイルを作成するか、SES API v2 aws sesv2 create-configuration-set コマンドで --generate-cli-skeleton オプションを使用して (値は指定しません)、サンプルの JSON 構造を標準出力に出力します。

   この例では、create-configuration-set.jsonという名前のファイルを使用します。

```
{
    "ConfigurationSetName": "sample-configuration-set",
    "TrackingOptions": {
        "CustomRedirectDomain": "some.domain.com",
        "HttpsPolicy": "REQUIRE"
    },
    "DeliveryOptions": {
        "TlsPolicy": "REQUIRE",
        "SendingPoolName": "sending pool",
        "MaxDeliverySeconds": 300
    },
    "ReputationOptions": {
        "ReputationMetricsEnabled": true,
        "LastFreshStart": timestamp
    },
    "SendingOptions": {
        "SendingEnabled": true
    },
    "Tags": [
        {
            "Key": "tag key",
            "Value": "tag value"
        }
    ],
    "SuppressionOptions": {
        "SuppressedReasons": ["BOUNCE","COMPLAINT"]
    }
}
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

2. 作成したファイルを入力として使用し、次のコマンドを実行します。

```
aws sesv2 create-configuration-set --cli-input-json file://create-configuration-set.json
```

> **Note**
>
> このコマンドの AWS CLI リファレンスを確認するには、「create-configuration-set」を参照してください。

Amazon SES の設定セットの管理

設定セットを作成した後は、SES コンソール、Amazon SES API v2、および Amazon SES CLI v2 を使用して、Amazon SES 設定セットの表示、更新、および削除などの管理ができます。設定セットを検証済み ID に割り当て、ID から E メールを送信するたびに、デフォルトの設定セットとして割り当てることもできます。

このセクションのトピック:

- 設定セットの表示、編集、削除 (コンソール)
- リスト設定セットの表示 (AWS CLI)
- 構成セットの詳細を取得する (AWS CLI)
- 設定セットの削除 (AWS CLI)
- 設定セットからの E メールの送信を停止する (AWS CLI)
- デフォルトの設定セットを理解する
- Amazon SES でのイベント送信先の作成

- Amazon SES での IP プールの割り当て
- カスタムドメインを設定してオープンとクリックの追跡を処理します

設定セットの表示、編集、削除 (コンソール)

既存の設定セットの詳細ページへアクセスする

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/ses/ で Amazon SES コンソールを開きます。
2. 設定 のナビゲーションペインで、設定セットを選択します。
3. 設定の詳細を表示するには、設定リスト内の名前を選択します。これにより、詳細ページに移動します。

設定セット詳細ページには、設定セットの詳細用の 2 つのタブがあり、各タブにパネルがあり、次のように表示、編集、削除できます。

- [Overview (概要)] タブ
  - 一般的な詳細— このパネルには、設定セットの一般的な詳細が表示されます。
    - 送信ステータス(現在有効になっているかどうかを示します)
    - 設定セット名
    - IP プールの送信
    - Transport Layer Security (TLS)
    - Custom redirect ドメイン
  - レピュテーションのオプション— このパネルには、送信レピュテーションに関する詳細が表示されます。
    - 評価メトリクス (指標をトラッキングしているかどうかを示します)
    - 前回の新規開始 (設定セットの評価メトリクスが最後にリセットされた日時)。
  - サプレッションリストのオプション – このパネルには、アカウントレベルのサプレッションリストを設定セットで上書きしているかどうか、および上書きしている場合はその詳細が表示されます。
    - サプレッションリストの設定 (アカウントレベルの設定が上書きされていることを示します。上書きされていない場合、パネルに表示されるのはこの項目だけです)

    - サプレッションリスト (アカウントレベルの設定をどのように上書きしているか (サプレッションリストの有効/無効) を示します)
    - サプレッションの理由 (バウンスや苦情が、受信者の E メールアドレスをサプレッションリストに追加する理由であるかどうかを示します)
- Virtual Deliverability Manager options (Virtual Deliverability Manager のオプション) – このパネルには、エンゲージメントの追跡と最適な共有配信に関するVirtual Deliverability Manager アカウントの設定を設定セットで上書きするかどうかと、上書きする場合はその詳細が表示されます。
    - Engagement tracking (エンゲージメントの追跡) (エンゲージメントの追跡が有効か無効かを示します)
    - Optimized shared delivery (最適な共有配信) (最適な共有配信が有効か無効かを示します)
- タグ— このパネルには、設定セットに添付したすべてのタグが表示されます。
    - キー
    - 値

これらのパネルでは、次のアクションを実行できます。

- 編集ボタン、またはタグパネルの場合はタグの管理ボタンをクリックして、各パネルの各詳細を編集します。
- フィールドの詳細については、設定セットを作成する (コンソール)ステップの関連セクションを参照してください。

> **Tip**
>
> 編集が完了したら、変更の保存をしてください。保存せずに設定セットの詳細ページに戻るには、キャンセルを選択します。

- イベントの発行先タブ
    - すべての発行先 (**#########**)— このパネルには、設定セットに対して入力したすべてのイベント送信先が一覧表示されます。各送信先には、次のように表示されます。
        - 名前
        - 送信先
        - イベントタイプ
        - イベントの発行

このパネルでは、次のアクションを実行できます。

- [Add destination] (送信先の追加) ボタンを選択して、イベントの送信先を新しく追加します。イベント送信先の追加の詳細については、イベント送信先の作成 を参照してください。
- 既存のイベント送信先の名前を選択すると、編集画面が表示され、イベントを変更できます。
- 既存のイベント送信先を削除するには、イベント名の横にあるチェックボックスをオンにして、[Delete] (削除) ボタンを選択します。

各設定セットの詳細ページの上部および概要またはイベントの送信先タブのどちらかには、次のオプションがあります。

- 削除— このボタンは設定セットを削除します。
- 送信の無効化— このボタンは、設定セットからの E メールの送信を停止します。

リスト設定セットの表示 (AWS CLI)

の list-configuration-sets コマンドを使用して、現在のリージョンのアカウントに関連付けられているすべての設定セットのリストを次のように AWS CLI 生成できます。

```
aws sesv2 list-configuration-sets
```

構成セットの詳細を取得する (AWS CLI)

で get-configuration-set コマンドを使用して、次のように特定の設定セットの詳細 AWS CLI を取得できます。

```
aws sesv2 get-configuration-set --configuration-set-name name
```

設定セットの削除 (AWS CLI)

で delete-configuration-set コマンドを使用して、次のように特定の設定セット AWS CLI を削除できます。

```
aws sesv2 delete-configuration-set --configuration-set-name name
```

設定セットからの E メールの送信を停止する (AWS CLI)

で put-configuration-set-sending-options コマンドを使用して、次のように特定の設定セットからの E メールの送信 AWS CLI を停止できます。

```
aws sesv2 put-configuration-set-sending-options --configuration-set-name name --no-sending-enabled
```

もう一度送信を開始するには、次のように、代わりにオプションを指定して同じ`--sending-enabled`コマンドを実行します。

```
aws sesv2 put-configuration-set-sending-options --configuration-set-name name --sending-enabled
```

デフォルトの設定セットを理解する

このセクションでは、検証済み ID でデフォルトとして使用される設定セットを割り当てる概念について説明します。利点とユースケースを理解するのに役立ちます。

デフォルト設定セットは、その設定セットに関連付けられた電子メール ID から送信するすべてのメッセージに、その規則を自動的に適用します。ID 作成時、または既存の ID の編集機能実行時に、E メールアドレスとドメイン ID の両方に対してデフォルトの設定セットを適用できます。

デフォルト設定セットに関する考慮事項

- ID に関連付ける前に、まず設定セットを作成する必要があります。
- デフォルト設定セットは、ID が検証された場合にのみ適用されます。
- E メール ID は、一度に 1 つの設定セットにのみ関連付けることができます。ただし、同じ設定セットを複数の ID に適用することができます。
- 電子メールアドレスレベルのデフォルトの設定セットは、ドメインレベルのデフォルトの設定セットよりも優先されます。たとえば、joe@example.com に関連するデフォルト設定セットが、example.com のドメインの設定セットをオーバーライドします。
- ドメインレベルで設定されたデフォルト設定は、そのドメインのすべての電子メールアドレスに適用されます (ドメインの特定アドレスを確認しない限り)。
- ID のデフォルト設定セットとして指定されている設定セットを削除し、その ID を使用して E メールを送信しようとすると、Amazon SES への呼び出しが「bad request」エラーで失敗します。

- デフォルトの設定セットは、代理送信者が使用している検証済み ID に割り当てることができません。
- 既存の設定セットを ID で使用されるデフォルトとして指定する方法は、実際には検証済み ID の関数であるため、ID ワークフローで手順が示されます。
  - ID の作成時にデフォルトの設定セットを指定する — ドメイン ID のデフォルト設定セットまたは E メール ID のデフォルト設定セットについては、Amazon での ID の作成と検証 SES 章にある、オプションのステップ 6 の手順に従います。
  - 既存の ID にデフォルトの設定セットを指定する — ステップ 5 の詳細に加えて、コンソールを使用して ID を編集する の手順に従ってください。

    a. [Configuration set] (設定セット) タブを選択します。

    b. デフォルト設定セットコンテナにある[Edit] (編集) を選択してください。

    c. リストボックスを選択し、デフォルトとして使用される既存の設定セットを選択します。

    d. コンソールを使用して ID を編集する の残りのステップを続行します。

> **Note**
>
> デフォルトとして割り当てた設定セットで評価メトリクスが有効になっている場合、デフォルト設定セットを使用して送信されたメールについて追加料金が発生します。「CloudWatch のメトリクスあたりの料金」を参照してください。

Amazon SES でのイベント送信先の作成

イベント送信先を使用すると、次の送信 E メール追跡アクションをモニタリングのために他の AWS サービスに発行できます。

- 送信数
- レンダリングの失敗
- 拒否
- 配信数
- ハードバウンス
- 苦情
- 配送の遅延
- サブスクリプション

- 開封数
- クリック数

イベント発行の詳しい設定方法については、the section called “イベント発行を使用して E メール送信をモニタリングする” を参照してください。

イベント送信先の作成

設定セットを作成したので、設定セットのイベント送信先を作成できます。これにより、イベントの送信先に対して指定したイベントタイプでトリガーされるイベント公開が有効になります。設定セットには、複数のイベントタイプが定義された複数のイベント送信先を含めることができます。

設定を作成していない場合は、「the section called “設定セットを作成します。”」を参照してください。

以下のステップでは、設定セットにイベント送信先を作成または追加する方法を示します。

SES コンソールを使用してイベント送信先を作成または追加するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/ses/ で Amazon SES コンソールを開きます。
2. 設定のナビゲーションペインで、設定セットを選択します。
3. [Name] (名前) 列から設定セットの名前を選択して、詳細にアクセスします。
4. [Event destinations] (イベントの送信先) タブを選択します。
5. 送信先の追加を選択します。
6. イベントタイプを選択します。

   E メール送信イベントは、Amazon SES で測定できる送信アクティビティに関連するメトリクスです。このステップでは、Amazon SES がイベントの送信先に発行する E メール送信イベントの種類を選択します。

   イベントタイプの詳細については、「Amazon SES 送信アクティビティのモニタリング」を参照してください。

   a. 発行するにはイベントタイプを選択します

      - 送信と配送 — 発行するイベントタイプを選択するには、それぞれのチェックボックスを選択するか、すべて選択を選択してすべてのイベントタイプを発行します。

イベントタイプ

- 送信数— 送信リクエストが成功すると、Amazon SES はそのメッセージを受信者のメールサーバーに配信しようと試行します。
- レンダリング失敗 – テンプレレンダリングの問題により E メールが送信されませんでした。このイベントタイプは、テンプレートデータが見つからない場合や、テンプレートのパラメータとデータが一致しない場合に発生します。(このイベントタイプは、SendTemplatedEmail または SendBulkTemplatedEmail API オペレーションを使用して E メールを送信する場合にのみ発生します。)
- 拒否 - Amazon SES は E メールを受け取りましたが、この E メールにウイルスが含まれていると判断して拒否したため、受信者のメールサーバーに E メール配信を試みませんでした。
- 配信 – Amazon SES が受信者のメールサーバーに E メールを正常に送信しました。
- ハードバウンス – 受信者のメールサーバーにより、メールが完全に拒否されました。(ソフトバウンスは、Amazon SES が一定期間にわたって再試行してもメールを配信できなかった場合に限ります。)
- 苦情数 — E メールは受信者のメールサーバーに正常に配信されましたが、受信者はスパムとしてマークしました。
- 配信の遅延 – 一時的な問題が発生したため、メールを受信者のメールサーバーに配信できませんでした。配信の遅延は、受信者の受信トレイがいっぱいになった場合や、受信側の電子メールサーバーで一時的な問題が発生した場合などに発生します。(このイベントタイプは Amazon Pinpoint ではサポートされていません。)
- サブスクリプション— E メールは正常に配信されましたが、受信者はE メールのヘッダーにあるList-Unsubscribeまたはフッターの Unsubscribeリンクをクリックし、サブスクリプション設定を更新しました。(このイベントタイプは Amazon Pinpoint ではサポートされていません。)

- オープンおよび追跡クリック — 購読者のエンゲージメントを測定するには、オープンおよびクリックを追跡するチェックボックスのいずれかまたは両方を選択します。
  - オープン– 受信者がメッセージを受け取り、E メールクライアントで開きました。
  - クリック – 受信者は E メール内の 1 つ以上のリンクをクリックしました。

> **Note**
>
> ここで、または他の構成設定で定義されているオープンおよびクリックイベントの公開は、Virtual Deliverability Manager ダッシュボードのエンゲージメント

追跡オプションには影響しません。これらは、Virtual Deliverability Manager のアカウント設定または構成セットのオーバーライドのいずれかによって定義されます。例えば、Virtual Deliverability Manager を使用してエンゲージメントの追跡を無効にしても、ここで設定したオープンおよびクリックイベントの公開は、SES のイベント送信先で無効になりません。

- 設定セットのリダイレクトドメイン — このフィールドは設定セットの作成時に表示されます。また、設定セットの作成時にカスタムのリダイレクトドメインを割り当てた場合には、そのリダイレクトドメイン名があらかじめ入力されています。

> **Note**
>
> そのドメインでのオープンおよびクリック追跡の設定セット内の [Custom redirect domain] を更新できます。設定セットを作成します。 のステップ 4 の「追跡オプション」を参照してください。オープンとクリックのカスタムドメインの設定の詳細については、「カスタムドメインを設定してオープンとクリックの追跡を処理します」を参照してください。

b. [次へ] を選択して続行します。

7. 送信先の指定

イベント送信先は、E メール送信イベントを発行できる AWS サービスです。適切な送信先を選択する方法は、キャプチャする詳細のレベルとデータを受信する方法によって異なります。

a. 送信先オプション

- 送信先タイプ – イベントを発行する AWS サービスの横にあるラジオボタンを選択すると、詳細パネルにサービスに対応するフィールドが表示されます。以下のリンクを選択すると、サービスの詳細パネルについての説明が表示されます。
  - Amazon CloudWatch (追加料金が適用されます。「CloudWatch のメトリクスあたりの料金」を参照してください)。
  - Amazon Data Firehose
  - Amazon EventBridge
  - Amazon Pinpoint (をサポートしません。配送の遅延またはサブスクリプションイベントのタイプ。)
  - Amazon SNS

イベント発行モデルを使用して電子メール操作を監視する方法の詳細については、Amazon SES イベント発行を使用して E メール送信をモニタリングするを参照してください。

- 名前— この設定セットに送信先の名前を入力します。名前には、英字、数字、ダッシュ、ハイフンを含むことができます。
- イベントの発行— この送信先のイベント発行　をオンにするには、[Enabled] チェックボックスを選択します。

b. 次へを選択して続行します。

8. 確認

入力が正しいことを確認したら、送信先を追加するを選択して、イベント送信先を追加します。

Amazon SES コンソール、Amazon SES API v2、または Amazon SES CLI v2 を使用して、イベント送信先を作成することもできます。

SES API を使用してイベント送信先を作成するには

- SES API を使用してイベント送信先を作成する方法については、「CreateConfigurationSetEventDestination」を参照してください。

イベント送信先の編集、無効化/有効化、または削除

SES コンソールを使用してイベント送信先を編集、無効/有効化、または削除するには、以下のステップに従います。

SES コンソールを使用してイベント送信先を編集、無効/有効化、または削除するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/ses/ で Amazon SES コンソールを開きます。
2. 設定のナビゲーションペインで、設定セットを選択します。
3. [Name] (名前) 列から設定セットの名前を選択して、詳細にアクセスします。
4. 設定セットの [Event destinations] (イベントの送信先) タブを選択します。
5. [Name] (名前) 列でイベント送信先の名前を選択します。
6. • 編集するには – 編集するフィールドセットの各パネルで [Edit] (編集) ボタンを選択し、変更を加えたら、[Save changes] (変更の保存) を選択します。

- 無効または有効にするには – 右上隅にある [Disable] (無効) または [Enable] (有効) ボタンのいずれかを選択します。
- 削除するには – 右上隅にある [Delete] (削除) ボタンを選択します。

Amazon SES コンソール、Amazon SES API v2、または Amazon SES CLI v2 を使用して、Amazon SES イベント送信先を編集、無効化/有効化、または削除することもできます。

SES API を使用してイベント送信先を編集、無効/有効化、または削除するには

1. SES API を使用したイベント送信先の無効化/有効化については、「UpdateConfigurationSetEventDestination」を参照してください。
2. SES API を使用してイベント送信先を削除する方法については、「DeleteConfigurationSetEventDestination」を参照してください。

Amazon SES での IP プールの割り当て

IP プールを使用すると、特定タイプの E メールを送信するための専用 IP アドレスのグループを作成できます。また、すべての Amazon SES の顧客が共有する IP アドレスのプールを使用できます。

設定セットに IP プールを割り当てる場合は、次のオプションから選択できます。

- 特定の専用 IP プール – 既存の専用 IP プールを選択すると、そのプールに属する専用 IP アドレスのみを使用して、設定セットを使用する E メールが送信されます。作成方法については、以下の手順を参照してください。
  - 新しい標準 IP プールについては、「専用 IP (標準) 用の専用 IP プールの作成」を参照してください。
  - 新しいマネージド IP プールについては、「専用 IP (マネージド) を有効にするためのマネージド IP プールを作成する」を参照してください。
- ses-default-dedicated-pool – このプールには、まだ IP プールに属していない、アカウントの専用 IP アドレスがすべて含まれています。プールに関連付けられていない設定セットを使用して E メールを送信する場合、または設定セットをまったく指定せずに E メールを送信する場合、E メールは、このデフォルトプール内にあるいずれかのアドレスから送信されます。このプールは SES によって自動的に管理され、編集することはできません。
- ses-shared-pool – このプールには、すべての Amazon SES ユーザー間で共有されている多数の IP アドレスが含まれています。このオプションは、通常の送信動作と異なる E メールを送信する必要がある場合に便利です。

IP プールを設定セットに割り当てる

このセクションでは、Amazon SES コンソールを使用した設定セットの IP プールの割り当てと変更の手順について説明します。

- コンソールを使用して、IP プールを設定セットに割り当てるには
  - 新しい設定セットを作成する – 設定セットを作成します。のステップ 4 のIP プールの送信を参照してください。
  - 既存の設定セットを変更する — 選択した設定セットの [General details] (一般的な詳細) パネル内の [Edit] (編集) ボタンをクリックし、設定セットを作成します。 のステップ 4 のIP プールの送信の指示に従います。

カスタムドメインを設定してオープンとクリックの追跡を処理します

イベント発行を使用して、オープンおよびクリックイベントをキャプチャすると、Amazon SES は送信する E メールに小さな変更を加えます。オープンイベントをキャプチャするために、SES は SES を通じて送信される各 E メールに、1 x 1 ピクセルの透明な GIF イメージを追加します。このイメージには各 E メールに一意のファイル名が含まれており、SES が運用するサーバー上でホストされます。このイメージがダウンロードされると、SES はどのメッセージが誰によって開かれたのかを正確に判断できます。

デフォルトでは、このピクセルは E メールの下部に挿入されます。ただし、E メールプロバイダーのアプリケーションによっては、特定のサイズを超えると E メールのプレビューが切り捨てられ、メッセージの残りの部分を確認するためのリンクが表示される場合があります。このシナリオでは、SES ピクセルの追跡イメージはロードされず、追跡しようとしているオープン率は破棄されます。これを回避するには、必要に応じて E メールの本文に `{{ses:openTracker}}` プレースホルダーを挿入して、E メールの先頭または他のいずれかの場所にピクセルを配置します。プレースホルダーを含むメッセージを SES が受信すると、そのメッセージはオープンの追跡ピクセルイメージに置き換えられます。

> **Important**
>
> - 複数ある `{{ses:openTracker}}` プレースホルダーは、送信時に SES が削除します。
> - E メールテンプレートには、使用する `{{ses:openTracker}}` プレースホルダーを 1 つのみ追加してください。複数のプレースホルダーを追加すると、`400 BadRequestException` エラーコードが返されます。

リンククリックイベントをキャプチャするために、SES は E メールのリンクを SES が運用するサーバーへのリンクに置き換えます。これにより、受信者はただちに目的の宛先にリダイレクトされます。このサーバーに送信されるリクエストのヘッダーの合計サイズは、Cookie を含め、8,192 バイトを超えてはなりません。超える場合は、`400 BadRequestException` エラーコードが返されます。

SES が所有および運用するドメインではなく、独自のドメインを使用して、受信者向けにより一貫性のあるエクスペリエンスを作成することができます。この場合、すべての SES インジケーターが削除されます。複数のカスタムドメインを設定して、オープンとクリックの追跡イベントを処理できます。これらのカスタムドメインは設定セットと関連付けられています。設定セットを使用して E メールを送信すると、その設定セットがカスタムドメインを使用するように設定されている場合、その E メール内のオープンとクリックのリンクは、その設定セットで指定されたカスタムドメインを自動的に使用します。

このセクションでは、所有するサーバー上にサブドメインを設定して、ユーザーを SES が運用する開封とクリックの追跡サーバーに自動的にリダイレクトする手順について説明します。これらのドメインの設定には 3 つのステップがあります。まず、サブドメイン自体を設定し、次にカスタムドメインを使用するよう設定セットを設定します。次に、カスタムドメインを使用するよう設定セットを設定し、オープンイベントおよびクリックイベントを発行するようにイベント送信先を設定します。このトピックには、これらのステップをすべて完了する手順が含まれています。

ただし、カスタムドメインを設定せずにオープンまたはクリック追跡を有効にするだけの場合は、オープンイベントやクリックイベントなど、指定したイベントタイプでトリガーされるイベント公開を有効にする設定セットのイベント送信先の定義に直接進むことができます。設定セットには、複数のイベントタイプが定義された複数のイベント送信先を含めることができます。Amazon SES でのイベント送信先の作成を参照してください。

パート 1: オープンとクリックのリンクのリダイレクトを処理するためのドメインの設定

リダイレクトドメインを設定する固有の手順は、ウェブホスティングプロバイダ (および HTTPS サーバーを使用する場合は Content Delivery Network) によって異なります。次のセクションの手順では、特定のステップではなく一般的なガイダンスを示します。

オプション 1: HTTP ドメインの設定

HTTP ドメインを使用してオープンとリンクのリンクを処理する予定の場合 (HTTPS ドメインではなく)、サブドメインを設定するプロセスには、数ステップのみが含まれます。

> **Note**
>
> HTTP プロトコルを使用するカスタムドメインを設定し、HTTPS プロトコルを使用するリンクを含む E メールを送信する場合、E メールのリンクをクリックすると警告メッセージが表示されることがあります。HTTPS プロトコルを使用するリンクが含まれる E メールを送信する場合は、HTTPS ドメインを使用して、クリックの追跡イベントを処理する必要があります。

オープンとクリックのリンクの処理のために、HTTP のサブドメインを設定するには

1. オープンとクリックの追跡リンクのために使用するサブドメインを作成します。SES では、このサブドメインはこれらのリンクの処理専用であり、追跡する E AWS リージョン メールを送信する ごとにサブドメインを作成することを推奨しています。
2. SES で使用するサブドメインを検証します。詳細については、「ドメイン ID の作成」を参照してください。
3. リクエストを SES 追跡ドメインにリダイレクトするサブドメインの DNS 設定に新しい CNAME レコードを追加します。リダイレクト先のアドレスは、カスタムサブドメイン AWS リージョン と同じ の である必要があります。

   SES が利用可能な AWS リージョン の追跡ドメインは、次の表のとおりです。カスタムドメインと同じリージョンにあるドメインを選択してください。

AWS リージョン	AWS 追跡ドメイン
米国東部(オハイオ)	`r.us-east-2.awstrack.me`
米国東部 (バージニア北部)	`r.us-east-1.awstrack.me`
米国西部 (北カリフォルニア)	`r.us-west-1.awstrack.me`
米国西部 (オレゴン)	`r.us-west-2.awstrack.me`
アフリカ (ケープタウン)	`r.af-south-1.awstrack.me`
アジアパシフィック (ジャカルタ)	`r.ap-southeast-3.awstrack.me`
アジアパシフィック (ムンバイ)	`r.ap-south-1.awstrack.me`

AWS リージョン	AWS 追跡ドメイン
アジアパシフィック (大阪)	r.ap-northeast-3.awstrack.me
アジアパシフィック (ソウル)	r.ap-northeast-2.awstrack.me
アジアパシフィック (シンガポール)	r.ap-southeast-1.awstrack.me
アジアパシフィック (シドニー)	r.ap-southeast-2.awstrack.me
アジアパシフィック (ジャカルタ)	r.ap-southeast-3.awstrack.me
アジアパシフィック (ジャカルタ)	r.ap-southeast-3.awstrack.me
アジアパシフィック (東京)	r.ap-northeast-1.awstrack.me
カナダ (中部)	r.ca-central-1.awstrack.me
欧州 (フランクフルト)	r.eu-central-1.awstrack.me
欧州 (アイルランド)	r.eu-west-1.awstrack.me
欧州 (ロンドン)	r.eu-west-2.awstrack.me
欧州 (ミラノ)	r.eu-south-1.awstrack.me
欧州 (ストックホルム)	r.eu-north-1.awstrack.me
イスラエル (テルアビブ)	r.il-central-1.awstrack.me
中東 (バーレーン)	r.me-south-1.awstrack.me
南米 (サンパウロ)	r.sa-east-1.awstrack.me
AWS GovCloud (米国西部)	r.us-gov-west-1.awstrack.me
AWS GovCloud (米国東部)	r.us-gov-east-1.awstrack.me

> **Note**
>
> ウェブホスティングプロバイダによっては、サブドメインの DNS レコードの変更が有効になるまでに数分かかることがあります。ウェブホスティングプロバイダや IT 組織はこれらの遅延に関する追加情報を提供できます。

オプション 2: HTTPS ドメインの設定

HTTPS ドメインを使用して、開封とリンクのクリックを追跡することもできます。開封とリンクのクリックを追跡するために HTTPS ドメインを設定するには、「HTTP ドメインの設定」に必要なステップ以外に、いくつかの追加ステップを実行する必要があります。

オープンとクリックのリンクの処理のために、HTTPS のサブドメインを設定するには

1. オープンとクリックの追跡リンクのために使用するサブドメインを作成します。SES では、このサブドメインはこれらのリンクの処理専用であり、追跡する E AWS リージョン メールを送信する ごとにサブドメインを作成することを推奨しています。
2. SES で使用するサブドメインを検証します。詳細については、「ドメイン ID の作成」を参照してください。
3. Amazon CloudFront などのコンテンツ配信ネットワークを使用して新しいアカウントを作成します。「基本的な CloudFront ディストリビューションの開始方法」を参照してください。
4. 例えば、`r.us-east-1.awstrack.me` などの SES 追跡ドメインであるオリジンに CDN を設定します。CDN は、カスタムドメインと同じリージョンにある AWS 追跡ドメインを指す必要があります。CDN は、リクエスタから提供される `Host` ヘッダーをオリジンに渡す必要があります。詳細については、この AWS re:Post 記事を参照してください。

   SES が利用可能な AWS リージョン の追跡ドメインは、次の表のとおりです。カスタムドメインと同じリージョンにあるドメインを選択してください。

AWS リージョン	AWS 追跡ドメイン
米国東部(オハイオ)	`r.us-east-2.awstrack.me`
米国東部 (バージニア北部)	`r.us-east-1.awstrack.me`

AWS リージョン	AWS 追跡ドメイン
米国西部 (北カリフォルニア)	r.us-west-1.awstrack.me
米国西部 (オレゴン)	r.us-west-2.awstrack.me
アフリカ (ケープタウン)	r.af-south-1.awstrack.me
アジアパシフィック (ジャカルタ)	r.ap-southeast-3.awstrack.me
アジアパシフィック (ムンバイ)	r.ap-south-1.awstrack.me
アジアパシフィック (大阪)	r.ap-northeast-3.awstrack.me
アジアパシフィック (ソウル)	r.ap-northeast-2.awstrack.me
アジアパシフィック (シンガポール)	r.ap-southeast-1.awstrack.me
アジアパシフィック (シドニー)	r.ap-southeast-2.awstrack.me
アジアパシフィック (東京)	r.ap-northeast-1.awstrack.me
カナダ (中部)	r.ca-central-1.awstrack.me
欧州 (フランクフルト)	r.eu-central-1.awstrack.me
欧州 (アイルランド)	r.eu-west-1.awstrack.me
欧州 (ロンドン)	r.eu-west-2.awstrack.me
欧州 (ミラノ)	r.eu-south-1.awstrack.me
欧州 (ストックホルム)	r.eu-north-1.awstrack.me
イスラエル (テルアビブ)	r.il-central-1.awstrack.me
中東 (バーレーン)	r.me-south-1.awstrack.me
南米 (サンパウロ)	r.sa-east-1.awstrack.me
AWS GovCloud (米国西部)	r.us-gov-west-1.awstrack.me

AWS リージョン	AWS 追跡ドメイン
AWS GovCloud (米国東部)	`r.us-gov-east-1.awstrack.me`

5. Route 53 を使用して、ドメインの DNS 設定を管理し、CDN として CloudFront を管理する場合、Route 53 で CloudFront ディストリビューションを参照するエイリアス記録を作成します (d111111abcdef8.cloudfront.netなど)。詳細については、Amazon Route 53 デベロッパーガイドの Amazon Route 53 コンソールを使用したレコードの作成を参照してください。

   それ以外の場合、サブドメインの DNS 設定で、CDN のアドレスを参照する CNAME レコードを追加します。

6. 信頼できる認証機関から SSL 証明書を取得します。証明書は、ステップ 1 で作成したサブドメインと、ステップ 3 ~ 5 で設定した CDN の両方をカバーする必要があります。証明書を CDN にアップロードします。

7. 次の curl コマンドを使用すると、新しく作成したカスタムドメインが適切なリージョンと HTTPS プロトコルを使用しているかを検証できます。次の例では、ドメイン名前を除き、すべてがリテラルです。

   ```
   curl --head https://custom.domain.com/favicon.ico
   ```

   以下の例のとおりの応答が返されます。

   ```
   (python-sdk-test) jdoe@12a34567b89c BaconRedirectService % curl --head https://
   custom.domain.com/favicon.ico
   HTTPS/1.1 200 OK
   x-amz-ses-region: us-east-1
   x-amz-ses-request-protocol: https
   Content-Type: image/x-icon
   Transfer-Encoding: chunked
   Date: Fri, 30 Aug 2024 13:50:14 GMT
   ```

   この応答は、次のプロパティで構成されます。

   - `x-amz-ses-region` ヘッダー値は、リクエストを受信した SES リージョンです。
   - `x-amz-ses-request-protocol` ヘッダー値は、ヘッダーの CDN と SES 間のリクエストに使用されるプロトコルです。

設定が適正である場合、リージョンはドメインを作成したリージョンを反映し、プロトコルは HTTPS であるはずです。

パート 2: 設定セットを使用したカスタムリダイレクトドメインと HTTPS ポリシーの指定

開封とクリックの追跡リダイレクトを処理するようにドメインを設定した後、設定セットでカスタムドメインと HTTPS ポリシーを指定する必要があります。

設定セットを使用して E メールを送信する場合、設定セットがカスタムドメインを使用するように設定されていると、E メールの開封とクリックのリンクは、その設定セットで指定されたカスタムドメインを自動的に使用します。

これは SES コンソール、または `CreateConfigurationSet` v2 API オペレーションを使用して実行できます。

コンソールを使用してカスタムリダイレクトドメインと HTTPS ポリシーを指定するには

- 設定セットを作成または編集する際に、「設定セットを作成します。」のステップ 4 の「追跡オプション」を使用して、カスタムリダイレクトドメインと HTTPS ポリシーのオプションを指定します。

を使用してカスタムリダイレクトドメインと HTTPS ポリシーを指定するには AWS CLI

SES API v2 の `CreateConfigurationSet` オペレーションを使用し、`TrackingOptions` プロパティを利用してカスタムリダイレクトドメインと HTTPS ポリシーを指定できます。次の例 AWS CLI に示すように、 からこのオペレーションを呼び出すことができます。

- E メールを送信および追跡する AWS リージョン で設定セットを作成します。

```
aws sesv2 create-configuration-set --cli-input-json file://create.json
```

- この例では、入力ファイルは `TrackingOptions` プロパティのパラメータを使用しています。`CustomRedirectDomain` は、開封とクリックの追跡リンクに使用するカスタムドメインを指定し、`HttpsPolicy` は HTTPS ポリシーオプションを指定します。

```
{
```

```
    "ConfigurationSetName": "my-config-set",
    "TrackingOptions": {
        "CustomRedirectDomain": "marketing.example.com",
        "HttpsPolicy": "REQUIRE"
    },
    "SendingOptions": {
        "SendingEnabled": true
    }
}
```

HttpsPolicy パラメータでは、次の値を指定して、カスタムリダイレクトドメインの開封とクリックの追跡リンクのプロトコルを設定できます。

- OPTIONAL – (デフォルト動作) 開封の追跡リンクは HTTP を使用してラップされます。クリックの追跡リンクは、リンクの元のプロトコルを使用してラップされます。
- REQUIRE – 開封とクリックの追跡リンクはどちらも HTTPS を使用してラップされます。
- REQUIRE_OPEN_ONLY – 開封の追跡リンクは HTTPS を使用してラップされます。クリックの追跡リンクは、リンクの元のプロトコルを使用してラップされます。

パート 3: 設定セットを使用した開封イベントタイプとクリックイベントタイプの指定

前のステップの設定セットでのカスタムドメインの指定後、設定セットを介してイベント送信先で開封イベントタイプおよび/またはクリックイベントタイプを指定する必要があります。

これは SES コンソール、または CreateConfigurationSetEventDestination v2 API オペレーションを使用して実行できます。

コンソールを使用して開封イベントタイプおよび/またはクリックイベントタイプを選択するには

- イベント送信先を作成または変更する際に、「the section called “イベント送信先の作成”」のステップ 6 の「開封とクリックの追跡」を使用して、イベントタイプを指定します。

メールの送信時に設定セットを指定する

E メールの送信時に設定セットを使用するには、E メールのヘッダー内で設定セットの名前を渡す必要があります。送信する E メールのヘッダー内で設定セットを渡すことは、すべての Amazon SES E メール送信手段 (AWS CLI、AWS SDK、Amazon SES SMTP インターフェイスなど) で許可されています。

SMTP インターフェイスまたはSendRawEmail API オペレーションを使用している場合、E メールに以下のヘッダーを含めることで設定セットを指定できます (*ConfigSet*は使用する設定セットの名前に置き換えます)。

```
X-SES-CONFIGURATION-SET: ConfigSet
```

このガイドでは、 AWS SDK および Amazon SES SMTP のインターフェイスを使用して E メールを送信するためのコード例を紹介します。それぞれの例には、設定セットの指定方法が含まれています。設定セットへの参照を含む E メールを送信するための手順を確認するには、以下を参照してください。

- AWS SDK を使用して Amazon SES 経由で E メールを送信する
- Amazon SES SMTP インターフェイスを使用して E メールを送信

レピュテーションメトリクスの表示とエクスポート

Amazon SES は、アカウント全体のバウンス率や苦情率についての情報を Amazon CloudWatch に自動的にエクスポートします。これらのメトリクスを使用して、CloudWatch でアラームを作成したり、Lambda 関数を使用して E メール送信を自動的に一時停止したりすることができます。

個々の設定セットの評価メトリクスを CloudWatch にエクスポートすることもできます。設定セットレベルで評価データをエクスポートすると、送信者の評価をより詳細に制御できます。

このセクションでは、Amazon SES API を使用して、個々の設定セットの評価データを CloudWatch にエクスポートする手順について説明します。

レピュテーションメトリクスのエクスポートの有効化

設定セット評価メトリクスのエクスポートを開始するには、UpdateConfigurationSetReputationMetricsEnabled API オペレーションを使用します。Amazon SES API にアクセスするには、 AWS CLI またはいずれかの AWS SDKs を使用することをお勧めします。

この手順では、 AWS CLI がコンピュータにインストールされ、適切に設定されていることを前提としています。のインストールと設定の詳細については AWS CLI、「 AWS Command Line Interface ユーザーガイド」を参照してください。

設定セットの評価メトリクスのエクスポートを有効化するには

- コマンドラインから、以下のコマンドを入力します。

```
aws ses update-configuration-set-reputation-metrics-enabled --configuration-set-
name ConfigSet --enabled
```

前のコマンドの *ConfigSet* を、評価メトリクスのエクスポートを開始する設定セットの名前に置き換えます。

レピュテーションメトリクスのエクスポートの無効化

また、UpdateConfigurationSetReputationMetricsEnabled API オペレーションを使用して、設定セットの評価メトリクスのエクスポートを無効にすることもできます。

設定セットの評価メトリクスのエクスポートを無効化するには

- コマンドラインから、以下のコマンドを入力します。

```
aws ses update-configuration-set-reputation-metrics-enabled --configuration-set-
name ConfigSet --no-enabled
```

前のコマンドの*ConfigSet*を、評価メトリクスのエクスポートを無効にする設定セットの名前に置き換えます。

Amazon SES でグローバルエンドポイントを使用する

Amazon SES グローバルエンドポイントは、E メール送信オペレーションの継続性と信頼性を強化する機能です。この章では、 グローバルエンドポイントの概念、セットアップ、使用について説明します。これにより、マルチリージョン送信 (MRS) を活用して、E メールワークロードの可用性を高め、ディザスタリカバリ機能を向上させることができます。

グローバルエンドポイントとは

グローバルエンドポイントは、SES アウトバウンドワークロードを 2 つに分散できるリソースです AWS リージョン。設定すると、SES は選択したプライマリリージョンとセカンダリリージョン間で送信トラフィックを自動的に分割します。いずれかのリージョンで障害が発生した場合、SES は自動的にトラフィックを影響を受けたリージョンから遠ざけて、送信オペレーションの継続性を維持します。

グローバルエンドポイントを使用する主な利点は次のとおりです。

- E メール送信の継続性の向上
- リージョン間の自動フェイルオーバー
- マルチリージョン設定の簡素化

グローバルエンドポイントの仕組み

グローバルエンドポイントを設定するときは、プライマリリージョン (エンドポイントが作成されるリージョン) とセカンダリリージョンを選択します。次に、SES は、E メール送信リクエストのエントリポイントとして機能するマルチリージョンエンドポイント (MREP) を作成します。

グローバルエンドポイントのセットアッププロセスでは、プライマリリージョンからセカンダリリージョンへのキーアーティファクトと送信制限が同期されます。これにより、両方のリージョンで同等の検証済み ID、設定セット、承認済みの送信制限が、予想されるすべてのボリュームに対して十分であることが保証されます。

グローバルエンドポイントの準備が完了し、SendEmail API コールでエンドポイント ID が指定されると、SES はアウトバウンドトラフィックをプライマリリージョンとセカンダリリージョン間で均等に自動的にルーティングします。いずれかのリージョンに障害が発生した場合、障害が解決されるまで、トラフィックはそのリージョンから別のリージョンに向かって重み付けされます。

グローバルエンドポイントのセットアップ

トピック

前提条件

グローバルエンドポイントを作成する前に、まずアカウントにサービスにリンクされたロール (SLRs) を作成するアクセス許可を SES に付与する必要があります。これらのロールにより、グローバルエンドポイントの作成、使用、モニタリングに必要な重要なサービス機能とリソースアクセスが可能になります。これを行うには、次のポリシーを実装します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "ses.amazonaws.com"
        }
      }
    }
  ]
}
```

グローバルエンドポイントの作成

新しいグローバルエンドポイントを作成するには：

1. https://console.aws.amazon.com/ses/ で SES コンソールを開きます。
2. ナビゲーションペインで、グローバルエンドポイントを選択します。
3. グローバルエンドポイントの作成を選択し、名前フィールドに名前を入力します。

4. ドロップダウンメニューからセカンダリリージョンを選択します。(プライマリリージョンは、デフォルトでコンソールにサインインしたリージョンになります)。
5. (オプション) グローバルエンドポイントに 1 つ以上のタグを追加します。
6. 設定を確認し、グローバルエンドポイントの作成を選択します。

作成プロセスには数秒かかる場合があります。完了すると、グローバルエンドポイントのステータスが「準備完了」に変わります。

の使用 AWS CLI :

```
aws sesv2 create-multi-region-endpoint --primary-region us-west-2 --secondary-
region us-east-1 --endpoint-name MyGlobalEndpoint
```

前の例では、以下のようになっています。

- *us-west-2* をグローバルエンドポイントのプライマリリージョンに置き換えます。
- *us-east-1* をグローバルエンドポイントのセカンダリリージョンに置き換えます。
- *MyGlobalEndpoint* をわかりやすい名前に置き換えて、グローバルエンドポイントを指定します。

グローバルエンドポイントの状態

グローバルエンドポイントには、次の状態があります。

- 作成中 – リソースはプロビジョニング中です
- 準備完了 – リソースは使用できる状態です
- 失敗 – リソースのプロビジョニングに失敗しました
- 削除中 – リソースはリクエストに応じて削除されています

セカンダリリージョンの準備

グローバルエンドポイントを作成したので、グローバルエンドポイントを使用して E メールを送信する前に、すべてのコンポーネント (ID、設定セット、E メールテンプレート、送信制限) を含む E メール送信設定がプライマリリージョンとセカンダリリージョンで一貫していることを確認する必要

があります。この調整は、潜在的な問題を回避し、適切な E メールの配信と追跡を確保するために不可欠です。

コンソールのリージョン重複機能は、リソースを自動的に複製し、アカウントレベルの設定をプライマリリージョンからセカンダリリージョンに複製することで、両方のリージョンで同等の設定をすばやく行うのに役立ちます。

リソースの依存関係に基づいて、リソースを複製する順序が重要になります。競合を回避するには、次のトピックの順序に従います。

トピック

- 設定セットの複製
- 検証済みドメイン ID の複製
- 本番稼働の制限の複製

設定セットの複製

プライマリリージョンから複数の設定セットを選択して、セカンダリリージョンで設定とともに複製できます。

「設定セットの複製」機能を使用すると、次のことができます。

- 複数の設定セットを一度にセカンダリリージョンに複製します。
- プライマリリージョンとセカンダリリージョンの設定セットの違いを確認します。

設定セットを複製するには：

1. グローバルエンドポイントページで、名前列から選択して、複製するグローバルエンドポイントを選択します。
2. 重複する設定セットカードで、設定セットアクションを展開し、重複を選択します。
3. 最大 10 個の設定セットを選択し、その後に確認を選択します。
4. ステータスが成功しなかった場合は、レポートを表示を選択して問題を特定します。
5. （オプション) 以前に複製した設定セットの場合、最後の 3 つのステップを繰り返しながら、差異の確認を選択して、プライマリリージョンとセカンダリリージョンの違いを確認できます。

> **Note**
>
> 複製した設定セットにイベント送信先または評価オプションが含まれている場合、または E メールテンプレートで参照されている場合、これらの設定はセカンダリリージョンで手動で設定する必要があります。

検証済みドメイン ID の複製

グローバルエンドポイント設定を効果的に機能させるには、プライマリリージョンとセカンダリリージョンの両方で送信ドメイン ID を検証する必要があります。SES は 決定論的簡単 DKIM (DEED) を使用してこのプロセスを簡素化します。

Deterministic Easy DKIM (DEED) は、Easy DKIM で設定された親ドメイン AWS リージョン に基づいて、すべての で一貫した DKIM トークンを生成する機能です。 ???この整合性により、SES は、プライマリリージョンで検証されたドメインを、追加の DNS レコードの更新を必要とせずに、セカンダリリージョンで自動的に検証できます。したがって、複製するドメイン ID、つまり親が Easy DKIM で既に設定されていることを確認する必要があります。

「検証済みドメイン ID の複製」機能を使用すると、次のことができます。

- 複数のドメイン ID をセカンダリリージョンに一度に複製します。
- Deterministic Easy DKIM (DEED) を使用して自動的に検証します。
- プライマリリージョンとセカンダリリージョンの ID の違いを確認します。

SES コンソールから ID を複製するには：

1. グローバルエンドポイントページで、名前列から選択して、複製するグローバルエンドポイントを選択します。
2. 重複検証済みドメイン ID カードで、ID アクションを展開し、重複を選択します。
3. 最大 10 個の ID を選択し、確認を選択します。
4. ステータスが成功しなかった場合は、レポートを表示を選択して問題を特定します。
5. （オプション) 以前に重複した ID の場合、最後の 3 つのステップを繰り返しながら、差異の確認を選択すると、プライマリリージョンとセカンダリリージョンの違いを確認できます。

> **Note**
>
> - BYODKIM で検証されたドメイン ID、または自己署名のドメイン ID は、DEED はこの場合適用されないため、セカンダリリージョンで手動で作成する必要があります。
> - Mail-from 属性、ポリシー、またはフィードバック転送と通知を使用するドメイン ID では、セカンダリリージョンでこれらの機能を手動で設定する必要があります。

本番稼働の制限の複製

SES は、送信制限がリージョン間で調整されているかどうかをチェックし、必要に応じてセカンダリリージョンで制限の引き上げをリクエストできます。

「本番稼働用制限の複製」機能を使用すると、次のことができます。

- 本番稼働の制限がプライマリリージョンとセカンダリリージョン間で調整されているかどうかを確認します。
- 必要に応じて、セカンダリリージョンで制限の引き上げをリクエストします。

本番稼働の制限を複製するには：

1. グローバルエンドポイントページで、名前列から選択して、複製するグローバルエンドポイントを選択します。
2. 重複本番稼働用制限カードで、ステータスに「送信制限が調整されていない」と表示される場合は、送信制限アクションを展開します。
3. セカンダリリージョンの送信制限の管理を選択します。
4. Service Quotas ページがセカンダリリージョンで開き、プライマリリージョンの値と一致するように「送信クォータ」と「送信レート」への引き上げをリクエストできます。

> **Tip**
>
> 両方のリージョンで対象となる最大クォータをリクエストすることをお勧めします。E メールトラフィックは通常の運用条件下で両リージョンに分散されますが、フェイルオーバーイベント中は、E メールトラフィックの全量が 1 つのリージョンに送信され、その制限は全ボリュームの負荷を処理するのに十分なはずです。

5. (オプション) 前の 2 つのステップを繰り返しながら、プライマリリージョンの送信制限の管理を選択して、プライマリリージョンの本番稼働の引き上げをリクエストすることもできます。

> **Important**
>
> グローバルエンドポイントの適切な機能を確保するためには、両方のリージョンで E メールを送信する予定の同等の検証済み ID と設定セット、および一致する送信制限を持つことが重要です。不一致があると、配信の失敗、フェイルオーバーの信頼性の低下、メトリクスの欠落が発生する可能性があります。

グローバルエンドポイントの使用

トピック

アプリケーションとの統合

アプリケーションでグローバルエンドポイントを使用するには、エンドポイント ID を取得する必要があります。

グローバルエンドポイントのエンドポイント ID を取得するには：

1. SES コンソールからグローバルエンドポイントページに移動し、名前列から選択して、使用するグローバルエンドポイントを選択します。
2. グローバルエンドポイントの詳細ページのエンドポイント ID にあるコピーアイコンを選択します。

の使用 AWS CLI：

```
aws sesv2 get-multi-region-endpoint --endpoint-name MyGlobalEndpoint --region us-west-2
```

前の例では、以下のようになっています。

- *MyGlobalEndpoint* を、作成時にグローバルエンドポイントに付けたわかりやすい名前に置き換えます。
- *us-west-2* を、グローバルエンドポイントを作成したプライマリリージョンに置き換えます。
- API レスポンスには、 などのエンドポイント ID の値が含まれます"EndpointId": "abcdef12.g3h"。

グローバルエンドポイントのエンドポイント ID を取得したら、 SendEmailまたは SendBulkEmail API コールを更新して、 endpoint-idパラメータのエンドポイント ID 値を含めることができます。を使用して SendEmail API コールでエンドポイント ID を指定する方法の例を次に示します AWS CLI。

```
aws sesv2 send-email \
                --from-email-address "sender@example.com" \
                --destination "ToAddresses=recipient@example.com" \
                --content "Subject={Data=Test
 email,Charset=UTF-8},Body={Text={Data=This is a test email sent using Amazon SES
 Global endpoints.,Charset=UTF-8}}" \
                --endpoint-id "abcdef12.g3h"
```

abcdef12.g3h を、コンソールまたは API で取得した実際のエンドポイント ID に置き換えます。

モニタリングおよびメトリクス

グローバルエンドポイント機能は、プライマリリージョンとセカンダリリージョンの両方で E メール送信ボリュームの統合ビューを提供します。これらのメトリクスには、SES コンソールのグローバルエンドポイントの詳細ページのクロスリージョンメトリクスタブからアクセスできます。

両方のリージョンの送信メトリクスにアクセスするには：

1. SES コンソールからグローバルエンドポイントページに移動し、名前列からメトリクスを選択して、メトリクスを表示するグローバルエンドポイントを選択します。
2. グローバルエンドポイントの詳細ページでクロスリージョンメトリクスタブを選択し、最大 31 日間の日付範囲を入力します。両方のリージョンのメトリクスは、指定された日付範囲で表示されます。

の使用 AWS CLI：

```
aws cloudwatch get-metric-statistics \
```

```
    --namespace AWS/SES \
    --metric-name SendCount \
    --dimensions Name=ses:multi-region-endpoint-id,Value=abcdef12.g3h \
    --start-time 2024-10-01T00:00:00Z \ --end-time 2024-10-31T23:59:59Z \
    --period 86400 \
    --statistics Sum
```

abcdef12.g3h を実際のエンドポイント ID に置き換えます。

ベストプラクティスと考慮事項

これらのベストプラクティスと考慮事項に従うことで、複数の にわたるグローバルエンドポイントの効果的な使用率、モニタリング、コスト最適化を確保し AWS リージョン 、E メール送信機能の可用性と信頼性を向上させることができます。

- リージョン間でアーティファクト (設定セット、検証済み ID など) に加えられた変更を定期的に同期して、送信の整合性を維持します。
- クロスリージョンメトリクスをモニタリングして、バランスの取れたトラフィック分散を確保し、潜在的な問題を特定します。
- グローバルエンドポイントは可用性を向上させますが、SES アウトバウンドのリージョンの可用性の物理的な状態は変更されないことに注意してください。
- 起動時に、グローバルエンドポイントは SMTP または VPC エンドポイントアクセスをサポートしていないことに注意してください。
- AWS アドレス変換ゲートウェイを使用する場合は、潜在的な出力料金を考慮してください。
- MREP 対応の遠いリージョンを呼び出すと、API レイテンシーがわずかに増加する可能性があることに注意してください。

料金

正確な料金の詳細は変更される可能性がありますが、グローバルエンドポイントは、同じ量のメールに対して単一リージョン送信よりも料金プレミアムを支払うことが予想されます。この増加にもかかわらず、全体的なコストは他の E メールサービスプロバイダーと比較して競争が続くと予想されます。

up-to-date料金情報については、Amazon SES 料金ページを参照してください。

Amazon 専用 IP アドレス SES

新しい Amazon SESアカウントを作成すると、デフォルトでは、SES他のユーザーと共有されている IP アドレスから E メールが送信されます。また、追加料金でリースすることで、お客様専用の IP アドレスを利用できます。これにより、送信者評価を完全に管理でき、E メールプログラム内のさまざまなセグメントに対する評価を分離できます。Amazon SES には、専用 IP アドレスをプロビジョニングおよび管理するための 2 つの方法があります。

- 標準 — 手動で設定と管理を行う専用 IP アドレスです。手動でウォームアップやスケールアウトを行うオプションや、手動で IP プールに出し入れするオプションが含まれます。(これらは、以前は で専用 IP アドレスと呼ばれていました) SES。
- マネージド型 - によって管理される専用 IP アドレスを迅速かつ簡単に使用開始SESできるように、 によって自動的にセットアップされる専用 IP アドレスを指しますSES。各専用 IP アドレスは、送信ボリュームに基づいてISP個別に自動的にウォームアップされ、自動スケーリングされるため、E メールの送信方法に基づいて専用 IP アドレスが最適に使用されるようになります。

共有 IP アドレスと、上で定義した専用 IP アドレス (2 種類) のどちらを選択するかを決める場合は、送信する E メールの種類、量、パターンに応じて最も多くの利点が得られる方を選択します。容易に判断できるように、これらの利点を次の表にまとめました。追加の情報については、[Benefit] (利点) 列の項目を選択してください。

利点	共有 IP アドレス	専用 IP アドレス (標準)	専用 IP アドレス (マネージド)
すぐに使用可能	あり	いいえ	いいえ
追加の設定が必要	いいえ	あり	あり
他のSES顧客から分離された IP アドレスと評価	いいえ	あり	あり
トラフィックが増加すると自動的にキャパシティが増加する	いいえ	なし	あり

利点	共有 IP アドレス	専用 IP アドレス (標準)	専用 IP アドレス (マネージド)
継続的で予測可能な送信パターンを持つお客様向け	はい	可能	はい
予測可能生が低い送信パターンを持つお客様向け	はい	なし	あり
送信量が多い送信者向け	はい	可能	はい
送信量が少ない送信者向け	はい	いいえ	いいえ
追加の月額料金	いいえ	あり	あり
送信者の評価に関する完全な制御	いいえ	あり	あり
メールの種類、受取人、またはその他の要因別の独立した評価	いいえ	あり	あり
変更されることがない既知の IP アドレスの提供	いいえ	あり	不可

Important

大量の E メールを定期的に送信する予定がない場合は、共有 IP アドレスの使用をお勧めします。送信パターンが非常に不規則な状況で専用 IP アドレスを使用する場合は、Dedicated IPs (マネージド) を使用することをお勧めします。

容易なセットアップ

共有 IP アドレス — 追加設定を行う必要はありません。E メールアドレスを確認してサンドボックス外に移動するとすぐに、SES アカウントはメールを送信できる状態になります。

専用 IP アドレス (標準) — AWS サポートセンターからリクエストを送信し、オプションで専用 IP プールを設定する必要があります。

専用 IP アドレス (マネージド) — 専用 IP アドレスのリクエストを送信する必要はありません。オプトインしてウォークスルーを 1 回だけ実行し、マネージド専用プールを作成すると、自動的に割り当てられます。

評価管理

IP アドレスの評価は、主に送信パターンと送信量の履歴に基づきます。長期にわたって一定量の E メールを送信している IP アドレスは、通常、評価が高くなります。

共有 IP アドレス - 複数のSES顧客間で共有されているこれらのアドレスは、まとめて大量の E メールを送信し、アウトバウンドトラフィックを AWS 慎重に管理して、共有 IP アドレスの評価を最大化します。

専用 IP アドレス (標準) — ウォームアップ後、IP アドレスはSES共有プールから分離され、一貫した予測可能な量の E メールを送信することで、送信者の評価を維持します。

> **Note**
>
> 専用 (標準SNDS) のスマートネットワークデータサービス IPs () データの詳細については、「」を参照してくださいSNDS 専用 の メトリクス IPs。

専用 IP アドレス (マネージド) — 新しい のウォームアップ後IPs、共有SESプールから分離され、送信者の評価を維持します。それぞれの評価を追跡ISPし、それに応じて送信送信を最適にスケジュールする利点があります。そのため、この自動化によって、送信者の評価を維持しながら、手動で設定した専用 IP アドレスによる同等のワークロードと比較して、全体的な配信性の向上とバウンス率の低減に貢献します。

送信パターンの予測可能性

E メール送信の一貫した履歴を持つ IP アドレスは、以前の送信履歴がなくて急に大量の E メール送信を開始する IP アドレスよりも評価が高くなります。

共有 IP アドレス — 予測可能なパターンとは異なる E メール送信パターンに適しています。共有 IP アドレスを使用すると、状況に応じて E メールの送信パターンを増減できます。

専用 IP アドレス (標準) — E メールの送信量を毎日少しずつ増やして、アドレスをウォームアップする必要があります。新しい IP アドレスをウォームアップするプロセスについては、「専用 IP アドレス (標準) のウォームアップ」で説明しています。専用 IP アドレスのウォームアップが完了したら、一貫した送信パターンを維持する必要があります。

専用 IP アドレス (マネージド) - 専用 IP アドレスは、ISP実際の送信パターンを考慮して個別にウォームアップを最適化する適応型ウォームアップ戦略 (SES共有プールと組み合わせて) を使用して、マネージドプール内の IP ごとに自動的にウォームアップされます。マネージド IP プールは、ISP固有のポリシーの使用と考慮事項ISPに基づいて、 ごとに自動的にスケールアウトされます。

送信メールのボリューム

共有 IP アドレス — 送信する E メールの量が少ないお客様に最適です。

専用 IP アドレス (標準) | 専用 IP アドレス (マネージド) — どちらも大量の E メールを送信するお客様に適しています。ほとんどの は、そのアドレスから大量のメールを受け取った場合にのみ、特定の IP アドレスの評価ISPsを追跡します。評価を育む各 について、少なくとも 1 か月に ISP 1 回、24 時間以内に数百通の E メールを送信する必要があります。場合によっては、どちらのタイプの専用 IP アドレスでも、少量の E メールにも有効な場合があります。例えば、正しく定義された小グループの受信者が、IP アドレスの評価ではなく特定の IP アドレスのリストに基づいて E メールを承認または拒否するメールサーバーを利用している場合、これらの受信者に送信するにはこれらの専用 IP アドレスが適していることがあります。

追加料金

共有 IP アドレス — 標準SES料金に含まれています。

専用 IP アドレス (標準) — リースする IP アドレスごとに、追加の月額料金がかかります。料金情報については、SES の料金表ページを参照してください。

専用 IP アドレス (マネージド) — 標準の月額料金 (IPs必要な の量に関係なく) とメッセージごとの使用料金で利用できます。料金情報については、SES の料金表ページを参照してください。

送信者の評価に関する制御

共有 IP アドレス — 送信者の評価は によって制御されますSES。

専用 IP アドレス (標準) | 専用 IP アドレス (マネージド) — 送信者評価は、すべてお客様の管理となります。これらのアドレスから E メールを送信できるのは、 SESアカウントだけです。したがって、送信者評価は、E メール送信状況に依存します。さらに、専用 IPs (マネージド) は、最もパフォーマンスの高い IP アドレスを使用して E メール送信に使用されるアウトバウンド IP アドレスを積極的にモニタリングし、受信者への E メール配信性能を向上させます。使用率データは、Amazon CloudWatch メトリクスや Amazon にある組み込みダッシュボードなどの追加サービスを使用して表示できますSES。

送信者の評価の分離可能性

共有 IP アドレス — 送信者評価はアカウントレベルで設定されるため、分離できません。

専用 IP アドレス (標準) | 専用 IP アドレス (マネージド) — 専用 IP プール (特定の種類の E メールの送信に使用可能な専用 IP アドレスグループ) を作成することで、E メールプログラム内の異なるコンポーネントに対して送信者評価を分離できます。例えば、マーケティングメールの送信用と取引メールの送信用に専用 IP アドレスのプールをそれぞれ作成できます。

変わることがない既知の IP アドレス

共有 IP アドレス - がメールの送信SESに使用する IP アドレスは不明であり、いつでも変更できます。

専用 IP アドレス (標準) — メールを送信するアドレスの値は、SESコンソールの専用IPsページにあります。これは、専用 IP アドレスが静的であるためです。

専用 IP アドレス (マネージド) -SES は、送信パターンに基づいて最適な数の専用 IP アドレスを自動的に設定します。つまり、プール内の専用 IP アドレスは表示されず、需要に応じて動的に増減します。

Amazon SES での専用 IP アドレス (標準)

専用 IP アドレス (標準) は、SES において手動で設定と管理を行う専用 IP アドレスです。SES 機能の the section called “マネージド” を使用して自動で設定および管理するものとは異なります。専用 IP アドレスを使用して送信評価を完全に管理できることに加え、専用 IP (標準) では、ウォームアップ、スケールアウト、IP プール管理など、専用 IP を完全に管理できます。

専用 IP (標準) と専用 IP (マネージド) は、どちらも追加料金により SES でリースする専用 IP アドレスを指しますが、実装方法と管理方法が異なります。両方に共通する利点はありますが、「専用 IP アドレス」で説明したように、E メール送信の種類によってそれぞれ異なる利点があります。

このセクションのトピックでは、SES で専用 IP (標準) を手動で設定と管理を行う方法について説明します。

トピック

- 専用 IP アドレスのリクエストと解放 (標準)
- 専用 IP アドレス (標準) のウォームアップ
- 専用 IP (標準) 用の専用 IP プールの作成

専用 IP アドレスのリクエストと解放 (標準)

専用 IP アドレス (標準) を使用するには、最初にリクエストする必要があります。また、不要になった場合は、解放する必要があります。AWS サポート センターを通じて 専用 IP (標準) のリクエストと解放を行います。お客様のアカウントには、Amazon SES で使用するためにリースする標準専用 IP アドレスごとに月額料金が追加で請求されます。専用 IP (標準) を使用する場合、最低コミットメントはありません。

専用 IP (標準) についてのコストの詳細については、「Amazon SES 料金表」を参照してください。

Amazon SES が現在利用可能なすべてのリージョンの一覧については、「Amazon Web Services 全般のリファレンス」の「AWS リージョンとエンドポイント」を参照してください。各 AWS リージョン で利用可能なアベイラビリティーゾーン数の詳細については、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。

専用 IP (標準) をリクエストする

AWS サポートセンターでサービスクォータの引き上げのケースを作成して、必要な専用 IP (標準) をリクエストできます。

専用 IP (標準) をリクエストするには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションペインで、[Dedicated IPs] (専用 IP) を選択します。

3. 次のいずれかを行います。

   a. アカウントに既存の専用 IP がない場合:

      - [Dedicated IPs] (専用 IP) オンボーディングページが表示されます。[Dedicated IPs (standard) overview] (専用 IP (標準) の概要) パネルで、[Request dedicated IPs] (専用 IP のリクエスト) を選択します。

        AWS サポートコンソールで [Create case] (ケースの作成) ページが開きます。

   b. アカウントに既存の専用 IP がある場合:

      i. [Dedicated IPs] (専用 IP) ページで [Standard IP pools] (標準 IP プール) タブを選択します。

      ii. [Standard overview] (標準の概要) パネルで、[Request or relinquish Standard dedicated IPs] (標準専用 IP のリクエストまたは解放) を選択します。

        AWS サポートコンソールで [Create case] (ケースの作成) ページが開きます。

4. [Create case] (ケースの作成) で、ページ上部にある [Service limit increase] (サービス上限の引き上げ) カードを選択します。
5. [Case details] で、以下のセクションを完了します。

   - [Limit type] (制限のタイプ) で [SES Service Limits] (SES サービスの制限) を選択します。
   - [Mail Type (メールの種類)] で、専用 IP アドレスを使用して送信を予定しているメールのタイプを選択します。複数の値が当てはまる場合は、送信する E メールの大部分に当てはまるオプションを選択します。
   - [Website URL] に、ウェブサイトの URL を入力します。この情報を提供することで、お客様が送信する予定のコンテンツタイプを正しく理解するのに役立ちます。
   - 特に、E メールをリクエストした受取人のみに送信する方法を詳細に説明する場合は、ユースケースに沿って応答します。
   - バウンスやクレームの通知を受け取った際に実行するプロセスを詳細に説明する場合は、ユースケースと整合性を取って応答します。
   - [Will you comply with AWS Service Terms and AUP (サービス規約と AUP に準拠していますか)] で、ユースケースに当てはまるオプションを選択します。
6. [Requests] で、以下のセクションに入力します。

   - [Region (リージョン)] で、リクエストが適用されるAWS リージョンを選択します。
   - [Limit] (制限) の場合は、[Desired Dedicated IP] (必要な専用 IP) のままにします。

- [New limit value] (新しい制限値) で、ユースケースを実装するために必要な専用 IP アドレスの数を入力します。

> **Note**
>
> 他の AWS リージョン で使用する専用 IP アドレスをリクエストする場合は、[Add another request] (別のリクエストを追加) を選択して、別の AWS リージョン について [Region] (リージョン)、[Limit] (制限)、[New limit value] (新しい制限値) の各フィールドを入力します。専用 IP アドレスを使用する各 AWS リージョン でこのプロセスを繰り返します。

7. Case description の [Use case description] に、専用 IP アドレスをリクエストする旨を記載します。特定の数の専用 IP アドレスを希望する場合は、それも記載します。専用 IP アドレスの数を指定しない場合は、前のステップで指定された送信レートの要件を満たすために必要な数の専用 IP アドレスが提供されます。

   次に、Amazon SES を使用して専用 IP アドレスから E メールを送信する方法について説明します。共有 IP アドレスではなく専用 IP アドレスを使用する理由に関する情報も含めます。この情報は、ユースケースを理解するために役立ちます。
8. [連絡先オプション] の [優先される問い合わせ言語] で、このケースに関する連絡を [英語] で受け取るか [日本語] で受け取るかを選択します。
9. 完了したら、[送信] を選択します。

フォームの送信後、リクエストが評価されます。リクエストが承認された場合は、サポートセンターでケースに返信し、新しい専用 IP アドレスがアカウントに関連付けられることを確認します。

標準専用 IP アドレスを解放する

専用 IP アドレスを使用していて、アカウントへの関連付けが不要になった場合、以下の手順に示されている手順で AWS サポートセンターにケースを作成して、それらを解放できます。

> **Important**
>
> 専用 IP アドレスの解放プロセスは、元に戻すことはできません。月の途中で専用 IP アドレスを解放した場合、専用 IP の月額使用料金を当月の経過日数に基づいて日割りで計算します。

専用 IP (標準) を解放するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションペインで、[Dedicated IPs] (専用 IP) を選択します。
3. [Dedicated IPs] (専用 IP) ページで [Standard IP pools] (標準 IP プール) タブを選択します。
4. [Standard overview] (標準の概要) パネルで、[Request or relinquish Standard dedicated IPs] (標準専用 IP のリクエストまたは解放) を選択します。
5. [Case details] (ケース詳細) の [Limit type] (制限タイプ) を [SES Service Limits] (SES サービス制限) のままにします。

> **Note**
>
> このセクションの残りのボックスは、専用 IP の解放には適用されません。この部分は空白のままにします。

6. [Requests] で、以下のセクションに入力します。

   - [Region] (リージョン) で、解放リクエストが適用される AWS リージョン を選択します。

     > **Note**
     >
     > 専用 IP アドレスは各 AWS リージョン に固有のものです。そのため、専用 IP アドレスが関連付けられる AWS リージョン を選択することが重要です。

   - [Limit] (制限) の場合は、[Desired Dedicated IP] (必要な専用 IP) のままにします。
   - [New limit value (新しい制限値)] で任意の数を入力します。ここで入力する番号は重要ではありません。次のステップで、放棄する専用 IP の数を指定します。

     > **Note**
     >
     > 単一の専用 IP アドレスは、単一のAWS リージョンでのみ使用できます。他の AWS リージョン で使用している専用 IP アドレスを解放する場合は、[Add another request] (別のリクエストを追加) を選択します。次に、追加する AWS リージョン の [Region] (リージョン)、[Limit] (制限)、[New limit value] (新しい制限値) フィールドを入力します。開放する専用 IP アドレスごとに、このプロセスを繰り返します。

7. Case Description (ケースの説明) の Use case description (ユースケースの説明) に、既存の専用 IP アドレスの解放を希望する旨を記載します。現在複数の専用 IP アドレスをリースしている場合は、解放する専用 IP アドレスの数を含めます。
8. [連絡先オプション] の [優先される問い合わせ言語] で、このケースに関する連絡を [英語] で受け取るか [日本語] で受け取るかを選択します。
9. 完了したら、[送信] を選択します。

リクエストが送信すると、専用 IP アドレスを解放する希望を確認するメッセージを受信できます。IP アドレスを解放することが確認されると、アドレスがアカウントから削除されます。

専用 IP アドレス (標準) のウォームアップ

E メールサービスプロバイダーは、メッセージを受け入れるか拒否するかを判断するとき、その送信元の IP アドレスの評価を考慮します。IP アドレスの評価に影響する要因の 1 つは、アドレスに高品質 E メールの送信履歴があるかどうかです。E メールプロバイダーが、ほとんど、または、まったく履歴のない新しい IP アドレスからメールを受け入れる可能性はあまりありません。ほとんど、または、まったく履歴のない IP アドレスから送信された E メールは、受信者の迷惑メールフォルダに振り分けられるか、完全にブロックされる可能性があります。

新しい専用 IP アドレスから E メールの送信を開始する場合は、そのアドレスから送信する E メールの量を徐々に増やし、その後でアドレスを最大容量まで使用する必要があります。このプロセスは IP アドレスのウォームアップと呼ばれます。

IP アドレスのウォームアップに必要な時間は、E メールプロバイダーによって異なります。一部の E メールプロバイダーでは 2 週間前後で良い評価を確立できますが、最大 6 週間かかるプロバイダもあります。新しい専用 IP アドレスをウォームアップするときは、最もアクティブなユーザーに E メールを送信して、苦情率が低いことを確認する必要があります。また、バウンスメッセージを慎重に調べて、ブロックまたはスロットリング通知を多数受け取っている場合は、E メールの量を減らします。バウンスのモニタリングについては、「Amazon SES 送信アクティビティのモニタリング」を参照してください。

専用 IP (標準) の自動ウォームアップ

専用 IP アドレス (標準) をリクエストすると、Amazon SES では自動的にウォームアップして、送信する E メールの配信を向上させます。IP アドレスの自動ウォームアップ機能はデフォルトで有効になっています。SES では、事前に定義されたウォームアッププランに基づいて、専用 IP を介して送信する E メールの数を徐々に増やすことで、専用 IP を自動的にウォームアップします。このように

段階的に増加させることで、IP はインターネットサービスプロバイダー (ISP) から高い評価を得られます。

自動ウォームアップの処理中に発生するステップは、既に専用 IP アドレスを持っているかによって異なります。

- 専用 IP アドレス (標準) を初めてリクエストすると、SES では、専用 IP アドレスと、他の SES のお客様と共有するアドレスとの間で E メールの送信を分散します。SES では、専用 IP アドレスから送信されるメッセージの数を時間の経過ともに徐々に増やします。
- 専用 IP アドレスを既にお持ちの場合、SES は既存の専用 IP (ウォームアップ済み) と新しい専用 IP (ウォームアップされていない) との間で E メールの送信を分散します。SES では新しい専用 IP アドレスから送信されるメッセージの数を時間の経過ともに徐々に増やします。

> **Note**
>
> 自動 IP ウォームアップは時間ベースのプロセスです。ウォームアップ率は、送信量とは無関係に 45 日間にわたって着実に増加します。

専用 IP アドレスのウォームアップ後は、良い評判を維持することを希望する E メールプロバイダーごとに、毎日 1,000 通前後の E メールを送信します。このタスクは、SES で使用する専用 IP アドレスごとに実行します。

ウォームアップ処理の完了直後に、大量の E メールを送信することは避けてください。その代わりに、ターゲットのボリュームに達するまで、送信する E メールの数を徐々に増やします。E メールプロバイダーでは、IP アドレスから送信される E メールの数が急に大量に増加したことを確認すると、そのアドレスからのメッセージの配信をブロックまたはスロットリングする場合があります。

専用 IP (標準) の自動ウォームアッププロセスを無効にする

標準専用 IP アドレスを新規で購入すると、アカウントでは IP アドレスの自動ウォームアップ機能がデフォルトで有効になっているため、Amazon SES によって自動ウォームアップします。専用 IP アドレスをご自身でウォームアップすることを希望する場合は、アカウントレベルで、すべての IP アドレスの自動ウォームアップ機能を無効にできます。

自動ウォームアップ機能を無効にすると、その後にリースされる専用 IP はウォームアップステータスが [Complete] (完了) の状態でアカウントに追加され、ウォームアップなしで使用できるようになります。この場合は、お客様において通常の送信に使用する前にこれらの IP を適切にウォームアッ

プする必要があります。自動ウォームアップ機能を無効にした時点で、ウォームアップ中だった IP は影響を受けません。

> **Important**
>
> 自動ウォームアップ機能を無効にする場合、専用 IP アドレスのウォームアップはお客様の責任となります。ウォームアップされていないアドレスから E メールを送信する場合、配信レートが低下する可能性があります。

アカウントのすべての専用 IP (標準) の自動ウォームアップ機能を無効にする (または再び有効にする) には

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションペインで、[Dedicated IPs] (専用 IP) を選択します。
3. [Dedicated IPs] (専用 IP) ページで [Standard IP pools] (標準 IP プール) タブを選択します。
4. [Standard overview] (標準の概要) パネルで [Disable auto warm-up] (自動ウォームアップを無効にする) を選択して自動ウォームアップを無効にするか、[Enable auto warm-up] (自動ウォームアップを有効にする) を選択して、自動ウォームアップを再び有効にします。

専用 IP (標準) を手動でウォームアップする

専用 IP (標準) のウォームアップ率を編集すると、現在の送信量を手動で増減する、ウォームアッププロセスを途中で終了する、現在の送信量を 0% に設定してウォームアッププロセスを再開できます。

専用 IP (標準) を手動でウォームアップするには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションペインで、[Dedicated IPs] (専用 IP) を選択します。
3. [Dedicated IPs] (専用 IP) ページで [Standard IP pools] (標準 IP プール) タブを選択します。
4. [All Standard dedicated IPs] (すべての標準専用 IP) パネルで、IP アドレスを選択し、[Edit warm up] (ウォームアップの編集) で、次のいずれかのオプションを選択します。

a. [Edit percentage] (割合の編集) — [Warm-up percentage] (ウォームアップ率) フィールドに値を入力し、ウォームアップ率を編集して IP の現在の送信量を増減し、[Save changes] (変更を保存) をクリックします。

   [Warm-up status] (ウォームアップステータス) 列には `In progress` が表示され、[Warm-up percentage] (ウォームアップ率) 列には入力した値が表示されます。

b. [Mark as Complete] (完了としてマークする) — [Mark warm-up as Complete?] (ウォームアップを完了してマークしますか) ダイアログを読んで、自動ウォームアッププロセスを途中で終了した場合の意味を理解したことを確認し、[Mark as Complete] (完了としてマークする) を選択します。

   [Warm-up status] (ウォームアップステータス) 列には `Complete` が表示され、[Warm-up percentage] (ウォームアップパー率) 列には `100%` が表示されます。

c. [割合のリセット] — [ウォームアップ率をリセットしますか] ダイアログを読み、IP の現在の送信量を 1% に設定することを確認して、自動ウォームアッププロセスを再開するか、ウォームアップ率を手動で設定してから、[リセット] を選択します。

   [Warm-up status] (ウォームアップステータス) 列には `In progress` が表示され、[Warm-up percentage] (ウォームアップ率) 列には `1%` が表示されます。

専用 IP (標準) 用の専用 IP プールの作成

Amazon SES で使用する複数の専用 IP アドレス (標準) を購入した場合、専用 IP プールと呼ばれるアドレスグループを作成できます。専用 IP (標準) を 1 つのプールにまとめると、管理がしやすくなります。一般的なケースとしては、マーケティング E メール送信用と取引 E メール送信用にそれぞれプールを作成します。取引 E メールの送信者の評価はマーケティング E メールの送信者の評価とは切り離されています。このシナリオでは、マーケティングキャンペーンで多数の苦情が発生した場合、取引 E メールの配信には影響しません。

このセクションでは、専用 IP プールを作成する手順について説明します。

> **Note**
>
> すべての SES のお客様が共有する IP アドレスのプールを使用する設定セットを作成することもできます。共有 IP プールは、通常の送信動作と異なる E メールを送信する必要がある

場合に便利です。設定セットで共有 IP プールを使用する方法については、「Amazon SES での IP プールの割り当て」を参照してください。

SES コンソールを使用して専用 IP (標準) 専用 IP プールを作成するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションペインで、[Dedicated IPs] (専用 IP) を選択します。

   > **Note**
   >
   > 現在、アカウントに専用 IP (標準) がない場合は、専用 IP (標準) を購入できる [Dedicated IPs] (専用 IP) オンボーディングページが表示されます。詳細については、「the section called “専用 IP (標準) をリクエストする”」を参照してください。

3. [Dedicated IPs] (専用 IP) ページで [Standard IP pools] (標準 IP プール) タブを選択します。
4. [All Dedicated IP (standard) pools] (すべての専用 IP (標準) プール) パネルで、[Create Standard IP pool] (標準 IP プールの作成) を選択します。

   [Create IP Pool] (IP プールの作成) ページが開きます。
5. [Pool details] (プール詳細) パネルで、

   a. [Scaling mode] (スケーリングモード) フィールドで [Standard (self managed)] (標準 (セルフマネージド)) を選択します。

   b. [IP pool name] (IP プール名) フィールドに IP プールの名前を入力します。

      > **Note**
      >
      > IP プール名は一意である必要があり、アカウントのマネージド IP プール名と同じ名前にはできません。

   c. (オプション) この IP プールに追加する既存の標準専用 IP アドレスがある場合は、[Dedicated IP addresses] (専用 IP アドレス) フィールドのドロップダウンリストからいずれかを選択します。

> **Note**
>
> 既に IP プールに関連付けられている IP アドレスを選択した場合、この IP プールのみに関連付けられるようになります。

6. (オプション) この IP プールを設定セットに関連付けるには、[Configuration sets] (設定セット) フィールドのドロップダウンリストからいずれかを選択します。

> **Note**
>
> - 既に IP プールに関連付けられている設定セットを選択した場合、この IP プールのみに関連付けられるようになります。
> - この IP プールの作成後に関連する設定セットを追加または削除するには、設定セットの [Sending IP pool] (IP プールの送信) パラメータを編集します。
> - 設定セットを作成していない場合は、「設定セット」を参照してください。

7. (オプション)1 つまたは複数のタグをこの IP プールに追加するには、タグキーとキーのオプションの値を指定します。

   a. 新しいタグを追加を選択し、キーを入力します。任意でタグに値を追加できます。

   b. タグを追加するには、[Save changes] を選択します。

   最大 50 個のタグを追加できます。削除を選択すると、タグを削除できます。

8. [Create pool] (プールの作成) を選択します。

> **Note**
>
> 標準 IP プールを作成した後で、それをマネージド IP プールに変換するオプションはありません。「マネージド IP プールの作成」を参照してください。

Amazon SES の専用 IP アドレス (マネージド)

専用 IP アドレス (マネージド) は、お客様に代わって専用 IP アドレスの設定と管理を自動で行う Amazon SES の機能です。これにより、SES が管理する専用 IP アドレスをすばやく簡単に使用開始

できます。これにより、専用 IP アドレスを、お客様の E メールの送信方法に対して効率よく、最適な状態で使用できます。

アカウントで専用 IP (マネージド) を有効にするには、マネージド IP プールを作成するだけです。SES によって他のすべての処理が実行されます。SES は、送信パターンに基づいて必要な専用 IP の数を決定して作成し、送信要件に基づいてスケールする方法を管理します。

有効にすると、マネージド IP プールを設定セットに関連付け、E メールの送信時にその設定セットを指定することで、E メール送信に専用 IP (マネージド) を利用できます。また、デフォルトの設定セットを使用して、設定セットを送信 ID に適用できます。

専用 IP (マネージド) の利点と機能

専用 IP (マネージド) で作成した専用 IP アドレスは、管理タスクを自動化して、E メールの送信方法に最適な方法で専用 IP アドレスを使用できるようにします。

- 簡単なオンボーディング — 専用 IP (マネージド) を使用開始するには、SES コンソールから直接マネージド IP プールを作成します。専用 IP アドレスがプールに自動的に割り当てられます。AWS サポートセンターからリクエストケースを開かなくても、マネージド IP プールを使用して送信を開始できます。
- ISP ごとの自動スケーリング – マネージド IP プールは使用状況に基づいて自動的にスケールアウトするため、専用 IP プールは手動でモニタリングしたりスケールしたりする必要はありません。ISP 固有のポリシーも考慮されます。例えば、SES によって ISP の 1 日の送信クォータが低いことを検出した場合、プールをスケールアウトして、その ISP へのトラフィックをより多くの IP アドレスに分散させます。
- インテリジェントウォームアップ — 専用 IP (マネージド) は、その容量に応じて ISP に E メールを送信し始めます。これは、現在どの程度ウォームアップされているかによります。各 ISP のウォームアップレベルを個別に自動で追跡します。さらに、専用 IP (マネージド) 機能では、Amazon CloudWatch メトリクスや組み込みダッシュボードの形式で、トップ ISP に実質的な日別レートで評価に関する情報を提供します。
  - ISP ごとのウォームアップ — SES では、各 ISP のマネージド IP プール内の各 IP の評価を個別に追跡します。例えば、すべてのトラフィックを Gmail に送信している場合、IP アドレスは Gmail でのみウォームアップされ、他の ISP ではコールドと見なされます。Hotmail に送信される E メールを増やしてトラフィックパターンを変更した場合、IP アドレスがまだウォームアップされていないため、SES は Hotmail のトラフィックを徐々に増やします。
  - アダプティブウォームアップ – ウォームアップの調整は、適応型であり、実際の送信パターンを考慮に入れています。ISP への送信量が低下すると、その ISP のウォームアップ率も低下

します。ウォームアップの初期段階では、現在のウォームアップレベルを超えた送信は、他の Amazon SES ユーザーと共有している IP アドレス (SES 共有プール) を介して送信されます。ウォームアップの後半の段階では、ウォームアップレベルを超えた送信は予防的に遅くなり、後で再試行されます。

> **Important**
>
> 専用 IP (マネージド) は専用 IP アドレスを自動的にウォームアップします。この自動プロセスの一部は SES 共有 IP プールとインタラクティブに連携します。
>
> - 新しい専用 IP のウォームアップ中に送信速度が速すぎる場合、SES は新しい専用 IP の評価を保護するために、送信の一部を自動的に SES 共有 IP プールにスピルオーバーします。
> - 新しい専用 IP が完全にウォームアップされた後でも、すべての送信が 100% 確実に実行されるとは限りません。例えば、送信レートが突然上昇し、追加の専用 IP アドレスを割り当てる必要があると専用 IP (マネージド) が判断した場合、共有プールの使用を含むウォームアッププロセスが開始されます。同様に、送信レートが突然非常に低下した場合、すべての送信が SES 共有 IP プールに切り替わる可能性があります。「the section called “ウォームアップの重要性”」を参照してください。

- 専用 IP アドレスの自動リクエストと解放 — 専用 IP(標準) を使用する場合のように、AWS サポートセンターからマネージド専用 IP アドレスをリクエストまたは解放する必要はありません。SES コンソール、CLI、API から直接専用 IP (マネージド) を使用してオンボーディングすると、専用 IP アドレスが自動的に割り当てられ、送信するメッセージの量に応じて料金が請求されます。専用 IP (マネージド) で作成した IP プールを削除したり、オプトアウトしたりすると、割り当てられた IP アドレスは自動的に解放され、課金もすぐに停止します。
  - 最初の専用 IP アドレスの取得 - 専用 IP (マネージド) 機能では、数日間で送信量が数百通のメールに達すると、最初の専用 IP アドレスが自動的に割り当てられます。これにより、送信元の IP が送信者の評価を高め、配信率を向上させることができます (送信量がこのレベルにならないと思われる場合は、共有 IP アドレスを使用してください。メールの送信に最適な IP アドレスの種類を確認するには、専用 IP アドレス の比較表を参照してください)。

適切な IP ウォームアップが重要な理由

メールが専用 IP アドレスを介して配信されるには、受信側の ISP から高い評価を得ている必要があります。ISP は、認識できない IP からは、少量のメールしか受け付けません。最初に割り当てられた IP アドレスは新しく、評価が関連付けられていないため、受信側の ISP には認識されませ

ん。IP の評価を確立するには、受信側の ISP との信頼を徐々に構築する必要があります。この段階的な信頼構築プロセスをウォーミングアップと呼びます。専用 IP (マネージド) が IP を割り当てた直後に、インテリジェントウォームアッププロセスが開始されます。

ISP ごとのウォームアップ機能と専用 IP (マネージド) の適応型ウォームアップ機能により、メールが確実に配信されるため、ウォームアップサイクル全体を通してビジネスの継続性が維持されます。ウォームアップフェーズが完了すると、余剰容量はキューに入れられ、専用 IP プールを介してのみ送信されます。ただし、専用 IP アドレスが 1 つで、送信が IP 評価の維持に必要な最小ボリュームを下回る場合、専用 IP (マネージド) は専用 IP アドレスを削除し、送信は SES 共有 IP プールを経由してルーティングされる場合があります。

> **Note**
>
> メールの送信量が少ない場合 (数日間にわたって毎日数百件未満)、SES 共有 IP プールを介して送信するほうが有益です。専用 IP アドレス の比較表を確認して、専用 IP (マネージド) がメールの送信に適しているかどうかを確認してください。

専用 IP (マネージド) を有効にするためのマネージド IP プールを作成する

専用 IP (マネージド) を有効にするには、最初にマネージド IP プールを作成します。マネージドプールを作成すると、その機能によって送信パターンに基づいて必要な専用 IP の数が決定され、要件に合わせてそれらが動的にスケーリングされます。

マネージドプールを使用して E メールを送信するには、マネージドプールを設定セットに関連付け、E メールの送信時にその設定セットを指定する必要があります。また、デフォルトの設定セットを使用して、設定セットを送信 ID に適用できます。

マネージド IP プールを作成するには、2 つの方法があります。

- 新しいプールを作成します。
- 既存のプールを標準プールからマネージドプールに変換します。

以下では、それぞれ方法の手順を示します。

SES コンソールを使用してマネージド IP プールを作成または変換するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。

2. 左側のナビゲーションペインで、[Dedicated IPs] (専用 IP) を選択します。
3. 新しいマネージド IP プールを作成するか、標準の専用 IP プールをマネージド IP プールに変換するかに応じて、それぞれの手順に従ってください。

Create new pool

新しいマネージド IP プールを作成するには

1. 次のいずれかを行います。
   a. アカウントに既存の専用 IP がない場合:
      - [専用 IP] オンボーディングページが表示されます。[Dedicated IPs (managed) overview] (専用 IP (マネージド) の概要) パネルで、[Enable dedicated IPs] (専用 IP を有効化) を選択します。

        [IP プールの作成] ページが開きます。
   b. アカウントに既存の専用 IP がある場合:
      i. [専用 IP] ページで [マネージド IP プール] タブを選択します。
      ii. [All Dedicated IP (managed) pools] (すべての専用 IP (マネージド) プール) パネルで、[Create Managed IP pool] (マネージド IP プールの作成) を選択します。

        [IP プールの作成] ページが開きます。
2. [Pool details] (プール詳細) パネルで、
   a. [Scaling mode] (スケーリングモード) フィールドで [Managed (auto managed)] (マネージド (自動マネージド)) を選択します。
   b. [IP pool name] (IP プール名) フィールドにマネージドプールの名前を入力します。

> **Note**
>
> - IP プールの名前は一意である必要があります。アカウントの標準専用 IP プール名と重複することはできません。
> - マネージド IP プールと標準 IP プールの両方を含めて、アカウントで AWS リージョンごとに 50 個を超える専用 IP プールを持つことはできません。

3. (オプション) このマネージド IP プールを設定セットに関連付けるには、[Configuration sets] (設定セット) フィールドのドロップダウンリストからいずれかを選択します。

> **Note**
>
> - 既に IP プールに関連付けられている設定セットを選択した場合、設定セットは選択したマネージドプールに関連付けられ、以前のプールとは関連付けられなくなります。
> - このマネージド IP プールの作成後に関連する設定セットを追加または削除するには、設定セットの [General details] (一般詳細) パネルで、[Sending IP pool] (IP プールの送信) パラメータを編集します。
> - 設定セットを作成していない場合は、「設定セット」を参照してください。

4. (オプション)1 つ以上のタグを IP プールに追加するには、タグキーとキーのオプションの値を指定します。

   a. 新しいタグを追加を選択し、キーを入力します。任意でタグに値を追加できます。最大 50 個のタグを追加できます。間違えた場合は、[Remove] (削除) を選択します。

   b. タグを追加するには、[変更の保存] を選択します。

      プールを作成した後は、マネージドプール、[Edit] (編集) の順に選択して、タグの追加、削除、または編集を行うことができます。

5. [Create pool] (プールの作成) を選択します。

> **Note**
>
> - マネージド IP プールを作成した後で、標準 IP プールに変換することはできません。
> - 専用 IP (マネージド) を使用する場合、アカウント内の送信 ID (ドメインと E メールアドレスの任意の組み合わせ) は 1 つの AWS リージョン あたり 10,000 までになります。

Convert standard to managed

標準の専用 IP プールをマネージドプールに変換するには

1. [Dedicated IPs] (専用 IP) ページで [Standard IP pools] (標準 IP プール) タブを選択します。
2. [すべての専用 IP (標準) プール] パネルで、標準プールからマネージドプールに変換する専用 IP プールのチェックボックスを選択します。
3. [マネージドプールに変換] を選択します。[マネージド IP プールに変換] ダイアログを読んで、標準の専用 IP プールをマネージド IP プールに変換する条件を理解したことを確認します。

> **Note**
>
> 専用 IP プールを標準プールからマネージドプールに変更する前に、次の点に注意してください。
>
> 1. 現在の専用 IP (標準) はすべてマネージドプールに移動されます。
> 2. 現在、送信ボリューム用にリースしている専用 IP (標準) が多すぎる場合、専用 IP (マネージド) では冗長 IP が削除されます。
> 3. 専用 IP (標準) のいずれかが他のアプリケーションの許可リストに含まれている場合は、冗長になると削除されてしまうため、マネージドプールに移動しないでください (ポイント 2 を参照してください)。
> 4. IP ごとに課金されるのではなく、マネージドプールを介して送信したボリュームに基づいて課金されるようになります。「Amazon SES の料金」を参照してください

4. 記載されている条件に同意する場合は、[確認] を選択します。標準の専用 IP プールがマネージドプールに変換されたことを確認するバナーが表示されます。

> **Note**
>
> 変換前に標準プールに関連付けていたすべての設定セットまたはタグがマネージドプールに関連付けられるため、設定セットを使用して送信する E メールをシームレスに移行できます。

イベントの公開を使用して、マネージドプールの送信パフォーマンスを追跡することもできます。詳細については、「the section called “イベント発行を使用して E メール送信をモニタリングする”」を参照してください。

Amazon SES コンソールでのマネージド IP プールの送信と容量の表示

作成したマネージド IP プールの場合、SES コンソールを使用すると、送信メトリクスや ISP の使用率と容量を示すカードや時系列グラフを使用して、E メール送信にどのように使用されているかを簡単に確認できます。

SES コンソールを使用してマネージド IP プールの送信と容量を表示するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションペインで、[Dedicated IPs] (専用 IP) を選択します。
3. [専用 IP] ページで [マネージド IP プール] タブを選択します。
4. Amazon SES コンソールと Amazon CloudWatch コンソールのどちらで送信と容量のメトリクスを表示するかに応じて、それぞれの手順に従ってください。

   Amazon SES console

   Amazon SES コンソールで送信と容量のメトリクスを表示するには

   1. [すべての専用 IP (マネージド) プール] テーブルで、[IP プール] 列に表示されているマネージド IP プールの名前を選択すると、その詳細が表示されます。

      選択した IP プールの詳細ページが開き、次のカードと時系列グラフが表示されます。

      a. カード:

         - 送信ステータス – 次の 2 つのステータスのいずれかを表示して、送信量と頻度が専用 IP を利用するのに十分であるかどうかを示します。
           - ボリュームが不十分 – 送信量が少なすぎます。
           - 専用 IP 経由の送信 – マネージドプールで 1 つ以上の専用 IP が使用されています。
         - マネージド型専用 IP 送信ボリューム – 過去 7 日間にマネージドプール内の専用 IP を介して送信された E メールの量。

- マネージド型専用 IP 送信の割合 – 過去 7 日間に、マネージドプール内の専用 IP を介して送信された E メールの割合。

b. グラフ:

- 送信量 – 過去 7 日間にマネージド専用 IP を介して送信された E メールの量を、共有 IP と比較したものです。
- 送信ボリュームの割合 – 過去 7 日間にマネージド専用 IP を介して送信された E メールの割合を、共有 IP と比較したものです。
- ISP 容量 – 最も広く利用されている上位 10 社の ISP ごとに、マネージドプール内の専用 IP を介して送信されている E メールの量と、送信中に利用可能な容量が表示されます。
  - ISP への送信数 (赤いバー) – 選択した ISP を通じて過去 24 時間に送信した E メールの量です。
  - ISP の容量 (青い線) – 選択した ISP が過去 24 時間に使用できた容量です。

2. [ISP キャパシティ] グラフで特定の ISP を絞り込むには、[ISP] リストボックスを選択して ISP を選択します。グラフは、選択した ISP のメトリクスで更新されます (ISP でフィルタリングしない場合、デフォルトで Gmail が表示されます)。

Amazon CloudWatch console

Amazon CloudWatch コンソールで送信と容量のメトリクスを表示するには

- [すべて専用 IP (マネージド) プール] 表で、[CloudWatch メトリクス] 列の [<pool_name> CloudWatch メトリクスを表示] リンクを選択すると、詳細が表示されます。

  選択した IP プールのページが CloudWatch コンソールで開き、以下のメトリクスが表示されます。

  - Send – 専用のマネージド IP と共有 IP の両方を介して送信される E メールの量です。
  - ApproximateDedicatedSendingPercentage – 専用 IP を介して配信されたトラフィックのおおよその割合を示します。
  - SentLast24Hours – 選択した ISP を通じて過去 24 時間に送信した E メールの量です (SES コンソールには [ISP への送信] というラベルが付いています)。
  - Available24HourSend – 選択した ISP が過去 24 時間において利用可能だった容量です (SES コンソールには [ISP の容量] というラベルが付いています)。

マネージド IP プールの削除と専用 IP (マネージド) のオプトアウト

マネージド IP プールを削除すると、割り当てられた IP アドレスはすべて自動的に放棄されます。マネージド IP プールが 1 つしかない場合にそれを削除するか、最後に残ったマネージド IP プールを削除した場合、専用 IP (マネージド) 機能からオプトアウトし、請求は直ちに停止されます。

SES コンソールを使用してマネージド IP プールを削除するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションペインで、[Dedicated IPs] (専用 IP) を選択します。
3. [専用 IP] ページで [マネージド IP プール] タブを選択します。
4. [All Dedicated IP (managed) pools] (すべての専用 IP (マネージド) プール) 表で、削除するマネージドプールの [IP pool] (IP プール) 名の横にあるラジオボタンを選択し、[Delete] (削除) を選択します。
5. ポップアップモーダルで、[Delete] (削除) を選択して選択を確定するか、[Cancel] (キャンセル) を選択してマネージドプールを維持できます。

> **Note**
>
> 1 つしかないマネージドプールや最後に残ったマネージドプールを削除すると、専用 IP (マネージド) 機能をオプトアウトし、請求されなくなることを知らせるポップアップモーダルが表示されます。[Delete] (削除) を選択する前に、確認フィールドに「`Disable`」と入力する必要があります。

Amazon SES での自分の IP アドレスを使用した E メールの送信

Amazon SES には、自分の IP アドレスの使用 (BYOIP = Bring Your Own IP) と呼ばれる機能があります。この機能を使用すると、Amazon SES で自分の IP アドレスを使用して E メールを送信できます。E メールの送信にすでに IP アドレス範囲を使用している場合は、この IP アドレス範囲を Amazon SES での E メール送信に使用することをリクエストできます。

> **Note**
>
> BYOIP は、手動で設定した専用 IP アドレスでのみ使用でき、専用 IP (マネージド) では使用できません。

BYOIP は、社内の E メール送信システムを使用して IP の良い評価を確立済みであるが、Amazon SES に移行したいという場合などに役立ちます。BYOIP を使用すると、IP アドレスの評価を再確立することなく、Amazon SES を通じてすぐに E メールの送信を開始できます。

要件

BYOIP を使用するには、IP アドレス範囲が以下の要件を満たしている必要があります。

- アドレス範囲は、American Registry for Internet Numbers (ARIN)、Réseaux IP Européens Network Coordination Centre (RIPE NCC)、Asia-Pacific Network Information Centre (APNIC) などの地域インターネットレジストリ (RIR) に登録している必要があります。アドレス範囲は、企業または組織法人に登録する必要があります。個人に登録することはできません。
- 署名された認可メッセージを送信することで、自分がアドレス範囲を所有していることを証明できる必要があります。
- IP アドレス範囲内のアドレスには、消去履歴が含まれている必要があります。当社は、IP アドレス範囲の評価を調査し、評価が低いか悪意のある行動に関連する IP アドレスが含まれている場合、IP アドレス範囲を拒否する権利を留保できるものとします。
- IP アドレスの範囲には、Amazon EC2 など、BYOIP のために別の AWS のサービス に取り込まれた IP アドレスの範囲を含めることはできません。

考慮事項

IP 範囲を Amazon SES に転送することを要求する前に、以下の点を考慮する必要があります。

- 指定できる最も具体的なアドレス範囲は /24 です。言い換えると、IP 範囲 203.0.113.0/24 を Amazon SES アカウントに転送した場合、203.0.113.0 から 203.0.113.255 までの合計 256 個のアドレスから送信できます。範囲全体を転送する必要があります。Amazon SES では現在、個々の IP アドレスを転送することはできません。
- 特定の IP アドレス範囲に BYOIP を使用する場合、この範囲にアクセスできるAWS リージョンは 1 つのみです。
- AWS アカウントには、リージョンあたり 5 つのアドレス範囲を持ち込むことができます。
- 自分の IP アドレスを使用する場合、共有 Amazon SES IP アドレスのプール内のアドレスは使用できません。これらの共有 IP アドレスを使用する必要がある場合は、別のAWS リージョンで Amazon SES を使用するか、新しいAWS アカウントを作成します。
- BYOIP で使用する IP アドレスごとに、月額料金が発生します。詳細については、「Amazon SES の料金」を参照してください。

Amazon SES での自分の IP アドレスの使用

迷惑なコンテンツや悪意のあるコンテンツを送信するためにシステムが悪用されないように、当社は各 BYOIP リクエストを慎重に検討する必要があります。

Amazon SES で自分の IP 範囲を使用する場合は、以下の情報を ses-byoip-request@amazon.com 宛に送信してください。

- AWS アカウント ID。
- IP 範囲を使用する AWS リージョン (ap-south-1 など)。
- ユースケースの説明。
- Amazon SES で使用する IP 範囲。
- 範囲が登録されているインターネットレジストリの名前。

リクエストをいただいてから 48 時間以内にご連絡いたします。お客様とのコミュニケーションにおいて、IP 範囲の所有権を証明するドキュメントなど、当社からお客様に追加情報をリクエストする場合があります。

Amazon SES の Virtual Deliverability Manager

E メール戦略を成功させるには、配信性能、つまり E メールが迷惑メールフォルダやジャンクメールフォルダではなく受信者の受信トレイに確実に届くようにすることが重要な要素となります。

Virtual Deliverability Manager は Amazon SES の機能の 1 つで、送信データや配信データに関するインサイトや、配信成功率と評価に悪影響を与えている問題の解決方法に関するアドバイスを提供することで、受信トレイへの配信性能や E メールのコンバージョンを高めるなど、E メールの配信性能を強化するのに役立ちます。

受信トレイの配信性能と送信者の評価が重要な理由

E メールのコンバージョン (受信者が E メールを開封した後にアクションを起こすこと) に関しては、受信トレイへの配信性能が重要な要素となります。顧客は、メッセージを受け取らなければ、エンゲージメントどころかそのメッセージを見ることすらできません。

送信の評価は、カスタマーエクスペリエンスレベルでの受信トレイへの配信性能に最も大きな影響を及ぼします。これにより、不要なメッセージが受信者に届くか、必要なメッセージが受信者のメールボックスに届く前に、迷惑メールフォルダに振り分けられたり、ブロックされたりするかどうかが決まります。

Virtual Deliverability Manager が配信性能と評価の向上にどのように役立つか

Virtual Deliverability Manager は、配信性能と評価の両方を改善するために役立ちます。そのために、アカウントの E メールプログラムの概要レベルと詳細レベルのビューの両方を表示し、問題のある領域を明らかにするために役立つダッシュボードと、E メールの配信性能と評価に悪影響を及ぼしているインフラストラクチャの問題を修正するソリューションを提供するアドバイザーを備えています。

- ダッシュボード – アカウント、ISP、送信 ID、設定セットレベルに焦点を当てた配信性能データに関するインサイトを提供します。これにより、問題のある領域や傾向をすばやく確認し、一時的な拒否 (延期) やブロックなど、配信性能に関する大きな問題に発展する前に、発生する可能性のある問題を把握できます。これらのインサイトは、E メールキャンペーンの顧客エンゲージメントとコンバージョンを向上させるのに最適な日時を計算することで、送信者の評価を改善するのにも役立ちます。
- アドバイザー – E メールの配信性能や評価に悪影響を及ぼしている設定の問題にフラグを付けることで、E メール送信を改善するための推奨事項を提供します。アドバイザーは、SPF、DMARC、または DKIM のレコードが存在しない場合や、DKIM キーが短すぎる場合など、送信ドメイン、IP スペース、認証レコードのインフラストラクチャでの問題に対する解決策を提供します。

Virtual Deliverability Manager の使用開始

Virtual Deliverability Manager の使用を開始する際には、アカウントで Virtual Deliverability Manager を有効にするステップを Amazon SES コンソールのオンボーディングウィザードで確認できます。「the section called “使用開始”」を参照してください。

トピック

Virtual Deliverability Manager の使用開始

アカウントで Virtual Deliverability Manager の使用を開始するためには、Amazon SES コンソールのオンボーディングウィザードを使用して、これを有効にする必要があります。このウィザードでは、エンゲージメントの追跡と最適な共有配信を設定します。Virtual Deliverability Manager は、エンゲージメントの追跡と最適な共有配信を使用して送信をモニタリングし、配信性能と評価を向上させるのに役立ちます。

- エンゲージメントの追跡 – ラップされたリンク内の追跡ピクセルを使用し、開封イベントやクリックイベントを通じて受信者のエンゲージメント行動をモニタリングする機能です。追跡ピクセルがトリガーされると、メッセージが開かれたときのタイムスタンプが表示され、受信者がどのリンクをクリックしたかがわかります。これをオンにすると、URL とリンクが変更され、Amazon SES エンゲージメントの追跡ラッパーが含まれます。
- 最適な共有配信 – E メール送信時に使用する最適な IP を自動的に選択し、ターゲットとする E メール受信者へのメッセージのエンドポイント配信を改善します。これは専用 IP アドレスには適用されません。

オンボーディングウィザードでは、エンゲージメントの追跡と最適な共有配信の両方がデフォルトでオンになっていますが、オフにするオプションもあります。Virtual Deliverability Manager を最大限に活用するには、両方の機能を有効にしておくことを強くお勧めします。

Amazon SES コンソールの Virtual Deliverability Manager ダッシュボードの使用開始

次の手順は、Amazon SES コンソールを使用して、Virtual Deliverability Manager の利用を開始する方法を示します。

Amazon SES コンソールを使用して Virtual Deliverability Manager の利用を開始するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左のナビゲーションペインで、[Virtual Deliverability Manager] を選択します。
3. [Virtual Deliverability Manager overview] (Virtual Deliverability Manager の概要) ページにあるいずれかの [Get started with Virtual Deliverability Manager] (Virtual Deliverability Manager の使用開始) ボタンを選択します。
4. [Select Engagement tracking] (エンゲージメントの追跡の選択) ページで、デフォルト設定をそのまま使用するか、[Turn off engagement tracking] (エンゲージメントの追跡を無効にする) を選択し、[次へ] を選択します。

   > **Note**
   >
   > エンゲージメントの追跡をオンにすると、URL とリンクが変更され、Amazon SES エンゲージメントの追跡ラッパーが含まれます。

5. [Select Optimized shared delivery] (最適な共有配信の選択) ページで、デフォルト設定をそのまま使用するか、[Turn off optimized shared delivery] (最適な共有配信を無効にする) を選択し、[次へ] を選択します。

   > **Important**
   >
   > 共有配信を最適化すると、送信者の評価を守るために、E メールの送信が先制的に遅延する可能性があります。遅延なく送信する必要がある重要なワークロードがある場合は、この設定を有効にしないことをお勧めします。代わりに、送信には設定セットを使用し、遅延を許容できる設定セットに対してのみ最適な共有配信を有効にします。

6. [Review and enable] (確認と有効化) ページで、エンゲージメントの追跡と最適な共有配信に関する選択を確認します。戻って変更する場合は [Previous] (前へ) を選択します。それ以外の場合

は、[Enable Virtual Deliverability Manager] (Virtual Deliverability Manager を有効にする) を選択します。

[Virtual Deliverability Manager settings] (Virtual Deliverability Manager の設定) ページが開きます。[サブスクリプションの概要] パネルには Virtual Deliverability Manager のステータスが表示され、[その他の設定] パネルには [Engagement tracking] (エンゲージメントの追跡) と [Optimized shared delivery] (最適な共有配信) のステータスが表示されます。

アカウントで Virtual Deliverability Manager を有効にすると、Virtual Deliverability Manager で定義されている方法を上書きすることで、設定セットでエンゲージメントの追跡と最適な共有配信を使用する方法のカスタム設定を定義できます。これにより、特定の E メールキャンペーンに合わせて E メール送信を柔軟に調整できます。例えば、マーケティング E メールではエンゲージメントの追跡と最適な共有配信を有効にし、トランザクション E メールではこれらを無効にすることができます。設定セットを作成または編集する際は、「Virtual Deliverability Manager のオプション」を参照してください。

AWS CLI を使用した Virtual Deliverability Manager の利用開始

次の例は、AWS CLI を使用して Virtual Deliverability Manager の利用を開始する方法を示しています。

AWS CLI を使用して Virtual Deliverability Manager の利用を開始するには

Amazon SES API v2 の `PutAccountVdmAttributes` オペレーションを使用して、Virtual Deliverability Manager の利用を開始できます。次の例に示すように、このオペレーションは、AWS CLI から呼び出すことができます。

- アカウントで Virtual Deliverability Manager を有効にする:

```
aws --region us-east-1 sesv2 put-account-vdm-attributes --vdm-attributes
 VdmEnabled=ENABLED
```

- 入力ファイルを使用して、エンゲージメントの追跡と最適な共有配信の両方を有効にします。

```
aws --region us-east-1 sesv2 put-account-vdm-attributes --cli-input-json file://
attributes.json
```

入力ファイルは、次のようになります。

```
{
    "VdmAttributes": {
        "VdmEnabled": "ENABLED",
        "DashboardAttributes": {
            "EngagementMetrics": "ENABLED"
        },
        "GuardianAttributes": {
            "OptimizedSharedDelivery": "ENABLED"
        }
    }
}
```

パラメータ値と関連するデータ型は、Amazon SES API v2 リファレンスの `VdmAttributes` データ型からリンクして確認できます。

> **Note**
>
> エンゲージメントの追跡をオンにすると、URL とリンクが変更され、Amazon SES エンゲージメントの追跡ラッパーが含まれます。

> **Important**
>
> 共有配信を最適化すると、送信者の評価を守るために、E メールの送信が先制的に遅延する可能性があります。遅延なく送信する必要がある重要なワークロードがある場合は、この設定を有効にしないことをお勧めします。代わりに、送信には設定セットを使用し、遅延を許容できる設定セットに対してのみ最適な共有配信を有効にします。

- 結果を検証するには:

```
aws --region us-east-1 sesv2 get-account
```

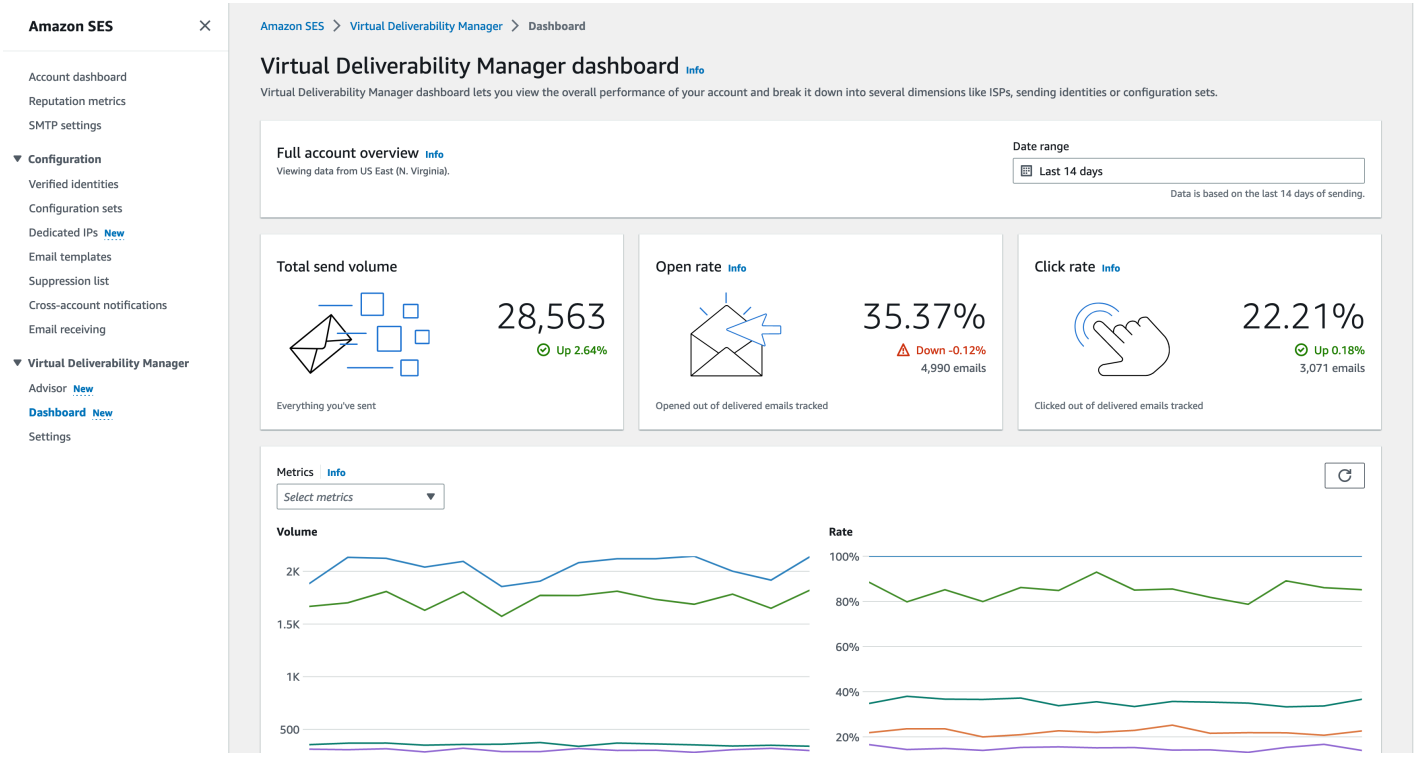
- Virtual Deliverability Manager で定義されている方法を上書きすることで、設定セットでエンゲージメントの追跡と最適な共有配信を使用する方法のカスタム設定を定義する方法については、「the section called “設定”」で AWS CLI の例を参照してください。

Virtual Deliverability Manager ダッシュボード

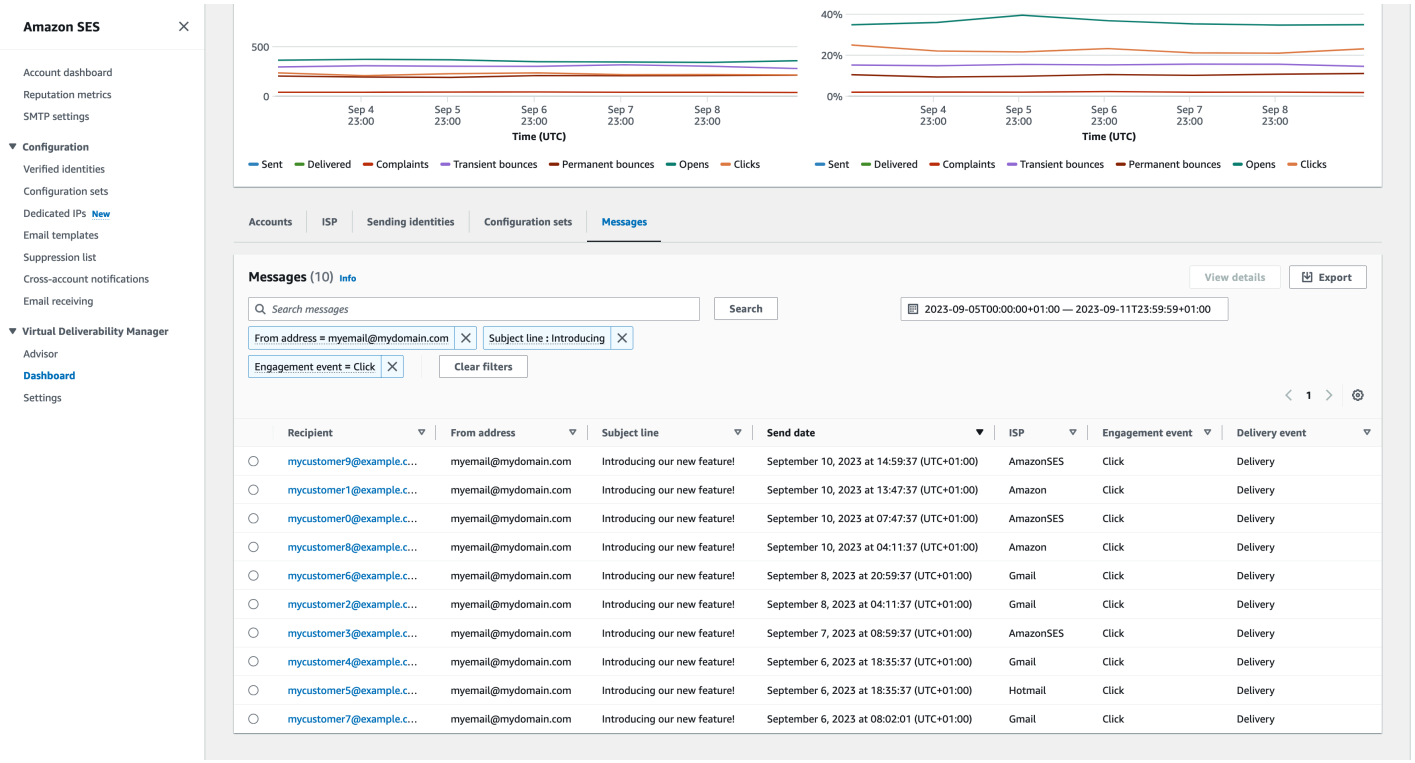
このダッシュボードには、開封/クリック、配信率、バウンス/クレーム統計を通じて配信性能や評価を読みやすいカードや時系列グラフなどで示す、アカウントの配信性能プログラムの概要が表示されます。ダッシュボードにはさらに詳細なビューが表示されるため、特定の ISP、送信 ID、E メールキャンペーンに関連する設定セットに関して問題がある場合に、より詳細な特定のテーブルデータにドリルダウンできます。

全体的な概要を確認できるほか、特定の詳細も表示できるため、E メールプログラム全体を見直すことなく、配信性能で問題のある領域に焦点を合わせることができます。また、このレベルのインサイトにより、遅延やブロックなど配信性能の大きな問題に発展する前に、傾向や潜在的な問題を把握できます。

カードと時系列グラフを示す Virtual Deliverability Manager ダッシュボードのアカウント概要



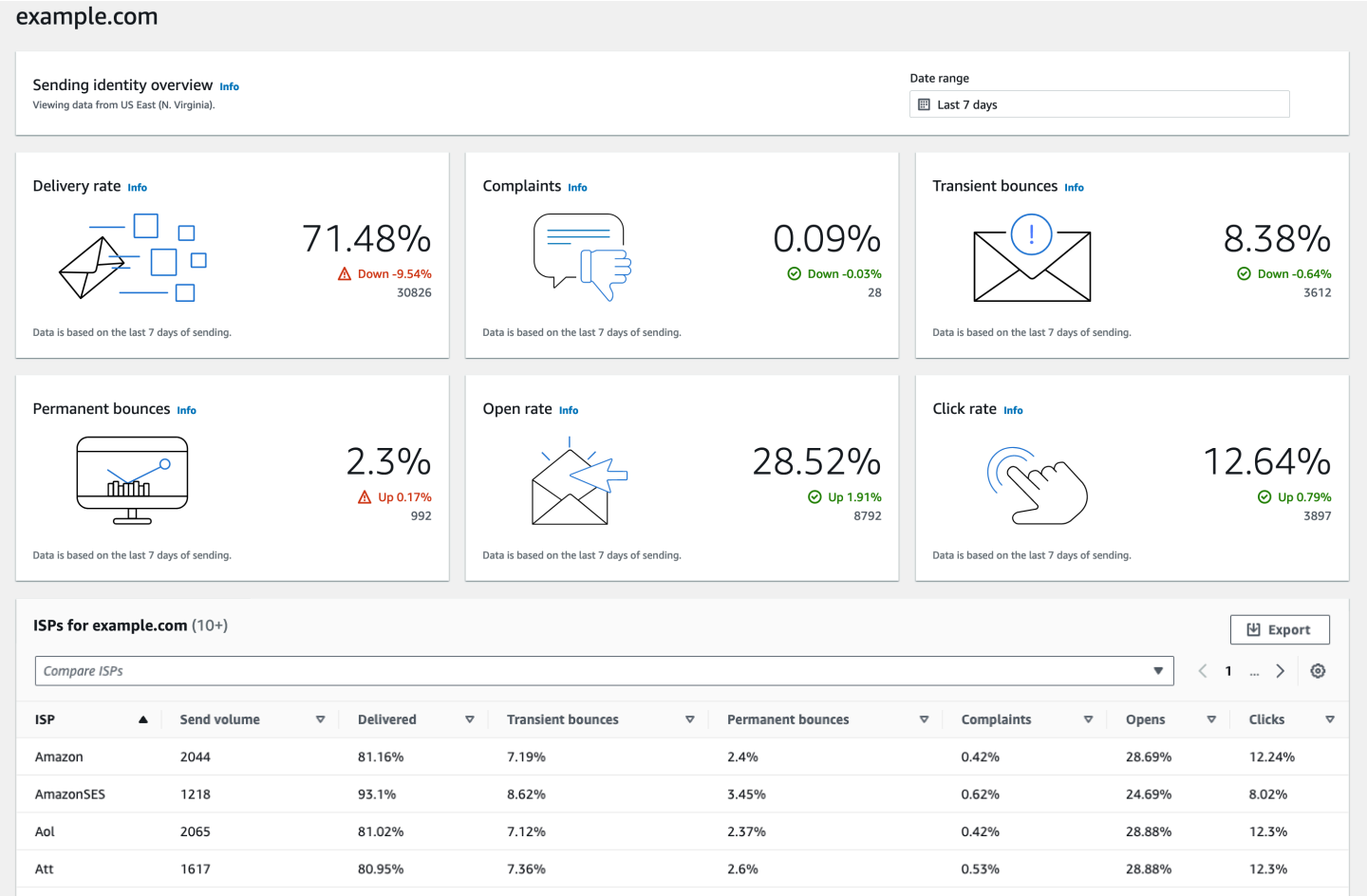
日付範囲およびフィルター条件に一致する送信済みメッセージを示す Virtual Deliverability Manager ダッシュボードで選択されたメッセージテーブル。



ダッシュボードが提供する詳細なデータは、特定のデータセットにドリルダウンできるため、送信者の評価の向上や、E メールプログラムでのエンゲージメントとコンバージョンを向上させるのに最適な日時の算出に役立ちます。

- ISP データ – 特定の ISP やメールボックスプロバイダーへの配信性能に問題がある場合に役に立ちます。それがなければ問題がない可能性があるアカウント全体の調整を試みる代わりに、問題のあるエンドポイントに集中し、そのベストプラクティスに沿って ISP に対する送信者の評価を高め、受信者に届くよう、受信トレイへの良好な配信性能を回復させることができます。また、1 つの ISP やメールボックスプロバイダーへの送信量が他の ISP またはメールボックスプロバイダーよりも多くなる場合があるため、ISP のディストリビューションを理解することも重要です。E メールのコンバージョンにプラスの影響を与えるには、トラフィックが常に最終の受信者に配信され、エンゲージメントが発生していることを確認する必要があります。
- ID および設定セットデータの送信 – アカウント配信に関する全体的な問題の原因となっている送信 ID と設定セットを特定するのに役立ちます。これらの問題に特に集中して設定を調整し、問題が解決するまで特定の ID で送信する回数を減らすことができます。例えば、送信 ID が誤ってサプレッションリストに送信され、すべてのトラフィックがその ID を通過するとします。その ID は設定セットと関連付けられているため、配信性能の問題が発生します。このような場合、送信 ID または設定セットを特定できれば、配信性能の問題の根本原因を特定するためにアカウント全体をくまなく調べるよりも、問題の修正に集中できるため有用です。

Virtual Deliverability Manager ダッシュボードに、選択した送信 ID (example.com) のドリルダウンデータが表示されます。カードには、配信性能と評価のメトリクスが表示されます。このテーブルには、送信 ID が E メールを送信したすべての ISP が、入力された日付範囲内の各 ISP のメトリクス率とともに表示されます。



Amazon SES コンソールでの Virtual Deliverability Manager ダッシュボードの使用

以下の手順は、Amazon SES コンソールで Virtual Deliverability Manager ダッシュボードを使用して、全体的な配信性能と評価の統計を表示し、問題のある領域をドリルダウンする方法を示します。

Virtual Deliverability Manager ダッシュボードを使用して、アカウントの配信性能メトリクスの概要と詳細を表示できます。

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。

2. 左側のナビゲーションペインで、[Virtual Deliverability Manager] の下の [ダッシュボード] を選択します。

> **Note**
>
> - アカウントで Virtual Deliverability Manager を有効にしていない場合、[ダッシュボード] は表示されません。詳細については、「the section called “使用開始”」を参照してください。
> - ダッシュボードメトリクスは、ほぼリアルタイムで表示されます。
> - ダッシュボードメッセージは、送信時間から数分以内に表示されます。

3. [アカウント全体の概要] パネルで、カード、時系列グラフ、ドリルダウンテーブルのすべてのメトリクスに使用する日付範囲を選択します。

- [日付範囲] フィールドで、[相対範囲] (デフォルト) または [絶対範囲] を選択します。
  - 相対範囲 – 希望の日数に対応するラジオボタンを選択します。
    - カスタム範囲 – 日 (最大 60 日)、週 (最大 8 週間)、または月 (最大 2 か月) の範囲を入力します。
  - 絶対範囲 – 最初に選択した日付が [開始日]、2 番目の日付が [終了日] となります。合計が 60 日を超えないようにしてください。1 日を指定するには、[開始日] と [終了日] の両方でその日を選択します。

> **Note**
>
> 以下はダッシュボードのすべての日付範囲に適用されます。
>
> - すべての日時は UTC です。
> - [相対範囲] の日付の場合、最終日は UTC の午前 0 時のタイムスタンプで終了します。例えば、[Last 7 days] (過去 7 日間) を選択すると、7 日目は昨日の午前 0 時に終了することになります。
> - 日付範囲が 30 日を超える場合、[アカウント統計] テーブルの [差異率] とカードの変更率には値が入力されません (ダッシュ - で示されます)。

4. カード、時系列グラフ、すべてのドリルダウンテーブル、[アカウント統計]、[ISP]、[送信 ID]、および [設定セット] では、入力した日付範囲から計算されたメトリクス合計が表示され、「ダッシュボードメトリクスの計算方法」で説明されているメトリクス計算が使用されます。

   - [ISP]、[送信 ID]、または [設定セット] テーブルのいずれかで現在表示しているデータのローカル `.csv` ファイルを作成するには、その [エクスポート]] ボタンを選択します。

5. 入力した日付範囲の [ボリューム] と [レート] の推移を示す時系列グラフが [メトリクス] ペインに表示されます。グラフの日付間隔にカーソルを合わせると、日次集計に基づく正確なボリューム数またはレートパーセンテージが表示されます。[メトリクスを選択してください] ドロップダウンを使用して、表示したいメトリクスを絞り込むことができます。

6. [アカウント] タブを選択すると、[アカウント統計] テーブルが表示されます。

   - この表には、入力した日付範囲から計算された [送信済み]、[配信済み]、[苦情]、[Transient & Permanent bounces] (一時的/永続的なバウンス)、[Opens & Clicks] (開封とクリック) の合計 [Volume] (件数)、[% Rate] (発生率)、[% Difference] (差異率) を示す、配信性能と評価のメトリクスの概要が表示されます。

   > **Note**
   >
   > 日付範囲が 30 日を超える場合、[差異率] には値が入力されません (ダッシュ - で示されます)。

7. [ISP] タブを選択して [ISP] テーブルを表示します。

   - この表には、送信先の ISP ごとに、入力した日付範囲から計算された [Send volume] (送信量)、[配信済み]、[Transient & Permanent bounces] (一時的/永続的なバウンス)、[苦情]、[Opens & Clicks] (開封とクリック) のメトリクスが表示されます。
   - 特定の ISP をフィルタリングするには、[ISP の比較] 検索ボックス内で、含める各 ISP に対応するチェックボックスをオンにします。
   - このテーブルで現在表示しているデータのローカル `.csv` ファイルを作成するには、その [エクスポート] ボタンを選択します。

8. [Sending identities] (送信 ID) タブを選択して、[Sending identities] (送信 ID) 表を表示します。

   - この表には、使用した各送信 ID ごとに、入力した日付範囲から計算された [Send volume] (送信量)、[配信済み]、[Transient & Permanent bounces] (一時的/永続的なバウンス)、[苦情]、[Opens & Clicks] (開封とクリック) のメトリクスが表示されます。

- 特定の送信 ID をフィルタリングするには、[ID の比較] 検索ボックス内で、含める各 ID に対応するチェックボックスをオンにします。
- 特定の送信 ID をドリルダウンするには、[Sending identity] (送信 ID) 列でその名前を選択します。
  - 選択された送信 ID について、入力した日付範囲から計算された [配信率]、[苦情]、[一時的/永続的なバウンス]]、[開封とクリック]] を示すカードが表示されます。
  - 時系列グラフが更新され、入力した日付範囲から計算された、選択済み送信 ID のすべてのメトリクスが表示されます。
  - 送信 ID が E メールを送信したすべての ISP と、入力された日付範囲から計算された各 ISP のメトリクスを一覧表示する ISP テーブルが表示されます。
- このテーブルで現在表示しているデータのローカル `.csv` ファイルを作成するには、その [エクスポート] ボタンを選択します。

9. [設定セット] タブを選択して [設定セット] 表を表示します。

   - この表には、E メールの送信に使用された各設定セットについて、入力した日付範囲から計算された [Send volume] (送信量)、[配信済み]、[Transient & Permanent bounces] (一時的/永続的なバウンス)、[苦情]、[Opens & Clicks] (開封とクリック) のメトリクスが表示されます。
   - 特定の設定セットをフィルタリングするには、[設定セットの比較] 検索ボックス内で、含める各設定セットに対応するチェックボックスを選択します。
   - 特定の設定セットを詳しく調べるには、[設定セット] 列でその名前を選択します。
     - 選択された設定セットについて、入力した日付範囲から計算された [配信率]、[苦情]、[一時的/永続的なバウンス]、[開封とクリック]] を示すカードが表示されます。
     - 時系列グラフが更新され、入力した日付範囲から計算された、選択済み設定セットのすべてのメトリクスが表示されます。
     - 入力された日付範囲から計算された、各 ISP で指定されたメトリクスで E メールの送信に設定セットが使用された、すべての ISP を一覧表示する ISP のテーブルが表示されます。
   - このテーブルで現在表示しているデータのローカル `.csv` ファイルを作成するには、その [エクスポート] ボタンを選択します。

10. [メッセージ] タブを選択して、[メッセージ] テーブルを表示します。

    これは、送信済みメッセージを検索して見つけることができる、インタラクティブなテーブルです。各メッセージについて、現在の配信ステータス、エンゲージメントステータス、イベント履歴を追跡したり、メールボックスプロバイダーから返されたレスポンスを確認したりできます。特定のメッセージを検索する方法のポイントは、以下のとおりです。

- 日付範囲選択ツール内で日付範囲を選択すると、過去 30 日間に送信したメッセージをフィルタリングできます。日付範囲を選択しない場合、デフォルトの検索範囲は、タイムゾーン内の現在の日付を含む過去 7 日間になります。
- [メッセージを検索] フィールドでは、[受信者]、[差出人のアドレス]、[件名]、[ISP]、[エンゲージメントイベント]、[配信イベント]、および [メッセージ ID] でフィルタリングでき、以下のプロパティが適用されます。
  - フィルタータイプに応じて、大文字と小文字を区別してテキスト文字列を入力するか、リストから値を選択します。
  - [エンゲージメントイベント] は 1 つの値に限定されます。[件名] には 1 回の検索で最大 2 つの値を指定でき、他のすべてのフィルターには最大 5 つの値を指定できます。[メッセージ ID] によるフィルタリングでは、日付範囲など、選択した他のフィルターはすべて除外されます。
  - [メッセージ ID] 列はデフォルトでは非表示になっていますが、歯車アイコンを選択して、[メッセージ] テーブルを表示する方法をカスタマイズすることで、表示できます。
- フィルターと日付範囲を選択し、[検索] を選択すると、検索条件に一致するメッセージがテーブルに入力されます。このテーブルには最大 100 件のメッセージをロードできます。検索で 100 件を超えるメッセージが返された場合、テーブル内の 100 件のメッセージは、返されたメッセージの合計から無作為に抽出したものです。
- [詳細を表示] を選択した後にメッセージのラジオボタンを選択すると、[メッセージ情報] サイドバーが表示されます。このサイドバーには、メッセージの完全なイベント履歴の詳細が新しい順に表示されるほか、メールボックスプロバイダーから返されたレスポンスまたは診断コードが表示されます。
- このテーブルで現在表示しているデータのローカル `.csv` ファイルを作成するには、その [エクスポート] ボタンを選択します。

AWS CLI を使用した Virtual Deliverability Manager のメトリクスデータへのアクセス

次の例は、AWS CLI を使用して Virtual Deliverability Manager のメトリクスデータにアクセスする方法を示しています。これは、コンソールの Virtual Deliverability Manager ダッシュボードで使用されているのと同じデータです。

AWS CLI を使用して配信性能メトリクスデータにアクセスするには

Amazon SES API v2 の BatchGetMetricData オペレーションを使用して、配信性能メトリクスデータにアクセスできます。次の例に示すように、このオペレーションは、AWS CLI から呼び出すことができます。

- 配信性能メトリクスデータにアクセスする:

```
aws --region us-east-1 sesv2 batch-get-metric-data --cli-input-json file://sends.json
```

- 入力ファイルは、次のようになります。

```
{
 "Queries": [
   {
     "Id": "Retrieve-Account-Sends",
     "Namespace": "VDM",
     "Metric": "SEND",
     "StartDate": "2022-11-04T00:00:00",
     "EndDate": "2022-11-05T00:00:00"
    }
 ]
}
```

パラメータ値と関連するデータ型に関する詳細は、Amazon SES API v2 リファレンスの BatchGetMetricDataQuery データ型からリンクして確認できます。

AWS CLI を使用した配信性能メトリクスデータのフィルタリングとエクスポート

この例は、CreateExportJob オペレーションを使用して、AWS CLI により配信性能メトリクスデータをフィルタリングし、.csv ファイルまたは.json ファイルにエクスポートする方法を示しています。これは、Virtual Deliverability Manager ダッシュボードの [ISP]、[送信 ID]、および [設定セット] の各テーブルで使用されているものと同じデータです。

AWS CLI を使用して配信性能メトリクスデータをフィルタリングし、.csv ファイルまたは.json ファイルにエクスポートするには

Amazon SES API v2 の MetricsDataSource データ型と共に CreateExportJob オペレーションを使用して、メトリクスデータをフィルタリングし、.csv ファイルまたは .json ファイルにエクスポートできます。次の例に示すように、このオペレーションは、AWS CLI から呼び出します。

- 入力ファイルを使用して配信性能メトリクスデータをフィルタリングおよびエクスポートする:

```
aws --region us-east-1 sesv2 create-export-job --cli-input-json file://metric-export-input.json
```

- この例では、入力ファイルは MetricsDataSource パラメータを使用して、E メールを送信したすべての ISP をフィルタリングし、指定した日付範囲内の配信成功率を表示します。出力ファイルには .csv 形式が指定されています。

```
{
    "ExportDataSource": {
        "MetricsDataSource": {
            "Dimensions": {
                "ISP": ["*"]
            },
            "Namespace": "VDM",
            "Metrics": [
                {
                    "Name": "DELIVERY",
                    "Aggregation": "RATE"
                }
            ],
            "StartDate": "2023-06-13T00:00:00",
            "EndDate": "2023-06-20T00:00:00"
        }
    },
    "ExportDestination": {
        "DataFormat": "CSV"
    }
}
```

パラメータ値と関連するデータ型に関する詳細は、「Amazon SES API v2 リファレンス」で、ExportDataSource 型のオブジェクトとして MetricsDataSource で確認できます。

AWS CLI の使用による、送信済みメッセージの検索、その配信ステータスおよびエンゲージメントステータスの確認、および結果のエクスポート

これらの例は、CreateExportJob オペレーションを使用して、AWS CLI により、送信した特定のメッセージを検索して見つける方法、その現在の配信ステータスやエンゲージメントステータスを確認する方法、および検索結果を .csv ファイルまたは .json ファイルにエクスポートする方法について

示しています。これは、Virtual Deliverability Manager ダッシュボードの [メッセージ] テーブルで使用されているのと同じデータです。

AWS CLI を使用した、送信済みメッセージの検索、配信ステータスとエンゲージメントステータスの確認、および結果のエクスポート

Amazon SES API v2 の MessageInsightsDataSource データ型と共に CreateExportJob オペレーションを使用してフィルターを適用し、送信した特定のメッセージを見つけたり、その配信ステータスやエンゲージメントステータスを確認したり、結果を .csv ファイルまたは .json ファイルにエクスポートしたりできます。次の例に示すように、このオペレーションは、AWS CLI から呼び出します。

> **Note**
>
> フィルタリングした検索で 10,000 件を超えるメッセージが返された場合、API の結果セット内の 10,000 件のメッセージは、返されたメッセージの合計から無作為に抽出したものです。

- 入力ファイルを使用して、送信済みメッセージを検索し、現在のステータスを確認し、結果をエクスポートする:

```
aws --region us-east-1 sesv2 create-export-job --cli-input-json file://message-insights-export-input.json
```

- この例で、入力ファイルは MessageInsightsDataSource パラメータを使用して、件名が「Sale Ends Tonight!」と等しいメッセージをフィルタリングします。出力ファイルには .csv 形式が指定されています。

```
{
    "ExportDataSource": {
        "MessageInsightsDataSource": {
            "StartDate": "2023-07-01T00:00:00",
            "EndDate": "2023-07-10T00:00:00",
            "Include": {
                "Subject": [
                    "Sale Ends Tonight!"
                ]
            }
        }
    },
```

```
    "ExportDestination": {
        "DataFormat": "CSV"
    }
}
```

- この例で、入力ファイルは MessageInsightsDataSource パラメータを使用して、件名が「Hello」で始まり、FromEmailAddress に「information」を含み、「@example.com」で終わる送信先に送信されたメッセージをフィルタリングします。出力ファイルには .json 形式が指定されています。

```
{
    "ExportDataSource": {
        "MessageInsightsDataSource": {
            "StartDate": "2023-07-01T00:00:00",
            "EndDate": "2023-07-10T00:00:00",
            "Include": {
                "Subject": [
                    "Hello*"
                ],
                "FromEmailAddress": [
                    "*information*"
                ],
                "Destination": [
                    "*@example.com"
                ]
            }
        }
    },
    "ExportDestination": {
        "DataFormat": "JSON"
    }
}
```

- この例で、入力ファイルは MessageInsightsDataSource パラメータを使用して、件名が「Hello」で始まるメッセージをフィルタリングし、FromEmailAddress が「noreply@example.com」である結果を除外します。出力ファイルには .csv 形式が指定されています。

```
{
    "ExportDataSource": {
```

```
        "MessageInsightsDataSource": {
            "StartDate": "2023-07-01T00:00:00",
            "EndDate": "2023-07-10T00:00:00",
            "Include": {
                "Subject": [
                    "Hello*"
                ]
            },
            "Exclude": {
                "FromEmailAddress": [
                    "noreply@example.com"
                ]
            }
        }
    },
    "ExportDestination": {
        "DataFormat": "CSV"
    }
}
```

- この例で、入力ファイルは MessageInsightsDataSource パラメータを使用して、件名が「Hello」で始まり、FromEmailAddress に「information」を含み、「@example.com」で終わる送信先に送信され、ISP として Gmail を使用し、最終配信イベントが「DELIVERY」で、最終エンゲージメントイベントが「OPEN」または「CLICK」であるメッセージをフィルタリングします。出力ファイルには .json 形式が指定されています。

```
{
    "ExportDataSource": {
        "MessageInsightsDataSource": {
            "StartDate": "2023-07-01T00:00:00",
            "EndDate": "2023-07-10T00:00:00",
            "Include": {
                "Subject": [
                    "Hello*"
                ],
                "FromEmailAddress": [
                    "*information*"
                ],
                "Destination": [
                    "*@example.com"
                ],
```

```
                "Isp": [
                    "Gmail"
                ],
                "LastDeliveryEvent": [
                    "DELIVERY"
                ],
                "LastEngagementEvent": [
                    "OPEN", "CLICK"
                ]
            }
        }
    },
    "ExportDestination": {
        "DataFormat": "JSON"
    }
}
```

- この例で、入力ファイルは MessageInsightsDataSource パラメータを使用して、送信先が「@example1.com」、「@example2.com」、または「@example3.com」で終わるメッセージをフィルタリングし、LastDeliveryEvent が「SEND」または「DELIVERY」と等しいメッセージを除外します。出力ファイルには .csv 形式が指定されています。

```
{
    "ExportDataSource": {
        "MessageInsightsDataSource": {
            "StartDate": "2023-07-01T00:00:00",
            "EndDate": "2023-07-10T00:00:00",
            "Include": {
                "Destination": [
                    "*@example1.com",
                    "*@example2.com",
                    "*@example3.com"
                ]
            },
            "Exclude": {
                "LastDeliveryEvent": [
                    "SEND",
                    "DELIVERY"
                ]
            }
        },
```

```
    "ExportDestination": {
        "DataFormat": "CSV"
    }
}
```

パラメータ値と関連するデータ型に関する詳細は、「Amazon SES API v2 リファレンス」で、ExportDataSource 型のオブジェクトとして MessageInsightsDataSource で確認できます。

AWS CLI を使用したエクスポートジョブの管理

これらの例は、AWS CLI を使用して、一覧表示、情報の取得、およびキャンセルにより、エクスポートジョブを管理する方法について示しています。

AWS CLI を使用してエクスポートジョブを一覧表示するには

Amazon SES API v2 の ListExportJobs オペレーションを使用して、エキスポートジョブを一覧表示できます。次の例に示すように、このオペレーションは、AWS CLI から呼び出すことができます。

- エクスポートジョブを一覧表示する:

```
aws --region us-east-1 sesv2 list-export-jobs --export-source-type=METRICS_DATA
```

```
aws --region us-east-1 sesv2 list-export-jobs --job-status=CREATED
```

```
aws --region us-east-1 sesv2 list-export-jobs --cli-input-json file://list-export-jobs-input.json
```

- 入力ファイルは、次のようになります。

```
{
  "NextToken": "",
  "PageSize": 0,
  "ExportSourceType": "METRICS_DATA",
  "JobStatus": "CREATED"
}
```

ListExportJobs オペレーションのパラメータ値に関する詳細情報については、「Amazon SES API v2 リファレンス」に記載されています。

AWS CLI を使用してエクスポートジョブに関する情報を取得するには

Amazon SES API v2 の GetExportJob オペレーションを使用して、エキスポートジョブに関する情報を取得できます。次の例に示すように、このオペレーションは、AWS CLI から呼び出すことができます。

- エクスポートジョブに関する情報を取得する:

```
aws --region us-east-1 sesv2 get-export-job --job-id=<JobId>
```

```
aws --region us-east-1 sesv2 get-export-job --cli-input-json file://get-export-job-input.json
```

- 入力ファイルは、次のようになります。

```
{
    "JobId": "e2220d6b-dce5-45f2-bf60-3287a465b732"
}
```

GetExportJob オペレーションのパラメータ値に関する詳細情報については、「Amazon SES API v2 リファレンス」に記載されています。

AWS CLI を使用してエクスポートジョブをキャンセルするには

Amazon SES API v2 の CancelExportJob オペレーションを使用して、エキスポートジョブをキャンセルできます。次の例に示すように、このオペレーションは、AWS CLI から呼び出すことができます。

- エクスポートジョブをキャンセルする:

```
aws --region us-east-1 sesv2 cancel-export-job --job-id=<JobId>
```

```
aws --region us-east-1 sesv2 cancel-export-job --cli-input-json file://cancel-export-job-input.json
```

- 入力ファイルは、次のようになります。

```
{
    "JobId": "e2220d6b-dce5-45f2-bf60-3287a465b732"
}
```

CancelExportJob オペレーションのパラメータ値に関する詳細情報については、「Amazon SES API v2 リファレンス」に記載されています。

AWS CLI を使用した、メッセージの完全なイベント履歴と ISP のレスポンスの確認

以下の例は、AWS CLI を使用して、メッセージの完全なイベント履歴の詳細と、メールボックスプロバイダーから返されたレスポンスまたは診断コードを確認する方法について示しています。これは、Virtual Deliverability Manager ダッシュボードの [メッセージ] テーブルにあるメッセージのラジオボタンを選択した後で、[メッセージ情報] サイドバーで使用されているのと同じデータです。

AWS CLI を使用して、メッセージのイベント履歴と ISP のレスポンスを確認するには

Amazon SES API v2 の GetMessageInsights オペレーションを使用して、送信済みメッセージの詳細を確認できます。次の例に示すように、このオペレーションは、AWS CLI から呼び出すことができます。

- message-id で識別される送信済み E メールに関するメッセージ詳細を確認する:

```
aws --region us-east-1 sesv2 get-message-insights --message-id
 0100010000100dd-2a19190d-99d4-0000-9f00-deb5bbf2bfbe-000001
```

GetMessageInsights オペレーションのパラメータ値に関する詳細情報については、「Amazon SES API v2 リファレンス」に記載されています。

Virtual Deliverability Manager のダッシュボードメトリクスの計算方法

Virtual Deliverability Manager ダッシュボードに表示されるすべてのレートカードとドリルダウンのテーブルでは、[アカウント全体の概要] パネルに入力された日付範囲のメトリクスが計算されます。

ダッシュボードに表示されるメトリクスレートの割合は、表の説明のように計算されます。最後の 4 列は、表示されるメトリクスの取得に使用される基本的な計算の修飾子を表しています。例えば、

[Open rate] (開封率) は、エンゲージメントの追跡を有効にして配信された HTML メッセージの開封合計数を、配信された合計数で割った値です。エンゲージメントの追跡なしで、HTML でエンコードせずに送信したメッセージは反映されません。

レート割合	計算方法	エンゲージメントの追跡が有効で、HTML 形式	少なくとも 1 つの追跡対象リンクがある	SES FBL を使用して ISP に配信される	アカウントレベルのサプレッションリストに含まれる場合は除外
Open rate	開封合計/配信合計	✗			
クリック率	クリック合計/配信合計	✗	✗		
Complaint rate	苦情合計/配信合計			✗	✗
配信率	配信合計/送信合計				
一時的なバウンス率	一時的なバウンスの合計/送信合計				✗
永久バウンス率	永久バウンスの合計/送信合計				✗
合計送信量	レート割合が表示されない (送信したものすべてが常に 100%)				

すべてのメトリクスの差分率とボリューム合計の計算方法:

- 差分率 – 特定の日付範囲における前回のメトリクス合計と比較したメトリクス合計の差。例えば、過去 7 日間が指定された日付範囲の場合、直前の 7 日間のメトリクス率が、過去 7 日間のメトリクス率になります。

- 合計送信量 差分率の計算方法は異なります。例えば、(直前の 7 日間の送信量 - 過去 7 日間の送信量)/過去 7 日間の送信量になります。
- ボリューム – 各メトリクスの合計数。

> **Note**
>
> - ドリルダウン表の [配信済み] 列には、開封率、クリック率、苦情率の計算に使用される配信済み修飾子を除いたストレート配信数が表示されます。
> - Virtual Deliverability Manager は、受信者が 1 人である E メールのメトリクスのみを追跡します。複数の受信者がいる E メールは、Virtual Deliverability Manager ダッシュボードのメトリクスにはカウントされません。
>   - このような場合、CloudWatch メトリクスには複数の受信者がいる E メールが含まれるため、Virtual Deliverability Manager のメトリクスの数は Amazon CloudWatch メトリクスの数よりも少なくなります。
> - SES メールボックスシミュレーターに送信された E メールは、Virtual Deliverability Manager ダッシュボードのいずれのメトリクスにもカウントされません。
> - 代理送信者のアカウントを介して送信された E メール (以前のクロスアカウント送信) は、Virtual Deliverability Manager ダッシュボードのどのメトリクスにもカウントされません。

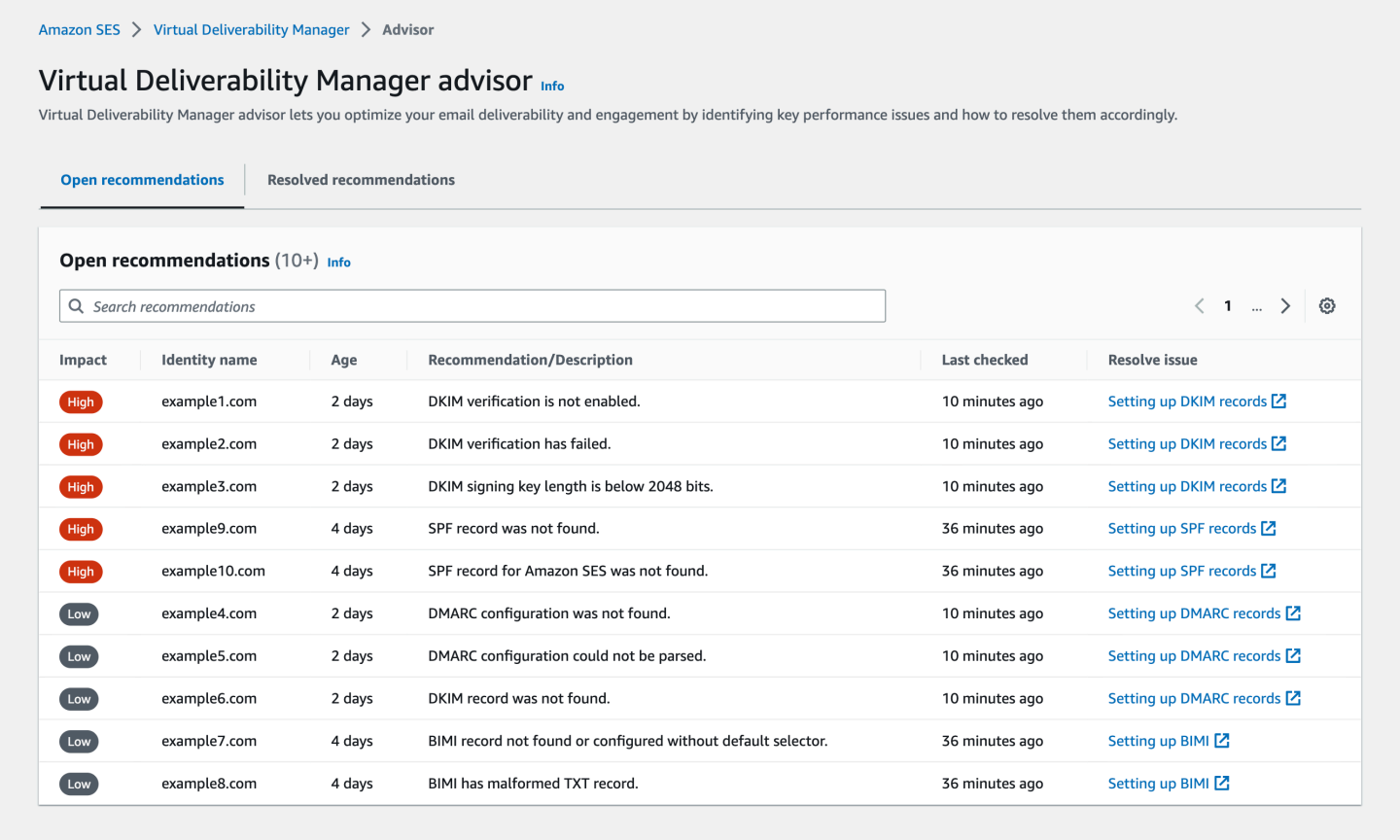
> **Important**
>
> Apple Mail のプライバシー保護とそのエンゲージメント率への影響: Apple が iOS15 から Apple デバイス向けにメールプライバシー保護 (MPP) 機能を実装した結果、必ずしも受信者がメッセージを開くかクリックしたときではなく、Apple Mail アプリケーションの起動時に MPP トリガーが開くことで、エンゲージメント数が増大しています。これにより、エンゲージメントデータが通常よりもはるかに高い数値で表示されるため、E メールマーケティング担当者はエンゲージメントを確認する際にこの点を考慮する必要があります。ウェブアクティビティ、アプリケーション/ポータルの使用状況など、エンゲージメントを特定する方法は他にもいくつかあります。また、Apple 以外のデバイスからのプロキシデータを使用して集計メトリクスを構築することもできます。注目すべき重要な点は、エンゲージメントの傾向です。これにより、E メール送信に問題があるかどうかがわかります。詳細については、「Apple Mail's Privacy Protection」を参照してください。

Virtual Deliverability Manager アドバイザー

Virtual Deliverability Manager アドバイザーは、アカウントでの主要なパフォーマンスとインフラストラクチャに関する問題を特定し、E メールの配信性能と評価に悪影響を及ぼしている ID レベルを送信することで、E メールの配信性能とエンゲージメントを最適化できるよう支援します。また、特定された問題の解決方法に関する具体的なガイダンスを通じて解決策を提供します。

アドバイザーによるインフラストラクチャの推奨事項は、[Open recommendations] (未解決の推奨事項) の表に一覧表示されます。推奨事項は、SPF、DKIM、DMARC、または BIMI レコードが存在しない場合や、形式が正しくない、キーの長さが短すぎるなどの設定に問題がある場合など、標準的な E メール認証の問題を特定します。それらは、影響の重大度、送信ドメインの ID 名、アラートの経過時間によって分類されます。検索バーのリストボックスには、影響レベル、インフラストラクチャカテゴリ、または送信 ID 名でフィルタリングするためのオプションが表示されます。「最終確認日」列には、「ちょうど今」や「15 分前」など、レコメンデーションが最後に更新された相対的な時間が表示されます。最後の列である [Resolve issue] (問題の解決) には、特定された問題の解決方法に関するガイダンスが記載された、Amazon SES デベロッパーガイドの関連セクションへのリンクが表示されます。

未解決の推奨事項は、影響レベル別にソートされて Virtual Deliverability Manager アドバイザーに表示されます。



アドバイザーからの通知がない場合は、未解決の推奨事項がないことを示すメッセージが表示されます。定期的にアドバイザーを確認することをお勧めします。オプションで、「EventBridge を使用したモニタリング」の説明に従ってこれらのアドバイザー通知イベントを Amazon EventBridge と統合して、スケーラブルなイベント駆動型アプリケーションを構築できます。

また、Virtual Deliverability Manager アドバイザーページから [Resolved recommendations] (解決済みの推奨事項) の表にアクセスすることもできます。この表には、アドバイザーのガイダンスを実装して解決したインフラストラクチャの問題が一覧表示されます。解決済みの推奨事項は、解決前の問題について説明する初期ステータスと共に一覧表示されます。解決済みの推奨事項は 30 日後に表示されなくなります。

Virtual Deliverability Manager アドバイザーのチェック内容

以前のセクションでは、Virtual Deliverability Manager のアドバイザーが送信ドメインに対してチェックを実行し、安全に認証されたインフラストラクチャの設定になっているかを判断して、高い E メール配信率を維持し、送信者の良好な評価を維持することを説明しました。Virtual Deliverability Manager アドバイザーを有効化する前に、アドバイザーがチェックする内容と、チェックで確認する内容を正確に把握しておくと役に立ちます。

この表をリファレンスとして使用して送信ドメインの設定を確認し、アドバイザーから問題として警告を受ける前に、この表に一覧表示されているような標準に合わない要素を修正することができます。

チェックのタイプ	アドバイザーのメッセージ	アドバイザーから警告を受ける理由	詳細はこちら
苦情率チェック	*ISP_name* ISP の苦情率は*#/#/#*です。	ID がこの ISP の苦情レコメンデーションのしきい値を超えています。	送信者評価のモニタリング
DKIM の設定	DKIM 検証が有効になっていません。	DKIM は ID ごとに有効にされていません。	SES での Easy DKIM
DKIM キーの強度	DKIM 署名キーの長さが 2048 ビット未満です。	DKIM 署名キーの長さで最低 2048 ビットが使用されていません。	SES での Easy DKIM
DKIM DNS レコードの検証	DKIM の検証が失敗しました。	DKIM CNAME レコードは、キーを検索して検証を試行した後、無効と判断しました。	DNS プロバイダーを使用した DKIM ドメイン ID の検証
DMARC の設定	DMARC 設定が見つかりませんでした。	DMARC TXT レコードがありません。	ドメインでの DMARC ポリシーのセットアップ
DMARC DNS レコード形式のチェック	DMARC 設定を解析できませんでした。	DMARC TXT レコードに無効な形式が見つかりました。	ドメインでの DMARC ポリシーのセットアップ
DMARC の DKIM の設定	DKIM レコードが見つかりませんでした。	DMARC に準拠するための DKIM レコード	DKIM を介した DMARC への準拠

チェックのタイプ	アドバイザーのメッセージ	アドバイザーから警告を受ける理由	詳細はこちら
		が見つかりませんでした。	
DMARC の DKIM の設定	DKIM レコードが調整されていません。	DKIM 署名で指定されたドメインと送信元アドレスのドメインとのアライメントがありません (調整されていません)。	DKIM を介した DMARC への準拠
SPF の設定	SPF レコードが見つかりませんでした。	カスタム MAIL FROM ドメインの SPF TXT レコードがありません。	カスタム MAIL FROM ドメインの設定
SPF「include」が設定されました。	Amazon SES の SPF レコードが見つかりませんでした。	`include:amazonses.com` が SPF TXT レコードにありません。	カスタム MAIL FROM ドメインの設定
SPF の実施が設定されました。	SPF のすべての修飾子が欠落しています。	`~all` が SPF TXT レコードにありません。	カスタム MAIL FROM ドメインの設定
SPF の実施の検証	SPF の設定に関する問題が検出されました。	72 時間以内に必要な SPF MX レコードを検出する試行が失敗しました。	カスタム MAIL FROM ドメイン設定のステータス
BIMI が設定されました	BIMI レコードが見つからないか、デフォルトのセレクタなしで設定されました。	BIMI TXT レコードがないか、セレクタ属性がありません。	BIMI のセットアップ

チェックのタイプ	アドバイザーのメッセージ	アドバイザーから警告を受ける理由	詳細はこちら
BIMI 形式の検証	BIMI の TXT レコード形式が正しくありません。	バージョン、証明書 URL、ロゴ URL の存在と有効な形式のチェック後、BIMI TXT レコードの設定が適切でないと判断されました。	BIMI のセットアップ

Amazon SES コンソールでの Virtual Deliverability Manager アドバイザーの使用

以下の手順では、Amazon SES コンソールで Virtual Deliverability Manager アドバイザーを使用して、Amazon SES コンソールで特定された配信性能に関する問題を解決する方法を示します。

Virtual Deliverability Manager を使用して配信性能と評価の問題を解決するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションペインで、[Virtual Deliverability Manager] の下の [Advisor] (アドバイザー) を選択します。

   > **Note**
   >
   > アカウントで Virtual Deliverability Manager を有効にしていない場合、[Advisor] (アドバイザー) は表示されません。詳細については、「the section called “使用開始”」を参照してください。

3. [Open recommendations] (未解決の推奨事項) の表がデフォルトで表示されます。推奨事項は、[影響] (高/低)、[Identity name] (ID 名) (送信ドメイン)、アラートの [Age] (経過時間)、[推奨事項/説明] (特定された問題) によって分類されます。検索バーで、[Impact] (影響) レベル、インフラストラクチャの問題の [Category] (カテゴリ)、または送信ドメインの [Identity name] (ID 名) でフィルタリングします。

4. [Recommendation/Description] (推奨事項/説明) 列に表示されている問題を解決するには、その行の [Resolve issue] (問題の解決) 列のリンクを選択し、推奨される解決策を実装します。

> **Note**
>
> 実装した解決策が反映されるまでには、最長で 6 時間かかることがあります。解決された問題は [Resolved recommendations] (解決済みの推奨事項) タブに表示されます。

AWS CLI を使用した Virtual Deliverability Manager の推奨事項へのアクセス

次の例は、AWS CLI を使用して Virtual Deliverability Manager の推奨事項にアクセスする方法を示しています。

AWS CLI を使用して Virtual Deliverability Manager の推奨事項にアクセスするには

Amazon SES API v2 の `ListRecommendations` オペレーションを使用して、配信性能に関する推奨事項を一覧表示できます。次の例に示すように、このオペレーションは、AWS CLI から呼び出すことができます。

- 推奨事項を一覧表示して配信性能についての問題を確認する

```
aws --region us-east-1 sesv2 list-recommendations
```

- フィルターを適用して、所有している特定のドメインに関する推奨事項を取得する

```
aws --region us-east-1 sesv2 list-recommendations --cli-input-json file://list-
recommendations.json
```

- 入力ファイルは、次のようになります。

```
{
  "PageSize":100,
  "Filter":{
    "RESOURCE_ARN": "arn:aws:ses:us-east-1:123456789012:identity/example.com"
   }
}
```

Virtual Deliverability Manager の設定

アカウントでの Virtual Deliverability Manager の設定は、いつでも表示または変更できます。Virtual Deliverability Manager は、有効または無効にすることができます。また、Amazon SES コンソールまたは AWS CLI から Virtual Deliverability Manager アカウントレベルで、エンゲージメントの追跡と最適な共有配信のオン/オフモードを指定できます。

Virtual Deliverability Manager のオプションは設定セットレベルでも提供されるため、Virtual Deliverability Manager で定義されている方法を上書きすることで、設定セットでエンゲージメントの追跡と最適な共有配信を使用する方法に関するカスタム設定を定義できます。これにより、特定の E メールキャンペーンに合わせて E メール送信を柔軟に調整できます。例えば、マーケティング E メールではエンゲージメントの追跡と最適な共有配信を有効にし、トランザクション E メールではこれらを無効にすることができます。

Amazon SES コンソールを使用した Virtual Deliverability Manager のアカウント設定の変更

次の手順は、Amazon SES コンソールを使用して Virtual Deliverability Manager のアカウント設定を変更する方法を示します。

Amazon SES コンソールを使用して Virtual Deliverability Manager のアカウント設定を変更するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションペインで、[Virtual Deliverability Manager] の [Settings] (設定) を選択します。

   [Virtual Deliverability Manager settings] (Virtual Deliverability Manager の設定) ページが開きます。[サブスクリプションの概要] パネルには Virtual Deliverability Manager のステータスが表示され、[その他の設定] パネルには [Engagement tracking] (エンゲージメントの追跡) と [Optimized shared delivery] (最適な共有配信) のステータスが表示されます。
3. [Engagement tracking] (エンゲージメントの追跡) または [Optimized shared delivery] (最適な共有配信) の設定を変更するには:

   a. [Additional settings] (その他の設定) パネルで、[Edit] (編集) を選択します。

   b. 対応するラジオボタンを選択して機能をオンまたはオフにし、[Submit settings] (設定を送信) を選択します。

[Virtual Deliverability Manager settings] (Virtual Deliverability Manager の設定) ページの [Additional settings] (その他の設定) パネルに変更の概要が表示されます。

> **Note**
>
> ここで定義するか、Virtual Deliverability Manager の構成セットでオーバーライドするエンゲージメント追跡オプションは、Virtual Deliverability Manager ダッシュボードで開封数とクリック数を報告するかどうかを制御します。これらは、オープンおよびクリックイベントを公開するイベント送信先設定には影響しません。例えば、ここでエンゲージメント追跡を無効にしても、ここで設定したオープンおよびクリックイベントの公開は、SES のイベント送信先で無効になりません。

4. (オプション) Virtual Deliverability Manager で定義されている方法を上書きすることで、設定セットでエンゲージメントの追跡と最適な共有配信を使用する方法のカスタム設定を定義するには、設定セットの作成または編集中に「Virtual Deliverability Manager のオプション」を参照してください。
5. Virtual Deliverability Manager を無効にするには:
   a. [Subscription overview] (サブスクリプションの概要) パネルで、[Disable Virtual Deliverability Manager] (Virtual Deliverability Manager を無効にする) を選択します。
   b. [Disable Virtual Deliverability Manager?] (Virtual Deliverability Manager を無効にしますか?) ポップアップウィンドウで、確認フィールドに「*Disable*」と入力し、[Disable Virtual Deliverability Manager] (Virtual Deliverability Manager を無効にする) を選択します。
   c. Virtual Deliverability Manager を無効にしたことを確認するバナーが表示されます。
6. Virtual Deliverability Manager を再度有効にする方法については、「the section called “使用開始”」を参照してください。

AWS CLI を使用した Virtual Deliverability Manager のアカウント設定の変更

Virtual Deliverability Manager のアカウント設定は、AWS CLI を使用して変更できます。

AWS CLI を使用して Virtual Deliverability Manager のアカウント設定を変更するには

Amazon SES API v2 の `PutAccountVdmAttributes` および `PutConfigurationSetVdmOptions` オペレーションを使用して、Virtual Deliverability Manager の

設定を変更できます。次の例に示すように、このオペレーションは、AWS CLI から呼び出すことができます。

- 入力ファイルを使用して、エンゲージメントの追跡、最適な共有配信、またはその両方を有効または無効にします。

```
aws --region us-east-1 sesv2 put-account-vdm-attributes --cli-input-json file://
attributes.json
```

この例では、エンゲージメントの追跡が ENABLED で、最適な共有配信が DISABLED です。この場合の入力ファイルは次のようになります。

```
{
    "VdmAttributes": {
        "VdmEnabled": "ENABLED",
        "DashboardAttributes": {
            "EngagementMetrics": "ENABLED"
        },
        "GuardianAttributes": {
            "OptimizedSharedDelivery": "DISABLED"
        }
    }
}
```

Amazon SES API v2 リファレンスの VdmAttributes データ型からリンクすると、パラメータ値と関連するデータ型に関する詳細を確認できます。

- Virtual Deliverability Manager で定義されている方法を上書きすることで、設定セットでエンゲージメントの追跡と最適な共有配信を使用する方法に関するカスタム設定を定義します。

```
aws --region us-east-1 sesv2 put-configuration-set-vdm-options --cli-input-json
 file://config-set.json
```

この例の example という名前の設定セットでは、エンゲージメントの追跡と最適な共有配信の両方が有効です。この場合の入力ファイルは次のようになります。

```
{
    "ConfigurationSetName": "example",
    "VdmOptions": {
        "DashboardOptions": {
```

```
            "EngagementMetrics": "ENABLED"
        },
        "GuardianOptions": {
            "OptimizedSharedDelivery": "ENABLED"
        }
    }
}
```

パラメータ値と関連するデータ型に関する詳細については、Amazon SES API v2 リファレンスの「VdmOptions のデータ型」を参照してください。

- 結果を検証するには:

```
aws –region us-east-1 sesv2 get-configuration-set –configuration-set-name example
```

- 設定セットレベルで DashboardOptions または GuardianOptions オプションを指定しないと、Virtual Deliverability Manager のアカウントレベルの設定が、その設定セットを通じて送信されるトラフィックに適用されます。

Amazon SES 向け Mail Manager

Mail Manager は、組織の E メールインフラストラクチャ強化、E メールワークフロー管理の簡素化、E メールコンプライアンス制御の合理化に役立つように設計された Amazon SES E メールゲートウェイ機能のセットです。Mail Manager は既存のインフラストラクチャと統合され、さまざまなビジネスアプリケーションを接続でき、インバウンド E メール処理を自動化します。Mail Manager は、E メールトラフィックを効率的に管理し、E メールアーカイブ機能でコンプライアンスを強化することで、E メールシステムのヘルスを維持するうえでの防御の最前線としても機能します。

Mail Manager は、既存の Amazon SES 機能に加えて、インバウンドトラフィックをサポートする以下の機能で構成されています。

- イングレスエンドポイント – 組織内で受信を許可する E メールと拒否する E メールを決定するために設定できるフィルタリングポリシーとルールを使用する、主要なインフラストラクチャコンポーネントです。
- トラフィックポリシーとルールセット – メール管理者は、定義する豊富な条件と例外に基づいて E メールを仕分け、分類し、優先度を付け、アクションを実行できる、高度にカスタマイズ可能なポリシーとルールを使用して、インバウンド E メールトラフィックを管理するためのルールを定義して適用できます。このインテリジェントなフィルタリングと自動ワークフローを組み合わせることが、E メール管理の効率化、効率の向上、組織の E メールポリシーへのコンプライアンスの確保につながります。
- SMTP リレー – 内部 E メールシステムを接続することで、ルールで定義する基準に基づいて E メールトラフィックを他の SMTP サーバーにリダイレクトし、自動転送による E メール管理を合理化します。複数のサーバーやゲートウェイにわたってトラフィックを分散できるため、ハイブリッド環境でも大量の E メールトラフィックを効果的に管理できます。
- E メールアーカイブ – 永続的で安全な長期ストレージにデータを保存することで E メールを保存して保護し、E メールを迅速に検索してアーカイブする方法を提供します。メールボックスサーバーのストレージ要件を増大させることなく、フルタイムのエンタープライズレベルのアーカイブが実現します。
- E メールアドオン – SES 認定プロバイダーからの専用のセキュリティツールのコレクションです。アドオンは、イングレスエンドポイントに送信される E メールの管理や、セキュリティ上の検出結果に基づくルーティングオプションの提供に利用できます。このツールは、既存の E メールワークフローにそのまま統合できる認定済みのセキュリティインテリジェンスおよびセキュリティ適用ソリューションであり、Mail Manager コンソールから直接有効化できます。

Mail Manager の開始方法

Mail Manager の使用を開始する際は、アカウントで Mail Manager を有効にするステップを Amazon SES コンソールのオンボーディングウィザードがガイドを提供します。「the section called “使用開始”」を参照してください。

トピック

Mail Manager の開始方法

Amazon SES Mail Manager の使用を開始するには、Amazon SES コンソールで Mail Manager の使用を開始ウィザードを使用します。ウィザードでは、イングレスエンドポイントを作成し、トラフィックポリシーとルールセットを使用して設定を行います。

イングレスエンドポイントは、Mail Manager のセットアップにおける最初の構成要素です。イングレスエンドポイントは、主要なインフラストラクチャコンポーネントであり、以下を使用します。

- トラフィックポリシー – トラフィックポリシーには、ポリシーステートメントの条件が満たされた場合に特定のタイプの E メールを許可またはブロックすることで受信メールを仕分けするように定義されたポリシーステートメントが含まれています。
- ルールセット – ルールセットには、ルールの条件が満たされた場合に、許可した E メールに対してアクションを実行するように定義したルールが含まれています。

ただし、イングレスエンドポイントの作成には、トラフィックポリシーと、既に作成済みのルールセットを選択し、イングレスエンドポイントに割り当てることが含まれます。次の手順のステップでは、最初のイングレスエンドポイントを適切に設定する手順を説明します。

SES コンソールを使用した Mail Manager の開始方法

次の手順では、SES コンソールで Mail Manager の使用を開始する方法を説明します。

Amazon SES コンソールを使用した Mail Manager の開始方法

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションペインで、[Mail Manager] を選択して、[Mail Manager の概要] ページのいずれかの [Mail Manager の使用を開始] ボタンをクリックします。
3. [セットアップを開始] ページで、[トラフィックポリシーを作成] カードの [トラフィックポリシーを作成] をクリックします。

    a. [トラフィックポリシーを作成] ページでワークフローを実行します。詳細が必要な場合は、「the section called “トラフィックポリシーとポリシーステートメントの作成 (コンソール)”」を参照してください。

    b. 最初のトラフィックポリシーとポリシーステートメントを作成したら、ブラウザの [戻る] ボタンを使用して、[セットアップを開始] ページに戻るか、左側のナビゲーションパネルの [Mail Manager] の下にある [セットアップを開始] を選択します。
4. [セットアップを開始] ページで、[ルールセットの作成] カードの [ルールセットの作成] をクリックします。

    a. [ルールセットの作成] ページでワークフローを実行します。詳細が必要な場合は、「the section called “ルールセットとルールの作成 (コンソール)”」を参照してください。

    b. 最初のルールセットとルールを作成したら、ブラウザの [戻る] ボタンを使用して、[セットアップを開始] ページに戻るか、左側のナビゲーションパネルの [Mail Manager] の下にある [セットアップを開始] を選択します。
5. 最初のトラフィックポリシーとルールセットを作成したら、最初のイングレスエンドポイントを作成できます。[セットアップを開始] ページで、[イングレスエンドポイントの作成] カードの [イングレスエンドポイントの作成] をクリックします。

    - [E メールイングレスエンドポイント] ページのワークフローの一環として、先ほど作成したトラフィックポリシーとルールセットのイングレスエンドポイントへの割り当てがありま

す。詳細が必要な場合は、「the section called “イングレスエンドポイントの作成 (コンソール)”」を参照してください。

最初のイングレスエンドポイントが作成されると、Mail Manager の使用を開始して、SMTP リレーや E メールアーカイブなどのその他の機能を利用できます。独自のトラフィックポリシーとルールセットを使用して追加のイングレスエンドポイントを作成し、すべての受信 E メールの管理方法をさらにカスタマイズすることもできます。

イングレスエンドポイント

イングレスエンドポイントは、Mail Manager の主要なインフラストラクチャコンポーネントです。イングレスエンドポイントは、ユーザーが設定したポリシーとルールを使用して、どの E メールを拒否し、どの E メールを許可し、どの E メールを処理するかを決定することにより、 E メールの受信、ルーティング、管理を行います。

各イングレスエンドポイントには、ブロックまたは許可する E メールを決定するための独自のトラフィックポリシーと、許可する E メールに対してアクションを実行するための独自のルールセットがあります。したがって、複数のイングレスエンドポイントを作成することで、各エンドポイントに特定のタイプの E メールの管理とルーティングを委任できます。このレベルのきめ細かさは、ビジネスニーズに応じてカスタマイズした E メール管理システムを構築するうえで役立ちます。

イングレスエンドポイントを作成するための前提条件ワークフロー

イングレスエンドポイントの作成時には、既に作成済みのトラフィックポリシーと、ルールセットに割り当てる必要があります。したがって、イングレスエンドポイントを作成するためのワークフローは、次の順序である必要があります。

1. まず、トラフィックポリシーを作成して、ブロックまたは許可する E メールを決定します。詳細については、「the section called “トラフィックポリシーとポリシーステートメントの作成 (コンソール)”」を参照してください。
2. 次に、許可する E メールに対してアクションを実行するルールセットを作成します。詳細については、「the section called “ルールセットとルールの作成 (コンソール)”」を参照してください。
3. 最後に、イングレスエンドポイントを作成して、先ほど作成したトラフィックポリシーとルールセット、または以前に作成済みの他のトラフィックポリシーとルールセットに割り当てます。

イングレスエンドポイントを作成したら、オンプレミスの SMTP クライアントやウェブベースの DNS ドメインホストなど、種類を問わず、E メールを受信するために使用する環境で設定する必要があります。この点については、「the section called “環境の設定”」を参照してください。

イングレスエンドポイントの使用に向けた環境の設定

「A」レコードの使用

イングレスエンドポイントを作成する際、エンドポイントの「A」レコードが生成され、その値が SES コンソールのイングレスエンドポイントの概要画面に表示されます。このレコードの値の使用方法は、以下のとおり、作成したエンドポイントのタイプとユースケースによって異なります。

- オープンエンドポイント – ドメインに送信されたメールは、イングレスエンドポイントに直接解決されます。認証は必要ありません。
  - 「A」レコードの値をコピーして、オンプレミスの SMTP クライアントの SMTP 設定に直接貼り付けるか、DNS 設定のドメインの MX レコードに貼り付けます。
  - サポートされているポート: 25
  - STARTTLS のサポート: はい
- 認証済みエンドポイント – ドメインに送信されるメールは、オンプレミスの E メールサーバーなど、SMTP 認証情報を共有している承認済みの送信者から送信される必要があります。
  - 「A」レコードの値とユーザー名およびパスワードをコピーして、オンプレミスの SMTP クライアントの SMTP 設定に直接貼り付けます。
  - サポートされているポート: 25、587 (RFC 2476)
  - STARTTLS のサポート: はい

MX レコードを使用している設定の場合は、DNS プロバイダーごとにレコードを設定するための手順とインターフェイスは異なるとはいえ、DNS 設定に入力する必要がある主な情報は、次の例に一覧表示されていることに留意します。

次のとおり、ドメインの DNS 設定で MX レコードの値としてイングレスエンドポイントの「A」レコードを入力したため、recipient@marketing.example.com に送信されるメールはすべてイングレスエンドポイントに送信されます。

- ドメイン – `marketing.example.com`
- MX レコード値 – `890123abcdef.ghijk.mail-manager-smtp.amazonaws.com` (これは、イングレスエンドポイントからコピーした「A」レコードの値です)。
- 優先度 – `10`

認証済みのエンドポイントへの接続

認証済みのエンドポイントに接続するために SMTP 認証情報を共有した承認済みの送信者の場合、サーバーへの接続を正常に確立するには、ユーザー名とパスワードについて次のプロトコルに準拠する必要があります。

- ユーザー名 – これはイングレスエンドポイント ID であり、Base64 でエンコードする必要があります。(イングレスエンドポイント ID を検索する方法については、「コンソール手順」の「ステップ 10」を参照してください)。
- パスワード – これはイングレスエンドポイントの作成時に使用したパスワードであり、Base64 でエンコードする必要があります。

次の例では、接続を確立する一般的な SMTP AUTH サーバーとクライアントの交信を説明しています。

```
S: 250 AUTH LOGIN PLAIN
C: AUTH LOGIN
S: 334 VXNlcm5hbWU6
C: SW5ncmVzc1BvaW50
S: 334 UGFzc3dvcmQ6
C: SW5ncmVzc1Bhc3N3b3Jk
S: 235 Authentication successful
```

この例は、次のプロパティで構成されています。

- S は「サーバー」を意味し、メッセージを受け入れる SMTP サーバーを指します。
- C は、「クライアント」を意味します。SMTP クライアントはサーバーとの接続を確立し、サーバーにメッセージを送信します。
- `250 AUTH LOGIN PLAIN` は、`AUTH LOGIN` または `AUTH PLAIN` のいずれかの AUTH 方式がサポートされているサーバーからの応答です。送信者はこのいずれかを選択して、認証仕様 RFC 2554 の SMTP サービス拡張に準拠した SMTP コマンドを送信できます。ここでは、`AUTH LOGIN` を使用しています。
- `334 VXNlcm5hbWU6` – Base64 でユーザー名を要求するサーバーです。
- `SW5ncmVzc1BvaW50` – Base64 のイングレスエンドポイント ID で応答するクライアントです。
- `334 UGFzc3dvcmQ6` – Base64 でパスワードを要求するサーバーです。
- `SW5ncmVzc1Bhc3N3b3Jk` – Base64 のイングレスエンドポイントパスワードで応答するクライアントです。

次のセクションの手順では、SES コンソールでイングレスエンドポイントを作成する方法について説明します。

SES コンソールでのイングレスエンドポイントの作成

次の手順では、SES コンソールの [イングレスエンドポイント] ページを使用して、イングレスエンドポイントを作成および管理する方法を説明します。

コンソールを使用してイングレスエンドポイントを作成して管理するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションパネルで、[Mail Manager] の下にある [イングレスエンドポイント] を選択します。
3. [イングレスエンドポイント] ページで、[イングレスエンドポイントの作成] をクリックします。
4. [新しいイングレスエンドポイントの作成] ページで、イングレスエンドポイントの一意の名前を入力します。
5. エンドポイントが [オープン] か [認証済み] かを選択します。
   - [認証済み] を選択した場合は、[SMTP パスワード] を選択してパスワード (承認済みの送信者と共有される) を入力するか、[シークレット] を選択して、[シークレット ARN] からシークレットのいずれかを選択します。既に作成済みのシークレットを選択する場合、新しいシークレットを作成するための次のステップで示されているポリシーが含まれている必要があります。
   - [新規作成] をクリックすると、新しいシークレットを作成するオプションを利用できます。AWS Secrets Manager コンソールが開き、新しいキーの作成を次のとおり実行できます。

   a. [シークレットのタイプ] で、[その他のシークレットのタイプ] を選択します。
   b. [キー/値のペア] で、[キー] に `password`、[値] に実際のパスワードを入力します。

> **Note**
>
> [キー] には、`password` のみを入力する必要があります (それ以外を入力すると、認証は失敗します)。

c. [新しいキーを追加] をクリックして、[暗号化キー] で KMS カスタマーマネージドキー (CMK) を作成します。AWS KMS コンソールが開きます。

d. [カスタマーマネージドキー] ページで [キーの作成] を選択します。

e. [キーを設定] ページではデフォルト値のままにして、[次へ] をクリックします。

f. [エイリアス] にキー名を入力して (必要に応じて、説明とタグを追加できます)、[次へ] をクリックします。

g. [キー管理者] でキー管理を許可するユーザー (自分以外) またはロールを選択して、[次へ] をクリックします。

h. [キーユーザー] でキーの使用を許可するユーザー (自分以外) またはロールを選択して、[次へ] をクリックします。

i. KMS CMK ポリシー をコピーして、[キーポリシー] の JSON テキストエディタに `"statement"` レベルで貼り付け、カンマ区切りの追加ステートメントとして追加します。リージョンとアカウント番号は、現在使用するものと置き換えます。

j. [Finish] を選択します。

k. AWS Secrets Manager の [新しいシークレットを保存する] ページが開いているブラウザタブをクリックして、[暗号キー] フィールドの横にある [更新] アイコン (円形の矢印) をクリックしてから、フィールド内をクリックして、新しく作成したキーを選択します。

l. [シークレットを設定] ページの [シークレットの名前] フィールドに、名前を入力します。

m. [リソースのアクセス許可] で、[許可を編集] をクリックします。

n. シークレットのリソースポリシー をコピーして、[リソースのアクセス許可] JSON テキストエディタに貼り付け、リージョンとアカウント番号を実際の値に置き換えます。(エディタ内のコード例は、必ずすべて削除します)。

o. [保存] を選択してから、[次へ] を選択します。

p. 必要に応じてローテーションを設定して、[次へ] をクリックします。

q. 新しいシークレットを確認して、[保存] をクリックして保存します。

r. SES の [新しいイングレスエンドポイントの作成] ページが開いているブラウザタブをクリックして、[リストを更新] をクリックしてから、[シークレット ARN] で新しく作成したシークレットを選択します。

6. トラフィックポリシーを選択して、ブロックまたは許可する E メールを決定します。

7. 許可する E メールに対して実行するルールアクションを含むルールセットを選択します。

8. [イングレスエンドポイントの作成] をクリックします。

9. [全般的な詳細] では、イングレスエンドポイントの作成中に「プロビジョン中」と表示されます。「アクティブ」が表示され、[ARecord] フィールドに値が表示されるまでページを更新します。「A」レコード値をコピーして、「環境の設定」で説明されているとおり、DNS 設定または SMTP クライアントに貼り付けます。
10. コンソールの [全般的な詳細] コンテナのすぐ上には、「inp」というプレフィックスが付けられている、ラベルのない値の高い数字が表示されています (ページ上部のパンくずリストにも表示されています)。例えば、inp-1abc2de3fghi4jkl5mnop6qr と表示されています。これは、イングレスエンドポイント ID と呼ばれ、この値は、イングレスサーバーにログインするためのユーザー名として使用されます。(エンドポイントに接続するには、この値を承認済みの送信者と共有する必要があります)。
11. 既に作成済みのイングレスエンドポイントは、[イングレスエンドポイント] ページで表示して管理できます。削除すべきイングレスエンドポイントがある場合は、そのイングレスエンドポイントのラジオボタンをオンにして、[削除] をクリックします。
12. イングレスエンドポイントを編集するには、イングレスエンドポイント名を選択して概要ページを開きます。

    - エンドポイントのアクティブステータスを変更するには、[全般的な詳細] で [編集] をクリックしてから、[変更の保存] をクリックします。
    - いずれかの [ルールセット] または [トラフィックポリシー] で [編集] を選択し、[変更を保存] をクリックすると、別のルールセットまたはトラフィックポリシーを選択できます。

トラフィックポリシーとポリシーステートメント

トラフィックポリシーとは、ポリシーステートメントの条件が満たされた場合に、特定のタイプの E メールを許可またはブロックすることで受信メールを仕分けできるように、イングレスエンドポイントに割り当てるポリシーステートメントのコンテナです。トラフィックポリシーは、複数のイングレスエンドポイントで使用できます。

> **Tip**
>
> トラフィックポリシーは「フィルターセット」、ポリシーステートメントは「フィルター」として考えることができます。トラフィックポリシー (フィルターセット) には、受信メールのフィルタリングに使用するポリシー (フィルター) が含まれます。

トラフィックポリシーを作成する際に、メッセージの最大サイズ (バイト単位) を設定するオプションがあります。メッセージが設定したサイズを超えると、直ちに破棄されます。これを設定すると、「最初の合格」フィルターとして機能します。次に、ポリシーステートメントの条件の範囲外となる E メールを許可またはブロックするようにデフォルトのアクションを設定します。これをトラフィックポリシーの「包括的な」アクションと考えることができます。

ポリシーステートメントは、ステートメントの条件が満たされる場合に実行される許可アクションまたはブロックアクションを使用して作成されることがあります。条件を作成するには、E メールプロトコルと、ポリシーステートメントが受信メッセージを許可またはブロックする前に一致すべき入力値について条件演算子を選択します。各ポリシーステートメントには複数の条件を含めることができます。

トラフィックポリシーには、複数のポリシーステートメントを含めることができ、以下のとおり、E メールの評価方法の暗黙的な階層に基づいた順序で実行できます。

- メッセージの最大サイズ – このオプションのパラメータが設定されている場合、このサイズを超えるメッセージはポリシーステートメントをバイパスして直ちに破棄されます。
- ブロックするポリシーステートメント – これらのステートメントは最初に評価され、ステートメントの条件を満たすメッセージをすべてブロックします。
- 許可するポリシーステートメント – これらのステートメントは次に評価され、ステートメントの条件を満たすメッセージをすべて許可します。
- トラフィックポリシーのデフォルトアクション – ポリシーステートメントの範囲外にある残りのメッセージは、このパラメータの定義方法に基づいて許可またはブロックされます。

トラフィックポリシーは、複数のイングレスエンドポイントで使用できる独立したリソースです。一方、ポリシーステートメントは、作成されたトラフィックポリシーにのみ属します。したがって、イングレスエンドポイントに送信される E メールを評価するポリシーステートメントを作成する前に、まずトラフィックポリシーを作成するか、既存のポリシーを編集する必要があります。

次のセクションの手順では、SES コンソールでトラフィックポリシーとポリシーステートメントを作成する方法について説明します。

SES コンソールでのトラフィックポリシーとポリシーステートメントの作成

次の手順では、SES コンソールの [トラフィックポリシー] ページを使用して、トラフィックポリシーとそのポリシーステートメントを作成および管理する方法を説明します。

コンソールを使用してトラフィックポリシーとポリシーステートメントを作成して管理するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションパネルで、[Mail Manager] の下にある [トラフィックポリシー] を選択します。
3. [トラフィックポリシー] ページで、[トラフィックポリシーを作成] をクリックします。
4. [トラフィックポリシーを作成] ページで、トラフィックポリシーの一意の名前を入力します。
5. (オプション) 特定のサイズを超えるメッセージを破棄する場合は、[メッセージの最大サイズ] フィールドに値をバイト単位で入力します。
6. [デフォルトアクション] で、ポリシーステートメントの条件の範囲外にあって対処されていないメッセージを [許可] するか [拒否] (ブロック) するトラフィックポリシーを選択します。
7. [新しいポリシーステートメントを追加] をクリックして、トラフィックポリシーのステートメントを作成します。
8. ステートメントの条件が満たされた場合に実行するアクションについて [許可] するか [拒否] (ブロック) するかを選択します。
9. 入力した値の E メールプロトコルと条件演算子を選択して条件を作成します。このポリシーステートメントにさらに条件を追加する場合は、[新しい条件の追加] をクリックします。条件のプロパティおよび演算子と有効な値の詳細については、「Policy statement conditions」のリファレンスを参照してください。

   - E メールアドオンをサブスクライブしている場合は、ここでアドオンを E メールプロトコルとして選択できます。
10. さらにポリシーステートメントと条件を追加する場合は、上記のステップ 7～9 を繰り返します。
11. ポリシーステートメントとポリシーステートメントの条件の作成が完了したら、[トラフィックポリシーを作成] をクリックします。
12. 既に作成したトラフィックポリシーは、[トラフィックポリシー] ページで表示して管理できます。削除すべきトラフィックポリシーがある場合は、該当するポリシーのラジオボタンをオンにして、[削除] をクリックします。
13. トラフィックポリシーのプロパティやトラフィックポリシーのポリシーステートメントのいずれかを編集するには、その名前を選択して概要ページを開いて、[編集] をクリックします。
14. [トラフィックポリシーの詳細] では、メッセージの最大サイズとデフォルトのアクションを変更できます。

15. すべての [ポリシーステートメント] コンテナ内で、許可または拒否プロパティを変更したり、条件を編集したりできます。ポリシーステートメントと条件を削除したり、新しいポリシーステートメントと条件を追加したりすることもできます。
16. すべての編集が完了したら、[変更を保存] をクリックして、変更内容を保存します。

ポリシーステートメント条件のリファレンス

ポリシーステートメントの条件

ポリシーステートメント条件の作成に使用できるすべてのポリシーステートメントプロトコルの一覧は、以下のリファレンス表のとおりです。プロトコルの式タイプを選択すると「SES Mail Manager API Reference」のリファレンスページが開き、該当するプロトコルで使用可能なすべての演算子と有効な値が一覧表示されます。

ポリシーステートメントの条件: プロトコル、演算子、値

[プロトコル]	式タイプ
受取人アドレス	文字列式の有効な演算子と値
送信者 IP 範囲	IP 式の有効な演算子と値
TLS プロトコルバージョン	TLS プロトコル式の有効な演算子と値
Abusix Mail Intelligence (サブスクライブしている場合) Spamhaus Domain Block List (サブスクライブしている場合)	ブール式の有効な演算子と値

ルールセットとルール

ルールセットとは、イングレスエンドポイントに割り当てるルールのコンテナです。ルールセットは、イングレスエンドポイントのトラフィックポリシーで許可された E メールに対してアクションを実行できます。ルールセットは、複数のイングレスエンドポイントで使用できます。

ルールは、メッセージがルールの条件を満たす場合にルールで定義されたアクションを実行することで、受信 E メールを処理する方法をイングレスエンドポイントに指示します。各ルールには、複数

の条件とアクションを含めることができます。ルールセット内で作成したルールは、ルールセット内で指定した順序で実行されます。

ルールの条件を作成するには、E メールのプロパティと、ルールがアクションを実行する前にメッセージと一致する必要がある入力値の条件演算子を選択します。実行するアクションとアクションの実行順序を定義します。

より詳細に設定するには、ルールに条件と同様に定義する例外を含めることもできますが、ここでは、メッセージが一致すべきでない条件を定義します。条件と例外は相互に依存せずに動作します。必要に応じて例外のみを含むルールを作成したり、条件と例外を混在させたりできます。

ルールセット内でルールを定義する方法は詳細になりうるため、以下のリストで、ルールセットのコンポーネントの関係を説明します。

- ルールセットのコンテンツ:
  - ルール – ルールセット内でルールを実行する順序を定義できます。

    ルールのコンテンツ:
    - 条件 – ルールは、メッセージが条件の評価と一致し、ルールに例外がある場合に適用されます (以下を参照)。
    - 条件 – ルールは、メッセージが例外の評価と一致せず、ルールに条件がある場合に適用されます (以下を参照)。
    - アクション – すべての条件が一致し、いずれの例外にも一致せず、ルールが適用されるとアクションがトリガーされます。

      ルール内でアクションを実行する順序を定義できます。

各ルールには複数の条件、例外、アクションを含めることができ、ルールとアクションの実行順序を定義できるため、特定のビジネス要件に合わせて高度にカスタマイズされ、自動化された E メール処理ソリューションを構築できます。

ルールセットは、複数のイングレスエンドポイントで使用できる独立したリソースです。一方、ルールは、作成されたルールセットにのみ属します。したがって、イングレスエンドポイントに送信される E メールに対してアクションを実行するルールを作成する前に、まずルールセットを作成するか、既存のルールセットを編集する必要があります。

次のセクションの手順では、 SES コンソールでルールセットとそのルールを作成する方法について説明します。

SES コンソールでのルールセットとルールの作成

次の手順では、SESコンソールのルールセットページを使用してルールセットとそのルールを作成し、既に作成したルールを管理する方法を示します。

コンソールを使用してルールセットとルールを作成して管理するには

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. 左側のナビゲーションパネルで、[Mail Manager] の下にある [ルールセット] を選択します。
3. [ルールセット] ページで、[ルールセットの作成] を選択して、ルールセットの一意の名前を入力します。
4. ルールセットの概要ページで、[編集] を選択して、編集ページで、[新しいルールの作成] をクリックします。
5. [ルールの詳細] サイドバーで、ルールの一意の名前を入力します。
6. 新しい条件を追加 を選択して、メッセージが一致する必要がある条件を作成します。またはEXCEPT、 の場合は チェックボックスをオンにし、新しい例外を追加 を選択して、メッセージが一致してはならない条件を作成します。
7. 入力した値の E メールのプロトコルと条件演算子を選択して、条件または例外を作成します。このルールにさらに条件または例外を追加する場合は、[新しい条件の追加] または [新しい例外を追加] をクリックします。条件のプロパティおよび演算子と有効な値の詳細については、「ルールの条件」のリファレンスを参照してください。

   - E メールアドオンにサブスクライブしている場合は、ここでアドオンを E メールのプロパティとして選択できます。
8. [新しいアクションの追加] をクリックして、ルールの条件が一致するか、例外が一致しない場合に実行するアクションを定義します。実行するアクションを追加するには、[新しいアクションの追加] をクリックします。アクションとアクションのパラメータの詳細については、「ルールのアクションリファレンス」を参照してください。

   - 複数のアクションを作成すると、上矢印と下矢印が表示され、実行順序を設定できます。
   - S3 への書き込み、メールボックスへの配信、またはインターネットへの送信のいずれかのルールアクションを実行するには、アカウントでそれぞれのアクセス許可ポリシーを有効にする必要があります。有効にしないと、ルールアクションは失敗します。

     アクションを選択したら、[ルールの詳細]]パネルからこれらのアクションのアクセス許可ポリシーを直接適用できます。

a. ロールフィールドに新しいロールの作成 を選択し、名前を入力し、その後にロールの作成 を選択します。 IAM （ このロールのIAM信頼ポリシーはバックグラウンドで自動的に生成されます ）。

b. IAM 信頼ポリシーは自動的に生成されたため、アクションのアクセス許可ポリシーをロールに追加するだけで済みます。ロールフィールドでIAMロールを表示を選択してIAMコンソールを開きます。

c. [許可] タブで、[アクセス許可の追加]、[インラインポリシーを作成] を選択します。

d. アクセス許可の指定ページで、ポリシーエディタJSONで を選択します。

e. ルールアクションのポリシー からそれぞれのアクセス許可ポリシーをコピーし、[ポリシーエディタ] に貼り付けて、赤字のテキストのデータを独自のデータに置き換えます。(エディタ内のコード例は、必ずすべて削除します)。

f. [Next (次へ)] を選択します。

g. ポリシーの作成を選択して、IAMロールのアクセス許可ポリシーを確認して作成します。

h. Mail Manager SES のルールセットの編集ページが開いているブラウザのタブを選択し、ルールを作成するための残りのステップに進みます。

9. ルールの条件、例外、アクションの作成が完了したら、左側の [ルールセットの編集] パネルにある [ルールセットの保存] を選択して、ルールセットに保存します。

10. ルールセットにさらにルールを追加する場合は、上記のステップ 4～9 を繰り返します。

    - 複数のルールを作成すると、ルールセットの [順序の変更] 列に上矢印と下矢印が表示され、実行順序を設定できます。

11. 既に作成したルールセットは、[ルールセット] ページで表示して管理できます。削除すべきルールセットがある場合は、そのポリシーのラジオボタンをオンにして、[削除] をクリックします。

12. ルールセットを編集するには、ルールセット名を選択して概要ページを開き、概要ページで [編集] をクリックします。このページでは、ルールの実行順序を変更したり、[新しいルールの作成] をクリックしてルールを追加したり、ラジオボタンをオンにしてから [削除] をクリックして、ルールを削除したりできます。

13. ルールを編集するには、ラジオボタンをオンにします。[ルールの詳細] サイドバーの任意のコンテナで、条件や例外を編集したり、アクションを変更したり、順序を変更したりすることができます。条件、例外、アクションを削除したり、新しいものを追加したりすることもできます。

14. すべての編集が完了したら、左側の[ルールセットの編集] パネルにある [ルールセットの保存] をクリックして変更を保存します。

ルール条件とアクションのリファレンス

ルール条件

次のリファレンステーブルには、ルール条件 (または例外) を作成するために使用できるすべてのルールプロパティが、式のタイプ別に分類されて一覧表示されています。同じ式タイプを共有するルールプロパティは、同じ演算子と値も共有します。プロパティの式タイプを選択すると、SESMail Manager API リファレンスのリファレンスページに移動し、そのプロパティで使用できるすべての演算子と有効な値が一覧表示されます。

ルール条件: プロパティ、演算子、値

プロパティ	式タイプ
送信元アドレス 送信先アドレス CC アドレス MAIL FROM 受取人アドレス 件名 Helo MIME ヘッダー	文字列式の有効な演算子と値
IP 範囲	IP 式の有効な演算子と値
メッセージ最大サイズ	数値式の有効な演算子と値
DKIM SPF Trend Micro のウイルススキャン (サブスクライブ済みの場合)	判定式の有効な演算子と値
TLS	ブール式の有効な演算子と値

プロパティ	式タイプ
TLS ラップ済み 受信の既読	
DMARC ポリシー	DMARC式の有効な演算子と値

ルールのアクション

次のリファレンステーブルには、ルールの条件が満たされた場合、または例外が満たされない場合に実行できるすべてのルールアクションが一覧表示されています。アクションを選択すると、SESMail Manager APIリファレンスのアクションのリファレンスページに移動し、アクションのパラメータとその形式が一覧表示されます。このテーブルでは、Mail Manager コンソールで採用されているアクション名を使用します。API名前は若干異なる場合があります。

> **Note**
>
> 一部のAPIリファレンスでは、アクションが失敗した場合に Continue または Drop に設定できる `ActionFailurePolicy`パラメータがあります。これは、 を使用する場合にのみ適用`ActionFailurePolicy`され、コンソールを使用するAPIときは、デフォルト値の Continue に設定されています。

ルールのアクション: アクションとパラメータ

アクションとアクションのパラメータ	説明
S3 への書き込み	E メールのMIME内容を S3 バケットに書き込みます。
SMTP リレーアクション	を介して別の特定のSMTPサーバーSMTPに E メールを中継します。
アクションのアーカイブ	Amazon アーカイブに配信することで、E メールをSESアーカイブします。
ヘッダーを追加する	受信 E メールにカスタムヘッダーを追加します。

アクションとアクションのパラメータ	説明
E メール受信者の書き換え	E メールエンベロープ受信者を指定された受信者のリストに置き換えます。このアクションの条件が受信者のサブセットにのみ適用される場合は、該当する受信者のみが置き換えられます。
メールボックスに配信する	E メールを Amazon WorkMail メールボックスに配信します。
Q Business に配信する	Amazon Q Business アプリケーションに E メールを配信し、ナレッジベースに取り込みます。
インターネットに送信する	SES を使用して、E メールの受信者リストの受信者に E メールを送信します (複数可) 。
アクションをドロップ	複数の受信者がある E メールの場合、このアクションがいずれかの受信者または複数の受信者 (すべての受信者ではなく) に適用される場合、該当する受信者は、E メールの受信者リストから削除され、残りの受信者には引き続きルール処理が適用されます。このアクションがすべての受信者に適用される場合、すべての受信者が受信者リストから削除され、E メールは受信されないため、ルールの処理は停止します。

SMTP リレー

Mail Manager は、E メール環境 (Microsoft 365、Google Workspace、オンプレミスの Exchange など) とインターネットの間にデプロイされるため、Mail Manager は SMTP リレーを使用して、Mail Manager が処理する受信 E メールを E メール環境にルーティングします。Mail Manager は、エンドユーザーである受信者に送信する前に、アウトバウンド E メールを別の Exchange サーバーやサードパーティーの E メールゲートウェイなどの別の E メールインフラストラクチャにルーティングすることもできます。

SMTP リレーは、E メールインフラストラクチャの重要なコンポーネントであり、ルールセットで定義されたルールアクションで指定された場合には、サーバー間で E メールを効率的にルーティングする役割を果たします。

具体的には、SMTP リレーを使用すると、SES Mail Manager と Exchange、オンプレミス の Exchange、またはサードパーティーの E メールゲートウェイなどの外部 E メールインフラストラクチャとの間で受信 E メールをリダイレクトできます。イングレスエンドポイントへの受信 E メールは、指定された E メールを指定された SMTP リレーにルーティングするルールが処理し、SMTP リレーで定義済みの外部 E メールインフラストラクチャに渡されます。

E メールを受信すると、イングレスエンドポイントはトラフィックポリシーを使用して、ブロックまたは許可する E メールを判断します。許可された E メールは、特定のタイプの E メールに対して定義したアクションを実行するための条件ルールを適用するルールセットに渡されます。定義できるルールアクションの 1 つに、SMTPRelay アクションがあります。このアクションを選択すると、E メールは SMTP リレーで定義された外部 SMTP サーバーに渡されます。

例えば、SMTPRelay アクションを使用して、イングレスエンドポイントからオンプレミスの Microsoft Exchange Server に E メールを送信できます。特定の認証情報を使用してのみアクセスできるパブリック SMTP エンドポイントを持つように Exchange サーバーを設定します。このような SMTP リレーを作成する場合、Exchange サーバーのサーバー名、ポート、認証情報を入力し、SMTP リレーに「RelayToMyExchangeServer」などの一意の名前を付けます。次に、イングレスエンドポイントのルールセットで、「送信元アドレスに『gmail.com』が含まれている場合は、RelayToMyExchangeServer という SMTP リレーを使用して SMTPRelay アクションを実行する」というルールを作成します。

これで、gmail.com からの E メールがイングレスエンドポイントに届くと、このルールが SMTPRelay アクションをトリガーし、SMTP リレーの作成時に指定した認証情報を使用して Exchange サーバーと交信し、その E メールを Exchange サーバーに配信します。このように、gmail.com から受信した E メールを Exchange サーバーにリレーできます。

SMTP リレーは、ルールアクションで指定する前に作成しておく必要があります。次のセクションの手順では、SES コンソールで SMTP リレーを作成する方法について説明します。

SES コンソールでの SMTP リレーの作成

次の手順では、SES コンソールの [SMTP リレー] ページを使用して、SMTP リレーを作成および管理する方法を説明します。

コンソールを使用して SMTP リレーを作成して管理するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/ses/ で Amazon SES コンソールを開きます。
2. 左側のナビゲーションパネルで、[Mail Manager] の下にある [SMTP リレー] を選択します。
3. [SMTP リレー] ページで、[SMTP リレーの作成] をクリックします。
4. [SMTP リレーの作成] ページで、SMTP リレーの一意の名前を入力します。
5. インバウンド (非認証) またはアウトバウンド (認証) の SMTP リレーを設定するかどうかに応じて、それぞれの手順に従ってください。

   Inbound

   インバウンド SMTP リレーを設定するには

   1. SMTP リレーをインバウンドゲートウェイとして使用して、Mail Manager にが処理する受信 E メールを外部 E メール環境にルーティングする場合は、まず E メールのホスティング環境を設定する必要があります。各 E メールホスティングプロバイダーには独自の GUI と設定ワークフローがあるとはいえ、Mail Manager SMTP リレーなどのインバウンドゲートウェイと連携するように設定する原則は似ています。

      これを説明するために、以下のセクションでは、Google Workspaces と Microsoft Office 365 を SMTP リレーを受信ゲートウェイとして連携するように設定する方法の例を説明します。

      - Google Workspaces のセットアップ
      - Microsoft Office 365 のセットアップ

      > **Note**
      >
      > 意図する受信者の送信先ドメインが SES 検証済みドメイン ID であることを確認します。例えば、E メールを受信者 abc@example.com と support@acme.com に配信する場合、example.com ドメインと acme.com ドメインの両方が SES で検証済みである必要があります。受信者のドメインが未検証の場合、SES は E メールのパブリック SMTP サーバーへの配信を試行しません。詳細については、「the section called “ID の作成と検証”」を参照してください。

2. インバウンドゲートウェイを使用するように Google Workspaces または Microsoft Office 365 を設定したら、プロバイダに応じて以下の値を使用してパブリック SMTP サーバーのホスト名を入力します。

   - Google Workspaces: `aspmx.l.google.com`
   - Microsoft Office 365: `<your_domain>.mail.protection.outlook.com`

     ドメイン名のドットを「-」に置き換えます。例えば、ドメインが acme.com の場合、`acme-com.mail.protection.outlook.com` と入力します。

3. パブリック SMTP サーバーのポート番号 25 を入力します。
4. [認証] セクションは空白のままにします (シークレット ARN を選択したり、作成したりしません)。

Outbound

アウトバウンド SMTP リレーを設定するには

1. リレーが接続する先のパブリック SMTP サーバーのホスト名を入力します。
2. パブリック SMTP サーバーのポート番号を入力します。
3. [シークレット ARN] からいずれかのシークレットを選択して、SMTP サーバーの認証をセットアップします。既に作成済みのシークレットを選択する場合、新しいシークレットを作成するための次のステップで示されているポリシーが含まれている必要があります。

   - [新規作成] をクリックすると、新しいシークレットを作成するオプションを利用できます。 AWS Secrets Manager コンソールが開き、新しいキーの作成を次のとおり実行できます。

     a. [シークレットのタイプ] で、[その他のシークレットのタイプ] を選択します。
     b. [キー/値のペア] に、以下のキーと値を入力します。

キー	value
`username`	my_username
`password`	my_password

> **Note**
>
> 両方のキーについて、この表のとおり、`password` と `username` のみを入力する必要があります (それ以外を入力すると、認証は失敗します)。値には、それぞれ現在使用しているユーザー名とパスワードを入力します。

c. 「新しいキーを追加」を選択して、暗号化キーで KMS カスタマーマネージドキー (CMK) を作成します。 AWS KMS コンソールが開きます。

d. [カスタマーマネージドキー] ページで [キーの作成] を選択します。

e. [キーを設定] ページではデフォルト値のままにして、[次へ] をクリックします。

f. [エイリアス] にキー名を入力して (必要に応じて、説明とタグを追加できます)、[次へ] をクリックします。

g. [キー管理者] でキー管理を許可するユーザー (自分以外) またはロールを選択して、[次へ] をクリックします。

h. [キーユーザー] でキーの使用を許可するユーザー (自分以外) またはロールを選択して、[次へ] をクリックします。

i. KMS CMK ポリシー をコピーして、[キーポリシー] の JSON テキストエディタに `"statement"` レベルで貼り付け、カンマ区切りの追加ステートメントとして追加します。リージョンとアカウント番号は、現在使用するものと置き換えます。

j. [Finish] を選択してください。

k. AWS Secrets Manager 新しいシークレットの保存ページが開いているブラウザのタブを選択し、暗号化キーフィールドの横にある更新アイコン (丸い矢印) を選択して、フィールド内をクリックし、新しく作成したキーを選択します。

l. [シークレットを設定] ページの [シークレットの名前] フィールドに、名前を入力します。

m. [リソースのアクセス許可] で、[許可を編集] をクリックします。

n. シークレットのリソースポリシー をコピーして、[リソースのアクセス許可] JSON テキストエディタに貼り付け、リージョンとアカウント番号を実際の値に置き換えます。(エディタ内のコード例は、必ずすべて削除します)。

o. [保存] を選択してから、[次へ] を選択します。

p. 必要に応じてローテーションを設定して、[次へ] をクリックします。

q. 新しいシークレットを確認して、[保存] をクリックして保存します。

r. SES の [新しいイングレスエンドポイントの作成] ページが開いているブラウザタブをクリックして、[リストを更新] をクリックしてから、[シークレット ARN] で新しく作成したシークレットを選択します。

6. [SMTP リレーの作成] をクリックします。
7. 既に作成した SMTP リレーは、[SMTP リレー] ページで表示して管理できます。削除すべき SMTP リレーがある場合は、そのポリシーのラジオボタンをオンにして、[削除] をクリックします。
8. SMTP リレーを編集するには、その名前を選択します。詳細ページで、対応する [編集] または [更新] ボタンをクリックすると、リレー名、外部 SMTP サーバー名、ポート、ログイン認証情報を変更し、[変更を保存] をクリックして変更内容を保存できます。

Google Workspaces でのインバウンド (非認証) SMTP リレーのセットアップ

次のウォークスルー例では、Mail Manager のインバウンド (非認証) SMTP リレーと連携するように Google Workspaces をセットアップする方法について説明します。

前提条件

- Google 管理者コンソールへのアクセス ([Google 管理コンソール] > [アプリケーション] > [Google Workspace] > [Gmail])
- Mail Manager のセットアップに使用されるドメインの MX レコードをホストするドメインネームサーバーへのアクセス

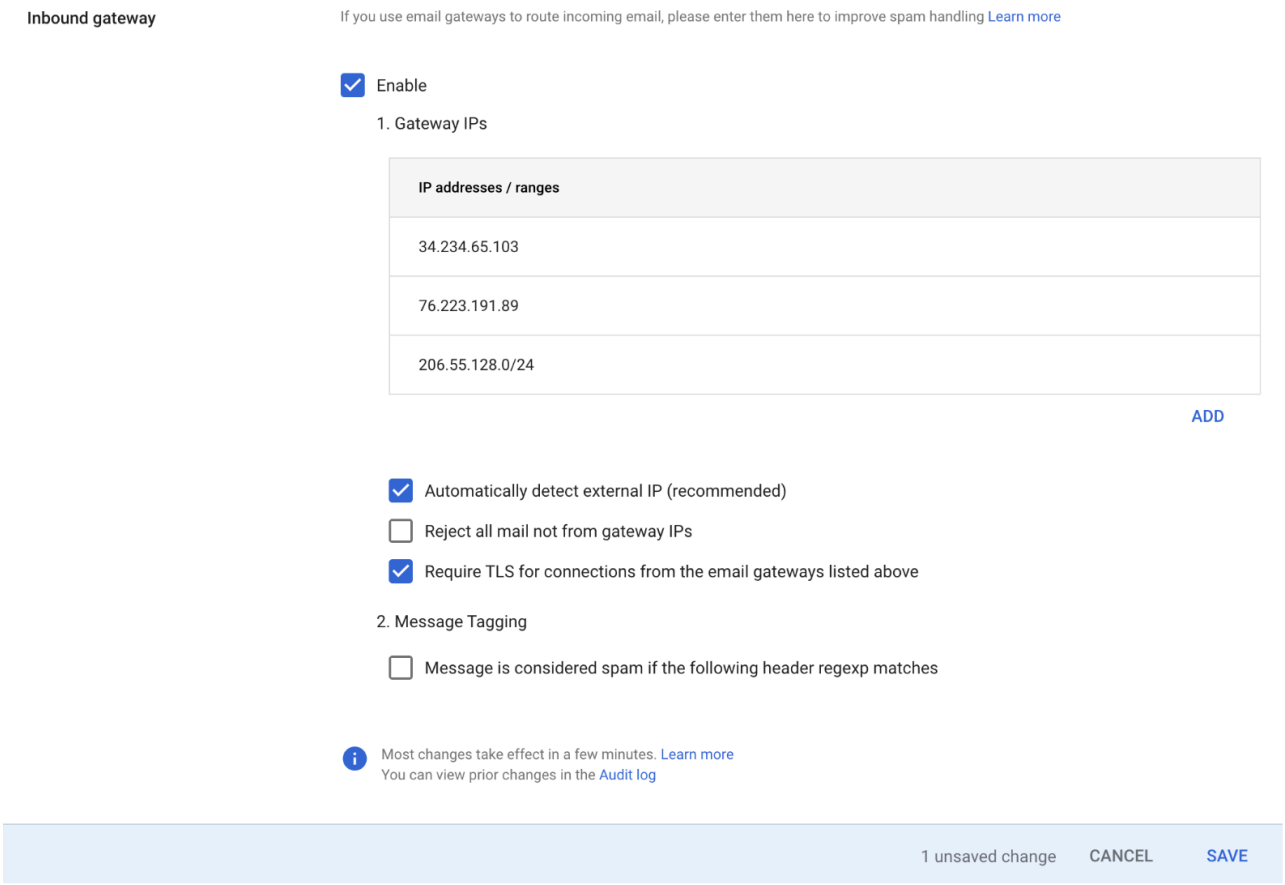
インバウンド SMTP リレーと連携するように Google Workspaces をセットアップするには

- インバウンドゲートウェイ設定に Mail Manager の IP アドレスを追加する

  a. [Google 管理コンソール] で、[アプリケーション] > [Google Workspace] > [Gmail] に移動します。

  b. [迷惑メール、フィッシング、不正なソフトウェア] を選択して、[受信ゲートウェイ] 設定に移動します。

  c. [受信ゲートウェイ] を有効にして、以下の詳細で設定します。



- ゲートウェイ IPs、 を追加 を選択し、SMTP リレー IPs 範囲 テーブルからリージョンに固有のイングレスエンドポイント IP を追加します。 https://docs.aws.amazon.com/general/latest/gr/ses.html#ses_mm_relay_ip_ranges
- [外部 IP を自動検出する] を選択します。
- [上記の電子メールゲートウェイからの接続に TLS を必須とする] を選択します。
- ダイアログボックスの下部にある [保存] を選択して、設定を保存します。保存すると、管理コンソールに [受信ゲートウェイ] が有効になっていると表示されます。

Microsoft Office 365 でのインバウンド (非認証) SMTP リレーのセットアップ

次のウォークスルー例では、Mail Manager インバウンド (非認証) SMTP リレーを使用するように Microsoft Office 365 を設定する方法を示します。

前提条件

- Microsoft セキュリティ管理センターへのアクセス ([Microsoft セキュリティ管理センター] > [メールとコラボレーション] > [ポリシーとルール] > [脅威対策ポリシー])。
- Mail Manager のセットアップに使用されるドメインの MX レコードをホストするドメインネームサーバーへのアクセス

インバウンド SMTP リレーと連携するように Microsoft Office 365 をセットアップするには

1. Mail Manager の IP アドレスを許可リストに追加する

   a. [Microsoft セキュリティ管理センター] で、[メールとコラボレーション] > [ポリシーとルール] > [脅威対策ポリシー] に移動します。

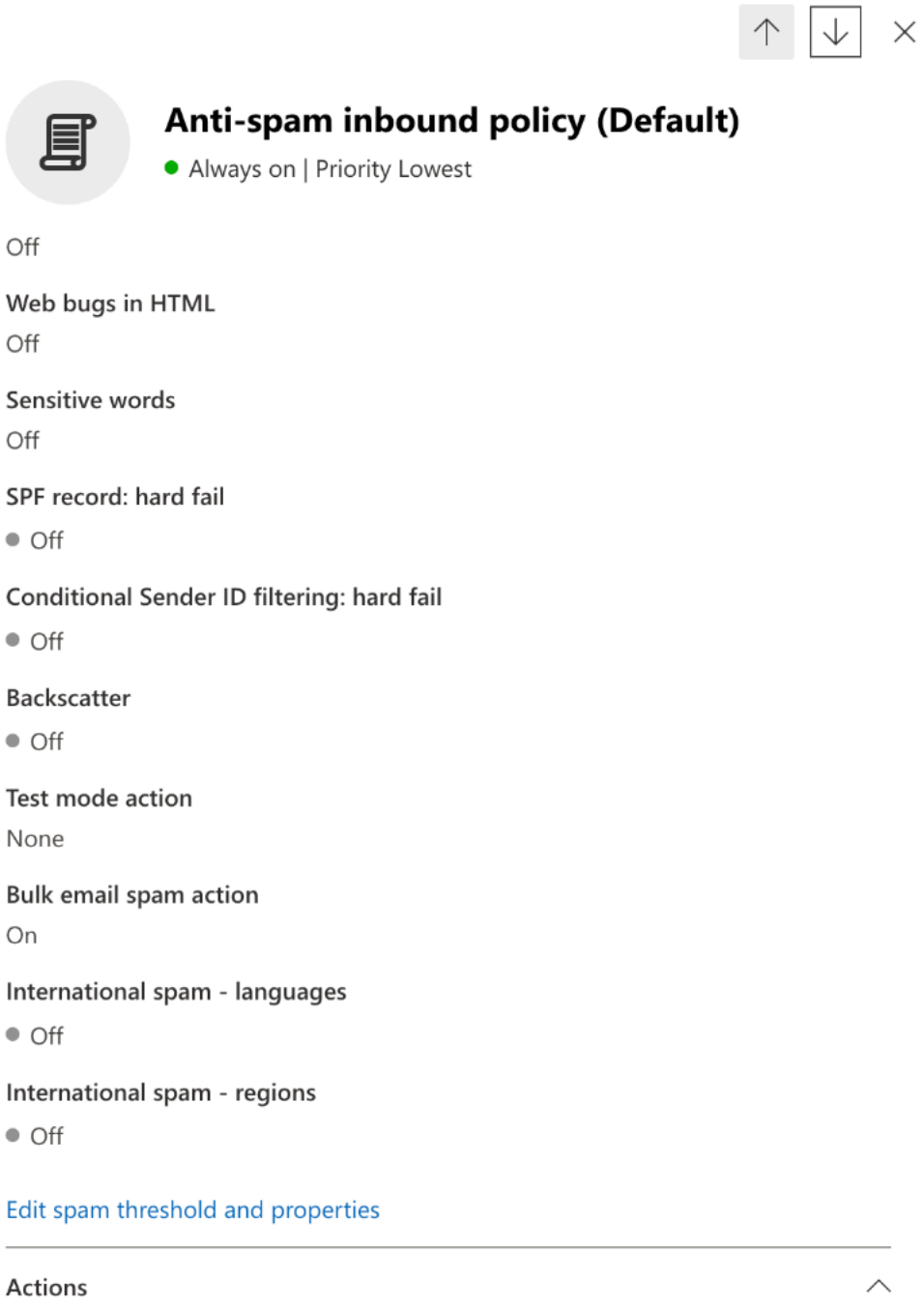
   b. [ポリシー] で [スパム対策] をクリックします。

   c. [接続フィルターポリシー] を選択して、[接続フィルターポリシーの編集] をクリックします。

      - 次の IP アドレスまたはアドレス範囲からのメッセージを常に許可するダイアログで、SMTP リレー IPs 範囲テーブルからリージョンに固有のイングレスエンドポイント IP を追加します。 https://docs.aws.amazon.com/general/latest/gr/ses.html#ses_mm_relay_ip_ranges
      - [保存] を選択します。

   d. [スパム対策] オプションに戻り、[スパム対策受信ポリシー] を選択します。

      - ダイアログの下部で、[スパムのしきい値とプロパティの編集する] をクリックします。



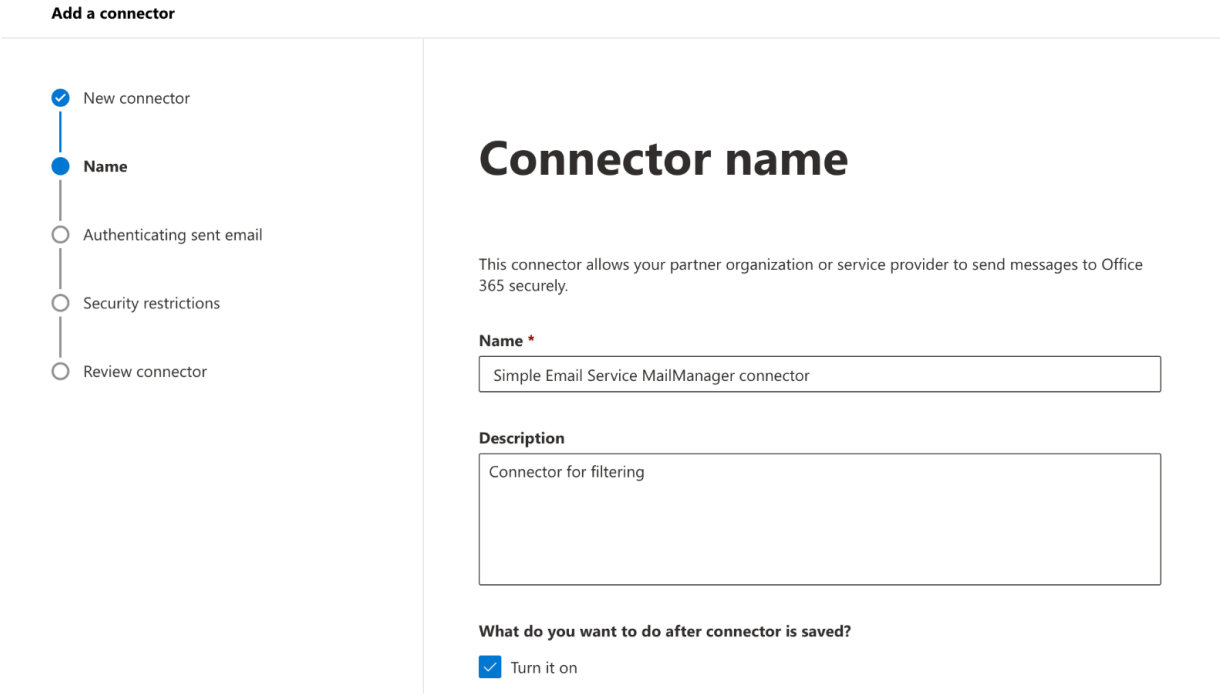
- [スパムとしてマークする] までスクロールして、[SPF レコード: ハードフェイル] が [オフ] に設定されていることを確認します。
- [保存] を選択します。

2. 拡張フィルタリング設定 (推奨)

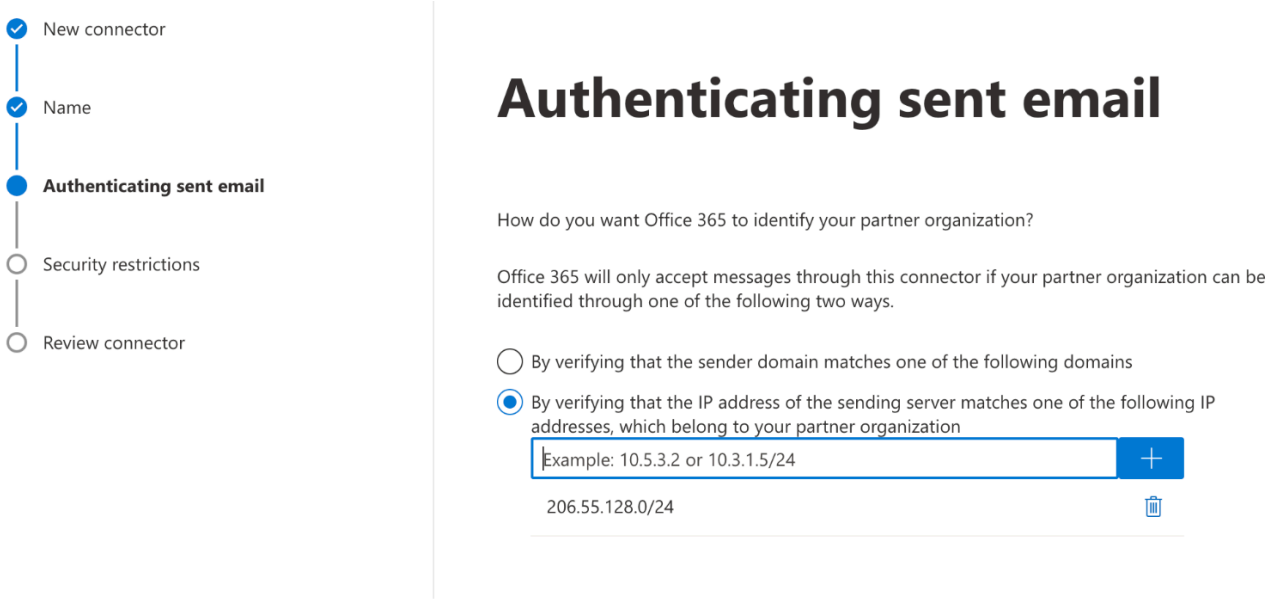
   このオプションを使用すると、Microsoft Office 365 は、SES Mail Manager がメッセージを受信する前に、元の接続 IP を適切に特定できます。

a. インバウンドコネクタを作成する

- 新たに [Exchange 管理センター] にログインして、[メールフロー] > [コネクタ] に移動します。
- [コネクタの追加] をクリックします。
- [接続元] で、[パートナー組織] を選択して、[次へ] をクリックします。
- フィールドに次のとおり入力します。
  - [名前] – Simple Email Service Mail Manager connector
  - [説明] – フィルタリング用コネクタ



- [次へ] を選択します。
- 送信済み E メールの認証で、送信サーバーの IP アドレスがパートナー組織に属する次のいずれかの IP アドレスと一致することを確認し、SMTP Relay IPs Ranges テーブルからリージョンに固有のイングレスエンドポイント IP を追加することを選択します。 https://docs.aws.amazon.com/general/latest/gr/ses.html#ses_mm_relay_ip_ranges

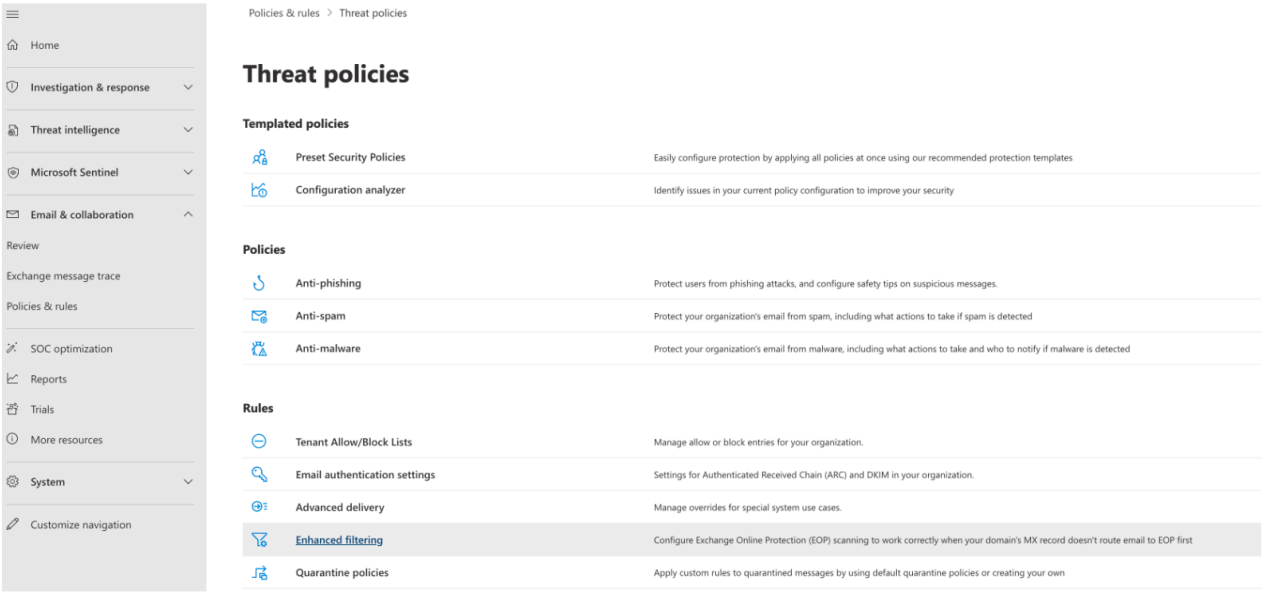


- [次へ] を選択します。
- [セキュリティ制限] では、[メール メッセージが TLS 経由で送信されない場合は拒否する] を受け入れ、[次へ] をクリックします。
- 設定を確認して、[コネクタの作成] をクリックします。

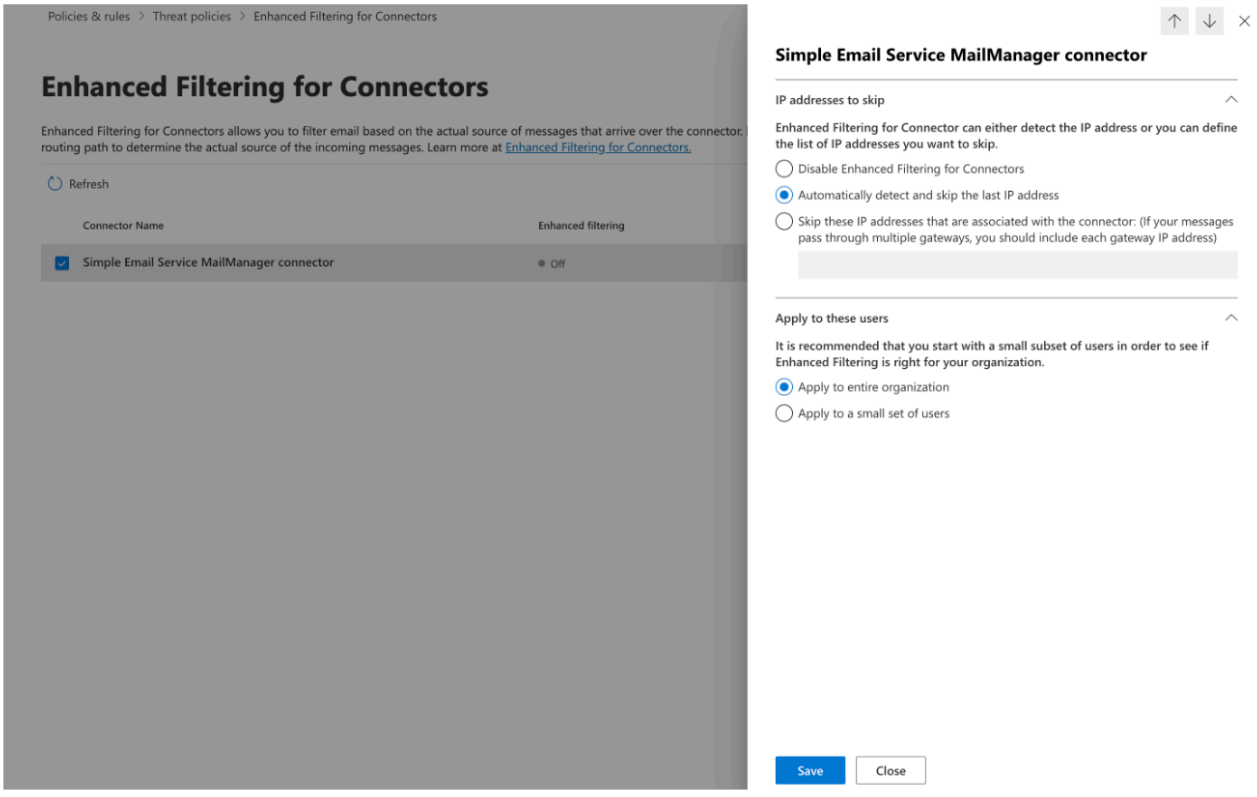
b. 拡張フィルタリングを有効にする

インバウンドコネクタの設定が完了したところで、[Microsoft セキュリティ管理センター] でコネクタの拡張フィルタリング設定を有効にする必要があります。

- [Microsoft セキュリティ管理センター] で、[メールとコラボレーション] > [ポリシーとルール] > [脅威対策ポリシー] に移動します。
- [ルール] で [拡張フィルタリング] をクリックします。



- 以前に作成した [Simple Email Service Mail Manager コネクタ] を選択して、設定パラメータを編集します。
- [最後の IP アドレスを自動的に検出してスキップする] と [organization 全体に適用する] の両方を選択します。



- [保存] を選択します。

アドレスリスト

アドレスリストは、メールマネージャーの機能です。この機能を使用すると、トラフィックポリシーとルールセットで使用できる E メールアドレスとドメインのリストを作成および管理し、メッセージの受信者または送信者が特定のリストに属しているかどうかに基づいて受信メールを処理できます。Address Lists は、E メールフローをより細かく制御し、複雑な E メールルーティングシナリオの管理を簡素化するのに役立ちます。

アドレスリストとは

アドレスリストは、E メールメッセージのフィルタリングと処理に使用できる E メールアドレスとドメインのコンテナです。関連するアドレスをグループ化し、ルーティングルールとトラフィックポリシーをまとめて適用する便利な方法を提供します。

アドレスリストの主なユースケースは次のとおりです。

- 既知のスパム送信者またはドメインをブロックするための拒否リスト
- 信頼できる送信者からの配信を確認するための許可リスト
- 存在しない受信者への E メールを早期に拒否する受信者の検証
- 受信者ロールに基づいて異なるルールを適用するためのロールベースのルーティング
- 特定のユーザーグループのポリシーを適用するためのグループベースのポリシー

アドレスリストの仕組み

SES のアドレスリストは、E メールアドレスとドメインのコレクションを作成および維持できるようにすることで、E メール管理を効率化します。作成後、これらのリストはトラフィックポリシーとルールを通じて E メールワークフローに統合されます。

SES は E メールを処理するときに、関連するアドレスリストをチェックして、送信者または受信者がメンバーであるかどうかを判断します。このメンバーシップと設定されたポリシーとルールに基づいて、SES は E メールのルーティング、フィルタリング、拒否などの適切なアクションを実行します。このプロセスにより、E メールトラフィックを効率的かつきめ細かく制御できます。

アドレスリストの設定

トピック

- アドレスリストの作成と入力

- アドレスリストの管理

アドレスリストの作成と入力

コンソールでアドレスリストを作成する一環は、1 つ以上のアドレスを入力することです。Mail Manager APIs を使用して、空のアドレスリストを作成し、後で入力できます。このセクションでは、コンソールの手順と AWS CLI 例の両方を使用して を実行する方法を示します。

アドレスリストを作成して入力するには：

1. https://console.aws.amazon.com/ses/ で SES コンソールを開きます。
2. Mail Manager のナビゲーションペインで、アドレスリストを選択します。
3. アドレスリストの作成を選択し、アドレスリスト名フィールドに名前を入力します。
4. 手動エントリまたは一括アップロードを選択し、それぞれのステップに従います。
5. 手動入力の場合 – コンソールに 1 つ以上の E メールアドレスまたはドメインを入力します。

   アスタリスク (*) ワイルドカードを使用する場合、次の形式が適用されます。

   1. アドレスには 1 つのみ*使用できます。
      - は @ の前または後*である必要があります。
      - * がローカルパートにある場合、ローカルパートは 0 文字または 3～19 文字で、 は除きます*。
      - * がドメインにある場合、サブドメインレベルは を除く 2～9 にすることができます*。
   2. 有効なワイルドカード形式の例：
      - *@domain.com
      - 123*@domain.com から 12345678901234567890*@domain.com
      - local@*.domain1.com から local@*.domain8.domain7...domain1.com
6. 一括アップロードの場合 - ファイルを選択を選択し、アップロードするアドレスを含む CSV または JSON ファイルをコンピュータから選択します。

   ファイルタイプごとに、例に示す形式を使用します。

   1. CSV ファイルの例 (ヘッダー addressは必須です)。

```
address
user1@domain.com
```

```
user2@*.domain.com
*@domain.com
```

2. JSON ファイルの例：

```
{
  "items": [
    {
      "address": "user1@domain.com"
    },
    {
      "address": "user2@*.domain.com"
    },
    {
      "address": "*@domain.com"
    }
  ]
}
```

7. アドレスの追加が完了したら、または一括ファイルを選択したら、アドレスリストの作成を選択します。

の使用 AWS CLI：

アドレスリストを作成する

```
aws mailmanager create-address-list --address-list-name "MyDenyList"
```

アドレスリストを入力します。

- シングルアップロード

```
aws mailmanager register-member-to-address-list \
                --address-list-id al-123456789abc \
                --address "user@example.com"
```

- 一括アップロード

  一括アップロードの場合は、まず CSV 形式または JSON 形式を指定するインポートジョブを作成する必要があります。

```
aws mailmanager create-address-list-import-job \
```

```
                --address-list-id "al-123456789abc" \
                --name "MyImportJob" \
                --import-data-format ImportDataType=CSV
```

これにより、ジョブ ID と署名付き URL が返されます。次の例に示すように、curl コマンドを使用して CSV または JSON ファイルを S3 バケットにアップロードするには、この署名付き URL を使用します。

```
curl -X PUT -T "/path/to/file" "pre-signed URL"
```

アップロード後、前のコマンドで返されたジョブ ID を使用してインポートジョブを開始します。

```
aws mailmanager start-address-list-import-job --job-id "job-123456789"
```

アドレスリストの管理

必要に応じて、アドレスリストを更新、表示、削除できます。

アドレスリストの更新

アドレスリストを更新するには、アドレスを追加または削除し、オプションでタグを追加または削除します。

アドレスリストを更新するには：

1. アドレスリストページで、編集するアドレスリストの名前を選択します。
2. アドレスを追加するには、「」の説明に従って E メールアドレスの追加を選択し、手動入力または一括アップロードの方法に進みますアドレスリストの作成と入力。
3. アドレスを削除するには、削除する各アドレスの横にあるチェックボックスを選択し、E メールアドレスを削除して削除を確認します。
4. （オプション) タグの管理を選択して、アドレスリストにタグを追加または削除します。

の使用 AWS CLI：

追加

```
aws mailmanager register-member-to-address-list \
                --address-list-id al-123456789abc \
```

```
                --address "user@example.com"
```

削除

```
aws mailmanager deregister-member-from-address-list \
                --address-list-id al-123456789abc \
                --address "user@example.com"
```

アドレスリストの詳細の表示

アドレスリストの詳細を表示するには：

- アドレスリストページで、アドレスリストの名前を選択して詳細を表示します。

の使用 AWS CLI：

```
aws mailmanager list-members-of-address-list --address-list-id al-123456789abc
```

アドレスリストの削除

アドレスリストを削除するには：

1. アドレスリストページで、削除するアドレスリストの横にあるラジオボタンを選択し、その後に削除を選択します。
2. 確認と削除を入力して、リストの削除を確認します。

の使用 AWS CLI：

```
aws mailmanager delete-address-list --address-list-id al-123456789abc
```

トラフィックポリシーとルールセットでのアドレスリストの使用

アドレスリストは、トラフィックポリシーステートメントとルール条件で使用して、E メールフローを制御するリストメンバーシップに基づいて E メールを処理できます。

トラフィックポリシーステートメントでのアドレスリストの使用

アドレスリストは、イングレスエンドポイントへの E メールの受信を許可または拒否するトラフィックポリシーステートメントの条件を構築するときに選択できます。

次のコンソール手順と AWS CLI 同等の手順では、受信者が指定されたアドレスリストに含まれている場合に、イングレスエンドポイントへのメッセージを許可するポリシーステートメントを作成する例を示します。

トラフィックポリシーステートメントでアドレスリストを使用するには：

1. 「」の説明に従って、新しいトラフィックポリシーを作成するか、既存のトラフィックポリシーを編集します。 トラフィックポリシーとポリシーステートメントの作成 (コンソール)
2. ポリシーステートメントコンテナで、ステートメントの条件が満たされたときにアクションの実行を許可するを選択します。
3. ステートメントの条件を次のように構築します。
   - プロトコルフィールドの受信者アドレスを選択します。
   - Operator フィールドのアドレスリストに Is を選択します。
   - Value フィールドのアドレスリストの名前を選択します。
4. これは 1 つの例にすぎませんが、任意のアドレスリストを持つさまざまな演算子に基づくポリシー条件を追加できます。

の使用 AWS CLI：

```
aws mailmanager create-traffic-policy \
  --default-action ALLOW \
  --traffic-policy-name "testpolicy" \
  --policy-statements '[{
    "Action": "ALLOW",
    "Conditions": [{
      "BooleanExpression": {
        "Evaluate": {
          "IsInAddressList": {
            "Attribute": "RECIPIENT",
            "AddressLists": [
              "arn:aws:ses:eu-west-3:123456789012:mailmanager-address-
list/al-123456789abc"
            ]
          }
        },
        "Operator": "IS_TRUE"
      }
  }]
```

```
}]'
```

ルールでのアドレスリストの使用

アドレスリストは、ルールのアクションをトリガーするためにルールセットの 1 つで使用されるルールの条件を構築するときに選択できます。

次のコンソール手順と AWS CLI 同等の手順では、受信者が指定されたアドレスリストにある場合にドロップアクションを呼び出すルールを作成する例を示します。

ルール条件でアドレスリストを使用するには：

1. 「」の説明に従って、新しいルールを作成するか、既存のルールを編集します。 ルールセットとルールの作成 (コンソール)
2. ルール条件コンテナで、ルールの条件を次のように構築します。
   - Select プロパティフィールドの受信者アドレスを選択します。
   - Select 演算子フィールドのアドレスリストに Is を選択します。
   - Value フィールドのアドレスリストの名前を選択します。
3. Actions コンテナで、新しいアクションを追加 を選択し、アクションの削除 を選択します。
4. これは 1 つの例にすぎませんが、実行するさまざまなアクションのアドレスリストを使用して、さまざまな演算子に基づくルール条件を追加できます。

の使用 AWS CLI：

```
aws mailmanager create-rule-set \
  --rule-set-name "testruleset2" \
  --rules '[{
    "Name": "addresslist",
    "Conditions": [{
      "BooleanExpression": {
        "Evaluate": {
          "IsInAddressList": {
            "Attribute": "RECIPIENT",
            "AddressLists": [
              "arn:aws:ses:us-east-1:123456789012:mailmanager-address-
list/al-123456789abc"
            ]
          }
```

```
        },
        "Operator": "IS_TRUE"
      }
    }],
    "Actions": [{
      "Drop": {}
    }]
  }]'
```

ベストプラクティスと考慮事項

- リストサイズに注意してください。リストが非常に大きいと、パフォーマンスに影響する可能性があります。
- アドレスリストはアカウント固有であり、同じ AWS アカウント内でのみ使用できます。
- ネストされたアドレスリストは現在サポートされていません。
- アドレスリストごとに最大 100,000 個のアドレスがサポートされています。

E メールアーカイブ

E メールアーカイブを使用すると、イングレスエンドポイントに送信される E メールのうち、指定したタイプの E メールをアーカイブできます。また、高度な検索フィルターの豊富なセットや、結果をエクスポートする機能を使って、アーカイブされたメッセージを検索できます。

E メールアーカイブは、永続的で安全な長期ストレージにデータを保存することで E メールを保存して保護し、E メールを迅速に検索してアーカイブする方法を提供します。メールボックスサーバーのストレージ要件を増大させることなく、フルタイムのエンタープライズレベルのアーカイブが実現します。

E メールを受信すると、イングレスエンドポイントはトラフィックポリシーを使用して、ブロックまたは許可する E メールを判断します。許可された E メールは、特定のタイプの E メールに対して定義したアクションを実行するための条件ルールを適用するルールセットに渡されます。定義できるルールアクションの 1 つに、アクションのアーカイブがあります。このアクションを選択すると、指定した E メールアーカイブに E メールがアーカイブ保存されます。

ルールアクションで指定する前に、まずアーカイブを作成する必要があります。次のセクションの手順では、SES コンソールでアーカイブを作成する方法について説明します。

Amazon SES コンソールでの E メールアーカイブの使用

SES コンソールの [E メールアーカイブ] ページは、[アーカイブの検索]、[検索履歴]、[履歴をエクスポート]、[アーカイブの管理] の 4 つのインタラクティブなテーブルで構成されており、アーカイブ内の E メールの検索、検索結果のエクスポート、アーカイブの管理に使用できます。以下では、各テーブルの手順を説明します。

アーカイブの検索、エクスポート、管理のために [E メールアーカイブ] ページを使用するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションパネルの [Mail Manager] で [E メールアーカイブ] を選択します。
3. [E メールアーカイブ] ページは、[アーカイブの検索]、[検索履歴]、[履歴をエクスポート]、[アーカイブの管理] の 4 つのテーブルで構成されています。これらの各テーブル独自の手順については、以下の対応するタブをクリックしてください。

Search archive

[アーカイブの検索] は、アーカイブされたメッセージを検索するためのインタラクティブなテーブルです。豊富なフィルターと日付セットにより詳細な検索条件が利用でき、特定の E メールから、広範なカテゴリに一致する多数の E メールまで、あらゆるメールを検索できます。指定した検索条件に一致するメッセージは、個別にダウンロードすることも、S3 バケットに一括エクスポートすることもできます。

アーカイブされた E メールを検索、ダウンロード、またはエクスポートするには

1. [アーカイブの検索] ページの [アーカイブの検索] タブをクリックして、[アーカイブの検索] テーブルを表示します。
2. [アーカイブ] フィールド内でクリックし、リストから [アーカイブの選択] を選択してから、[検索] をクリックするか、以下のステップに従って検索条件を絞り込みます。
3. [日付範囲] フィールドを選択して、検索に使用する以下の日付範囲オプションを展開します。

   - 相対範囲 (デフォルト) – 検索する日数に対応するラジオボタンをオンにするか、時間単位と最大 30 日間までの日付範囲を選んで、[カスタム範囲] を選択します。
   - 絶対範囲 – [開始日] と [終了日] (および必要に応じて時刻) を入力します。最大 30 日間まで指定できます。

> **Note**
>
> - アーカイブ内の検索は、一度に 30 日間分に制限されます。例えば、6 月 1 日から 7 月 31 日までのメッセージを検索する場合は、次のとおり 3 件の検索に分割する必要があります。
>
> 1. 6 月の 30 日間
> 2. 7 月の最初の 30 日間
> 3. 7 月 31 日
>
> - [相対範囲] の日付の場合、最後の日付の終了時刻は 0 時です。例えば、[Last 7 days] (過去 7 日間) を選択すると、7 日目は昨日の午前 0 時に終了することになります。

4. (オプション) [フィルター] フィールドを選択して、「送信元」、「送信先」、「CC」、「件名」、「添付ファイルあり」のフィルターから選択します。次のプロパティが適用されます。

   - フィルターは最大 10 件作成できます。
   - フィルターは、クリックして編集することも、[X] をクリックして削除することもできます。

5. [検索] を選択すると、指定した検索条件に一致するアーカイブ済みの E メールが [検索結果] テーブルに表示されます。

   - [メッセージ ID] 列はデフォルトでは非表示になっています。歯車アイコンをクリックしてテーブルの表示方法をカスタマイズすると、表示できます。
   - 実行するすべての検索は、一意の検索 ID を使用して自動的に保存され、[検索履歴] テーブルに一覧表示されます。

6. メッセージのテキストをエンベロープ情報とヘッダー情報と共に表示するには、メッセージのラジオボタンをオンにしてから、[詳細を表示] をクリックして、[メッセージの詳細] サイドバーを開きます。

7. メッセージのローカルファイルを作成するには、メッセージのラジオボタンをオンにして、[メッセージをダウンロード] を選択します。

8. フィルタリングした検索は、[S3 にエクスポート] をクリックして Amazon S3 バケットに保存できます。

a. 使用する S3 バケットの URI を把握している場合は、[S3 URI] フィールドに入力します。不明な場合は、[S3 を参照] を選択して、S3 ページで使用する S3 バケットとフォルダを選択します。

b. (オプション) エクスポートしたメッセージを暗号化するには、[KMS キー ARN] フィールドに独自の AWS KMS キーを入力するか、[新しいキーを作成] をクリックします。キーを指定しない場合、暗号化の方法は、送信先 S3 バケットで使用されている (使用されていない場合も) 任意の方法に設定されます。

c. [エクスポート] を選択すると、フィルタリングした検索で見つかったすべてのメッセージが、選択した S3 フォルダに個別のファイルとして保存されます。

> **Note**
>
> アーカイブに含めることができるメッセージの数に制限はありません。ただし、検索結果テーブルの [検索結果] は 1,000 行に制限されます。

Search history

検索の履歴はこのテーブルに一覧表示されているため、検索結果セットの復元や、以前に作成した複雑なフィルターセットの利用ができます。元の検索をベースにフィルターと日付を編集し、新しい検索を作成することもできます。新しい検索はすべて、一意の検索 ID を使用して自動的に保存され、このテーブルに一覧表示されます。

以前の検索を表示して操作するには

1. [E メールアーカイブ] ページの [検索履歴] タブをクリックして、[検索履歴] テーブルを表示します。このテーブルには、アーカイブ済みのすべての E メール検索の履歴が、最新のものから順に一覧表示されています。このテーブルは、初めてアクセスした際にデータをロードします。タブを切り替えてこのタブに戻る場合は、[更新] アイコンを使用して最新のデータを取得してください。
2. [アーカイブ] フィールド内をクリックし、リストから [アーカイブ] を選択すると、該当するアーカイブに属するすべての検索がテーブルに表示されます。以下のステップに従って、個別の検索を表示したり詳細を実行したりできます。
3. 以前の検索のラジオボタンをオンにしてから [検索結果の表示] をクリックすると、元の検索結果を復元できます。[アーカイブの検索] ページが開き、元の検索で使用されたフィルター

セットや日付範囲、この条件に基づいて以前検出されたすべてのメッセージが表示されます。元の検索は、以下の方法で拡張できます。

- 日付範囲とフィルターを変更して新しい検索を作成してから、[検索] をクリックします。
- 実行する新しい検索はすべて、一意の検索 ID を使用して自動的に保存され、[検索履歴] テーブルに一覧表示されます。

Export history

エクスポートの履歴は、このテーブルに一覧表示されており、エクスポートフォルダのコンテンツは、S3 コンソールで簡単にアクセスできます。

最近のエクスポートを表示するには

1. [E メールアーカイブ] ページの [履歴をエクスポート] タブをクリックすると、[検索履歴] テーブルが表示されます。このテーブルには、過去 30 日以内に S3 バケットにエクスポートしたアーカイブ済みのメール検索がすべて一覧表示されています。このテーブルは、初めてアクセスした際にデータをロードします。タブを切り替えてこのタブに戻る場合は、[更新] アイコンを使用して最新のデータを取得してください。
2. エクスポートのステータスが、「処理待ち」、「事前処理中」、または「処理中」の場合は、[キャンセル] をクリックしてキャンセルできます。
3. [S3 URI] をクリックすると、S3 コンソールでエクスポートのバケットフォルダを開き、フォルダ内のファイルを確認できます。

Manage archives

このテーブルにはアーカイブが一覧表示され、新しいアーカイブの作成、特定のアーカイブの検索と詳細の表示、アーカイブの編集、アーカイブの削除のオプションが提供されています。

アーカイブを作成して管理するには

1. [アーカイブの検索] ページの [アーカイブの管理] タブをクリックして、すべての E メールアーカイブが一覧表示されている [アーカイブ] テーブルを表示します。このテーブルは、初めてアクセスした際にデータをロードします。タブを切り替えてこのタブに戻る場合は、[更新] アイコンを使用して最新のデータを取得してください。
2. 特定のアーカイブを検索するには、[アーカイブ] フィールドに入力を開始します。
3. アーカイブの詳細を表示するには、[アーカイブ名] 列でその名前を選択します。

4. アーカイブを作成するには、[アーカイブの作成] をクリックします。

   a. [アーカイブ名] フィールドに一意の名前を入力します。

   b. (オプション) [保持期間] フィールドで保持期間を選択すると、デフォルトの保持期間の 180 日間を上書きできます。

   c. (オプション) アーカイブを暗号化するには、[KMS キー ARN] フィールドに独自の AWS KMS キーを入力するか、[新しいキーを作成] をクリックします。

   [Create archive] (アーカイブの作成) を選択します。

5. アーカイブを編集するには、ラジオボタンをオンにして、[編集] をクリックします。

   a. [アーカイブ名] フィールドの名前を編集するか、変更します。

   b. [保持期間] フィールドの保持期間を変更します。

   [アーカイブを更新] を選択します。

6. アーカイブを削除するには、ラジオボタンをオンにして、[削除] をクリックします。

   - [確認] フィールドに `delete` と入力してから、[削除] をクリックします。

     [アーカイブ] テーブルのアーカイブのステータスが削除保留中に切り替わり、30 日後に自動的に削除されます。

> **Note**
>
> この削除を元に戻す場合は、30 日以内に Amazon SES へのチケットを作成してください。

E メールアドオン

E メールアドオンとは、SES 認定プロバイダーの専用セキュリティツールのコレクションで、イングレスエンドポイントで許可する E メールのタイプを管理し、特定のタイプの E メールに対して実行するアクションを決定するために使用できます。このツールは、既存の E メールワークフローにそのまま統合できる認定済みのセキュリティインテリジェンスおよびセキュリティ適用ソリューションであり、Mail Manager コンソールから直接有効化できます。

これらのアドオンは、お客様のニーズに沿って最適化されていない可能性のある大規模な単一の製品ソリューションを購入するのと違い、従量制料金で使用でき、お客様独自のユースケースに適した検証済みの E メールセキュリティソリューションから選択できるという柔軟性を提供します。E メールアドオンは、コア脅威インテリジェンスとセキュリティ適用機能をワークロードごとに拡張するため、必要となるキャパシティについての推測も必要はありません。このような利点により、E メールのセキュリティ問題に積極的に取り組み、組織のために高レベルのサービス基準を維持することに集中できるようになります。

各アドオンの詳細については、Mail Manager コンソールにある E メールアドオンページから直接確認できます。このページでは、製品の説明、主な利点、料金情報にアクセスできます。使用するアドオンを決定したら、Mail Manager コンソールからサブスクライブします。サブスクライブすると、イングレスエンドポイントで許可される E メールを決定する際のトラフィックポリシー条件として、または特定の E メールに対して実行するアクションを決定するルールセット条件としてアドオンを選択できます。すべてのアドオンのプライマリサポートは AWS が提供します。サポートは、Mail Manager コンソールからもアクセスできます。

次のセクションの手順では、Mail Manager コンソールで E メールアドオンにサブスクライブする方法について詳しく説明します。

Mail Manager コンソールで E メールアドオンにサブスクライブする

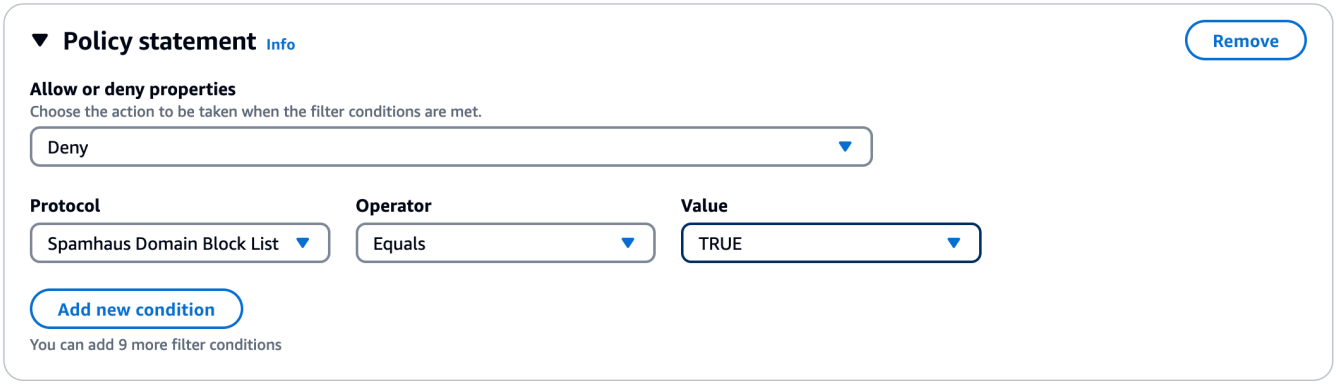
次の手順では、Mail Manager コンソールの [E メールアドオン] ページを使用してアドオンにサブスクライブし、任意のトラフィックポリシーまたはルールセットで使用できるようにする方法を説明します。

コンソールを使用して E メールアドオンにサブスクライブするには

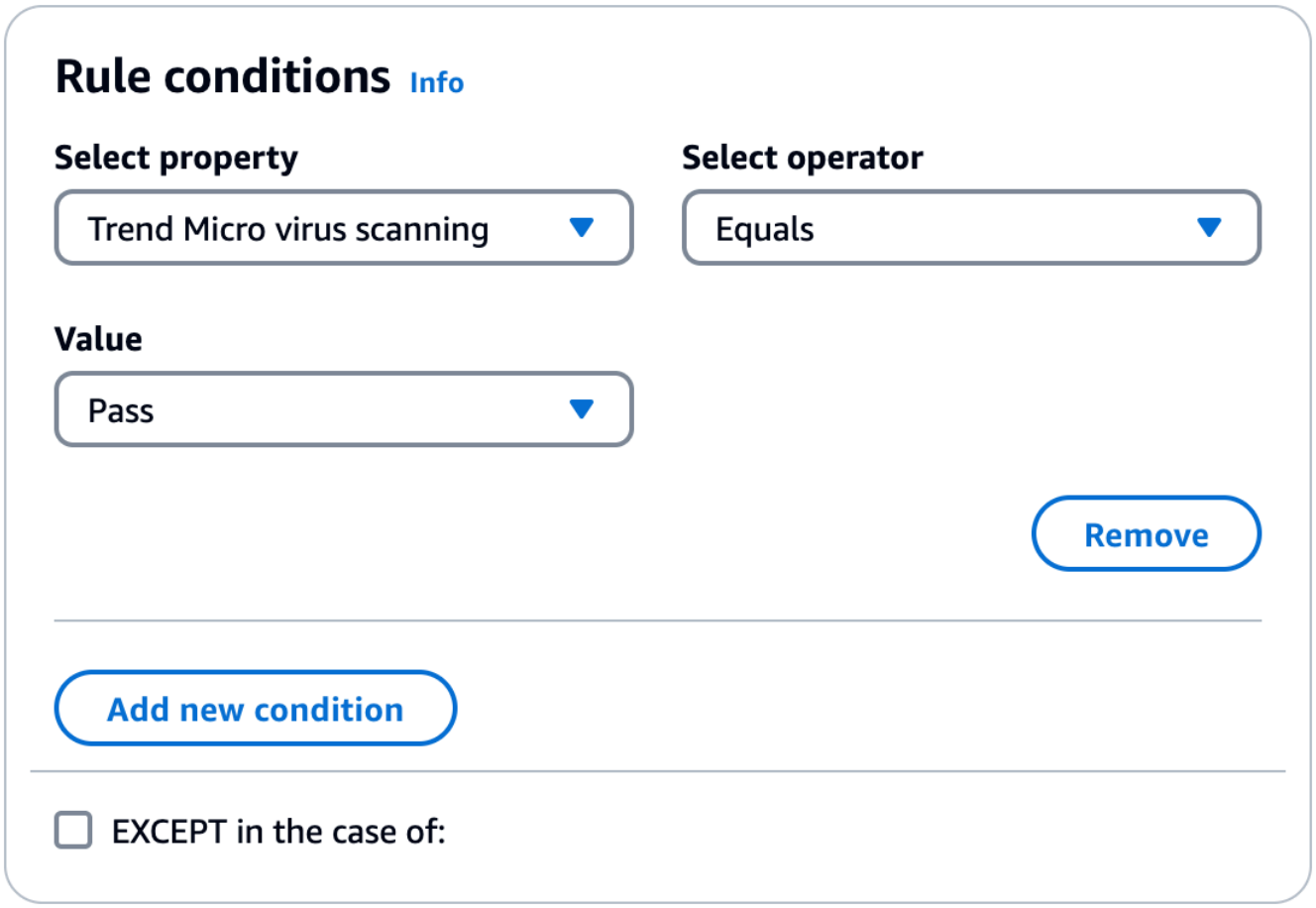
1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 左側のナビゲーションパネルの [Mail Manager] の下にある [E メールアドオン] を選択します。
3. [E メールアドオン] ページで、アドオンカードのいずれかのタイトルを選択して概要ページを開きます。概要ページでは、アドオンの機能、主な利点、料金情報の詳細を確認できます。このアドオンを使用する場合は、[サブスクライブ] を選択します。
   - 表示される [利用規約] を読み、[承諾します] ボックスをオンにして、[サブスクライブ] をクリックします。
4. アドオンにサブスクライブすると、イングレスエンドポイントへの E メールを拒否または許可するトラフィックポリシー条件として選択するか、対象となるメッセージに対して実行するアク

ションを決定するルールセット条件として選択して、E メールワークフローに統合できます。次の例は、ポリシーステートメント条件とルール条件でのアドオンの使用を説明しています。

- 次のとおり、ポリシーステートメント条件で [Spamhaus Domain Block List] アドオンを使用すると、Spamhaus にリストされているドメインから発信される E メールのイングレスエンドポイントでの受信をブロックできます。



- E メールアドオンを使用してトラフィックポリシーを作成し、ポリシーステートメント条件を構築する方法の詳細については、「the section called “トラフィックポリシーとポリシーステートメントの作成 (コンソール)”」を参照してください。
- 次のとおり、ルール条件で [Trend Micro のウイルススキャン] アドオンを使用すると、ウイルススキャンで問題なしと検証された E メールのルールアクションを決定できます。



- E メールアドオンを使用してルールセットを作成し、ルール条件を構築する方法の詳細については、「the section called “ルールセットとルールの作成 (コンソール)”」を参照してください。

5. サブスクライブしたアドオンの全般的な詳細を確認したり、サポートにアクセスしたりするには、[E メールアドオン] ページでアドオン名をクリックして、概要ページを開きます。

   - [全般的な詳細] では、サブスクライブした日付やアドオンの Amazon リソースネーム (ARN) を確認できます。
   - [サポート] タブをクリックして、AWS サポートへのリンクにアクセスします。

6. アドオンのサブスクライブを解除するには:

   a. まず、条件で定義したトラフィックポリシーまたはルールセットからアドオンを削除する必要があります。削除しないと、その後のサブスクライブ解除ステップは失敗します。

   b. [E メールアドオン] ページでアドオン名をクリックして概要ページを開いて、[サブスクリプションを解除] をクリックします。

c. [確認] フィールドに `confirm` と入力してから、[サブスクリプションを解除] をクリックします。

Mail Manager のアクセス許可ポリシー

この章のポリシーは、Mail Manager のさまざまな機能をすべて活用するうえで必要となるポリシーの単一のリファレンスポイントとなるように提供されています。

Mail Manager の機能ページには、機能を利用するために必要なポリシーを記載した、このページの各セクションへのリンクが提供されています。必要なポリシーのコピーアイコンをクリックして、各機能の説明の指示に従って貼り付けます。

次のポリシーでは、リソースアクセス許可ポリシーと ポリシーを通じて Amazon SES Mail Manager に含まれるさまざまな機能を使用するアクセス許可を付与します AWS Secrets Manager 。アクセス許可ポリシーを初めて使用する場合は、「the section called “ポリシー分析”」と「AWS Secrets Managerのアクセス許可ポリシー」を参照してください。

イングレスエンドポイントのアクセス許可ポリシー

このセクションのポリシーはどちらも、イングレスエンドポイントの作成に必要となります。イングレスエンドポイントを作成する方法と、これらのポリシーを使用する個所については、「the section called “イングレスエンドポイントの作成 (コンソール)”」を参照してください。

イングレスエンドポイントのための Secrets Manager シークレットのリソースアクセス許可ポリシー

がイングレスエンドポイントリソースを使用してシークレットにアクセスできるようにするには SES、次の Secrets Manager シークレットリソース許可ポリシーが必要です。

```
{
    "Version": "2012-10-17",
    "Id": "Id",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "000000000000"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:ses:us-east-1:000000000000:mailmanager-
ingress-point/*"
                }
            }
        }
    ]
}
```

KMS Ingress エンドポイントの カスタマーマネージドキー (CMK) キーポリシー

シークレットの使用中に がキーを使用SESできるようにするには、次のKMSカスタマーマネージドキー (CMK) キーポリシーが必要です。

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "ses.amazonaws.com"
    },
    "Action": "kms:Decrypt",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
           "kms:ViaService": "secretsmanager.us-east-1.amazonaws.com",
            "aws:SourceAccount": "000000000000"
        },
        "ArnLike": {
            "aws:SourceArn": "arn:aws:ses:us-east-1:000000000000:mailmanager-ingress-
point/*"
        }
    }
}
```

SMTP リレーのアクセス許可ポリシー

SMTP リレーを作成するには、このセクションのポリシーの両方が必要です。SMTP リレーを作成する方法と、これらのポリシーを使用する場所については、「」を参照してくださいthe section called “SMTP リレーの作成 (コンソール)”。

SMTP リレーの Secrets Manager シークレットリソース許可ポリシー

SMTP がリレーリソースを使用してシークレットにアクセスSESできるようにするには、次の Secrets Manager シークレットリソースのアクセス許可ポリシーが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Principal": {
                "Service": [
                    "ses.amazonaws.com"
                ]
            },
            "Resource": "*",
            "Condition": {
               "StringEquals": {
                   "aws:SourceAccount": "888888888888"
                },
               "ArnLike": {
                   "aws:SourceArn": "arn:aws:ses:us-east-1:888888888888:mailmanager-
smtp-relay/*"
               }
           }
        }
    ]
}
```

KMS SMTPリレー用の カスタマーマネージドキー (CMK) キーポリシー

シークレットの使用中に がキーを使用SESできるようにするには、次のKMSカスタマーマネージドキー (CMK) キーポリシーが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-east-1.amazonaws.com",
                    "aws:SourceAccount": "000000000000"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:ses:us-east-1:000000000000:mailmanager-
smtp-relay/*"
                }
            }
        }
    ]
}
```

E メールアーカイブのアクセス許可ポリシー

基本的なアーカイブIAMアイデンティティポリシー

これらは、アーカイブオペレーションを承認するための IAM ID ポリシーです。これらのポリシーだけでは、一部のオペレーションでは十分ではない場合があります (「 による保管時の暗号化のアーカイブKMSCMK」および「エクスポートのアーカイブ」を参照) 。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ses:CreateArchive",
                "ses:TagResource"
            ],
            "Resource": [
                "arn:aws:ses:us-east-1:000000000000:mailmanager-archive/*"
            ],
            "Condition": {
```

```
                "ForAnyValue:StringEquals": {
                    "aws:RequestTag/key-name": [
                        "value1",
                        "value2"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ses:ListArchives"
            ],
            "Resource": [
                "arn:aws:ses:us-east-1:000000000000:mailmanager-archive/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ses:GetArchive",
                "ses:DeleteArchive",
                "ses:UpdateArchive"
            ],
            "Resource": [
                "arn:aws:ses:us-east-1:000000000000:mailmanager-archive/MyArchiveID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ses:ListArchiveSearches"
            ],
            "Resource": [
                "arn:aws:ses:us-east-1:000000000000:mailmanager-archive/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ses:GetArchiveSearch",
                "ses:GetArchiveSearchResults",
                "ses:StartArchiveSearch",
                "ses:StopArchiveSearch"
```

```
            ],
            "Resource": [
                "arn:aws:ses:us-east-1:000000000000:mailmanager-archive/MyArchiveID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ses:GetArchiveMessage",
                "ses:GetArchiveMessageContent"
            ],
            "Resource": [
                "arn:aws:ses:us-east-1:000000000000:mailmanager-archive/MyArchiveID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ses:ListArchiveExports"
            ],
            "Resource": [
                "arn:aws:ses:us-east-1:000000000000:mailmanager-archive/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ses:GetArchiveExport",
                "ses:StartArchiveExport",
                "ses:StopArchiveExport"
            ],
            "Resource": [
                "arn:aws:ses:us-east-1:000000000000:mailmanager-archive/MyArchiveID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ses:ListTagsForResource",
                "ses:UntagResource"
            ],
            "Resource": [
                "arn:aws:ses:us-east-1:000000000000:mailmanager-archive/MyArchiveID"
            ]
```

```
        }
    ]
}
```

エクスポートのアーカイブ

これらは、 に必要な IAM ID ポリシー (上記の基本アーカイブポリシーに加えて) ですStartArchiveExport。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::MyDestinationBucketName"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectTagging",
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::MyDestinationBucketName/*"
        }
    ]
}
```

これは、保存先のバケットのためのポリシーです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
```

```
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::MyDestinationBucketName"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectTagging",
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::MyDestinationBucketName/*"
        }
    ]
}
```

> **Note**
>
> アーカイブは、混乱した代理条件キー (aws:、aws:SourceArn、SourceAccountaws:SourceOrgID、aws:) をサポートしていません SourceOrgPaths。これは、Mail Manager の E メールアーカイブでは、実際にエクスポートを開始する前に、転送アクセスセッションを使用して、呼び出し元の ID にエクスポート先のバケットへの書き込みアクセス許可があるかをテストすることで、混乱した代理の問題を回避しているためです。

を使用した保管時の暗号化のアーカイブ KMS CMK

これらは、アーカイブの作成と操作 (アーカイブ と呼ばれるCMK) に必要なKMSカスタマーマネージドキー () ポリシー (上記の基本アーカイブポリシーに加えて) による保管時の暗号化ですAPIs。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
```

```
        "Action": [
            "kms:DescribeKey",
            "kms:Decrypt",
            "kms:GenerateDataKey"
        ],
        "Resource": "arn:aws:kms:us-west-2:111122223333:key/MyKmsKeyArnID"
    }
}
```

これは、E メールのアーカイブに必要なKMSキーポリシーです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/MyUserRoleOrGroupName"
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "ses.us-east-1.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

ルールアクションを実行するためのアクセス許可と信頼ポリシー

SES ルール実行ロールは、 AWS サービスとリソースにアクセスするためのルール実行アクセス許可を付与する AWS Identity and Access Management （ IAM) ロールです。ルールセットでルールを作成する前に、必要な AWS リソースへのアクセスを許可するポリシーを持つ IAMロールを作成する必要があります。 は、ルールアクションを実行するときにこのロールをSES引き受けます。例えば、ルールの条件が満たされた場合に実行するルールアクションとして、S3 バケットに E メールメッセージを書き込むアクセス許可が付与されたルール実行ロールを作成できます。

したがって、S3 への書き込みルールアクション、メールボックスへの配信ルールアクション、インターネットへの送信ルールアクションを実行するために必要となる、このセクションに記載の個別のアクセス許可ポリシーに加え、以下の信頼ポリシーが必要となります。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "ses.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
          "StringEquals": {
                   "aws:SourceAccount": "888888888888"
          },
          "ArnLike": {
             "aws:SourceArn": "arn:aws:ses:us-east-1:888888888888:mailmanager-rule-set/
*"
          }
        }
      }
    ]
  }
```

S3 への書き込みルールアクションのアクセス許可ポリシー

受信した E メールを S3 バケットに配信する S3 への書き込みルールアクションを使用するには、以下のポリシーが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObject",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::MyDestinationBucketName/*"
            ]
        },
        {
            "Sid": "AllowListBucket",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::MyDestinationBucketName"
            ]
        }
    ]
}
```

サーバー側の暗号化が有効になっている S3 バケットに AWS KMS カスタマーマネージドキーを使用している場合は、IAMロールポリシーアクション を追加する必要があります"kms:GenerateDataKey*"。前の例を使用して、このアクションをポリシーに追加すると、以下のとおりになります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowKMSKeyAccess",
            "Effect": "Allow",
```

```
            "Action": "kms:GenerateDataKey*",
            "Resource": "arn:aws:kms:us-east-1:888888888888:key/*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "kms:ResourceAliases": [
                        "alias/MyKeyAlias"
                    ]
                }
            }
        }
    ]
}
```

AWS KMS キーへのポリシーのアタッチの詳細については、 AWS Key Management Service デベロッパーガイドの「 でのキーポリシーの使用 AWS KMS」を参照してください。

メールボックスに配信ルールアクションの許可ポリシー

受信した E メールを Amazon WorkMail アカウントに配信するメールボックスへの配信ルールアクションを使用するには、次のポリシーが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": ["workmail:DeliverToMailbox"],
        "Resource": "arn:aws:workmail:us-
east-1:888888888888:organization/MyWorkMailOrganizationID>"
      }
    ]
  }
```

インターネットに送信ルールアクションの許可ポリシー

受信した E メールを外部ドメインに送信するインターネットに送信ルールアクションを使用するには、以下のポリシーが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
```

```
        "Effect": "Allow",
        "Action": ["ses:SendEmail", "ses:SendRawEmail"],
        "Resource": "arn:aws:ses:us-east-1:888888888888:identity/example.com"
      }
    ]
  }
```

Q Business への配信ルールアクションのアクセス許可ポリシー

受信した E メールを Amazon Q Business インデックスに配信する「Deliver to Q Business」ルールアクションを使用するには、次のポリシーが必要です。

Amazon Q Business ポリシー：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccessToQBusiness",
            "Effect": "Allow",
            "Action": [
                "qbusiness:BatchPutDocument"
            ],
            "Resource": [
                "arn:aws:qbusiness:us-east-1:888888888888:application/ApplicationID/
index/IndexID"
            ]
        }
    ]
}
```

KMS Amazon Q Business の ポリシー：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccessToKMSKeyForQbusiness",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey*",
                "kms:Encrypt",
```

```
                "kms:DescribeKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:888888888888:key/*"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "qbusiness.us-east-1.amazonaws.com",
                    "kms:CallerAccount": "888888888888"
                 },
                 "ForAnyValue:StringEquals": {
                    "kms:ResourceAliases": [
                        "alias/MyKeyAlias"
                    ]
                }
            }
        }
    ]
}
```

AWS KMS キーへのポリシーのアタッチの詳細については、 AWS Key Management Service デベロッパーガイドの「 でのキーポリシーの使用 AWS KMS」を参照してください。

Mail Manager のログ記録

Mail Manager のログ記録は、Mail Manager オペレーションを詳細に可視化します。ログ記録機能は、設定されたルールセットとルールに基づいて、イングレスエンドポイントでの最初の受信からメッセージ処理までのメッセージフローを追跡します。

Mail Manager では、次のリソースのログ記録が提供されます。

- イングレスエンドポイント
- ルールセット

Mail Manager は Amazon CloudWatch Logs サービスを使用してログを配信し、ログは CloudWatch Logs、Amazon S3、または Amazon Data Firehose のいずれかの送信先に配信できます。

Mail Manager ログ配信の設定

動作しているログ配信は、次の 3 つの要素で構成されます。

- DeliverySource – ログを送信するリソースを表す論理オブジェクト。イングレスエンドポイントまたはルールセット。
- DeliveryDestination – 実際の配信先 (CloudWatch ログ、S3、または Firehose) を表す論理オブジェクト。
- 配信 — 配信ソースを配信先に接続します。

このセクションでは、Mail Manager のログ記録を使用するために必要なアクセス許可とともに、これらのオブジェクトを作成する方法について説明します。

前提条件

Mail Manager のログ記録を設定する前に、次の点を確認してください。

1. イングレスエンドポイントまたはルールセットを作成しました。
2. SES Mail Manager リソースから配信先に CloudWatch ログを配信するために必要な Logs および Mail Manager アクセス許可があります。

必要なアクセス許可

Amazon Logs ユーザーガイドの「追加のアクセス許可 CloudWatch [V2] を必要とするログ記録」セクションで説明されているように、発行されたログのアクセス許可を設定し、配信先に対応するアクセス許可を適用する必要があります。

- ログに送信された CloudWatch ログ
- Amazon S3 に送信されたログ
- Firehose に送信されたログ

さらに、Mail Manager では、ログ配信を設定するために次のユーザーアクセス許可が必要です。

- `ses:AllowVendedLogDeliveryForResource` - 例に示すように、Mail Manager がユーザーに代わって特定のリソースのログを CloudWatch Logs に配信できるようにするために必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
           "Sid": "AllowSesMailManagerLogDelivery",
```

```
            "Effect": "Allow",
             "Action": [
                 "ses:AllowVendedLogDeliveryForResource"
            ],
            "Resource" [
                "arn:aws:ses:us-east-1:1234567890:mailmanager-ingress-point/inp-xxxxx",
                "arn:aws:ses:us-east-1:1234567890:mailmanager-rule-set/rs-xxxx"
            ]
        }
    ]
}
```

SES コンソールでのログ記録の有効化

コンソールを使用して Mail Manager リソースのログ記録を有効にするには：

1. https://console.aws.amazon.com/ses/ で SES コンソールを開きます。
2. Mail Manager のナビゲーションペインで、イングレスエンドポイントまたはルールセットを選択し、ログ記録を有効にする特定のリソースを選択します。
3. リソースの詳細ページで、ログ配信の追加 を展開し、CloudWatch ログ、S3、または Firehose への配信を選択します。
4. 選択した送信先の配信先の追加ダイアログボックスで、プロンプトに従って送信先タイプに固有のログ配信オプションを設定します。
5. （オプション) 追加設定を展開して、レコードのフィールド、出力形式、フィールド区切り文字、および送信先タイプに固有のその他のパラメータをカスタマイズします。

CloudWatch ログを使用したログ記録の有効化 API

CloudWatch Logs を使用して Mail Manager リソースのログ記録を有効にするにはAPI、以下を実行する必要があります。

1. を使用して配信ソースを作成しますPutDeliverySource。
2. を使用して配信先を作成しますPutDeliveryDestination。
3. を使用して、1 つの配信元と 1 つの配信先をペアリングして配信を作成しますCreateDelivery。

Amazon CloudWatch Logs ユーザーガイドの「追加のアクセス許可 [V2]」セクションで、特定のログ記録先に必要なすべてのアクセス許可を持つIAMロールとアクセス許可ポリシーの例を表示

し、CloudWatch ログ、S3、Firehose などの特定のログ記録先リソースへの更新を許可するなど、ログ記録先に対するIAMロールとアクセス許可ポリシーの例に従うことができます。

> **Note**
>
> を作成する場合 DeliverySource、 は Ingress エンドポイントARNまたはルールセット `resourceArn`でARN、Mail Manager ログ`logType`の SES は に設定する必要がありますAPPLICATION_LOGS。

ログの解釈

ログを使用して、Mail Manager によって処理される受信メッセージのフローに関する追加のインサイトを取得できます。

次のセクションでは、各リソースのログのさまざまなフィールドについて詳しく説明します。

イングレスエンドポイントログ

ログはメッセージごとに生成されます。

```
{
  "resource_arn": "arn:aws:ses:us-east-1:1234567890:mailmanager-ingress-point/inp-
xxxxx",
  "event_timestamp": 1728562395042,
  "ingress_point_type": "OPEN" | "AUTH",
  "ingress_point_name": "MyIngressPoint",
  "message_id": "0000llcki1jmushh817gr586f963a5inhkvnh81",
  "message_size_bytes": 100000,
  "rule_set_id": "rs-xxxx",
  "sender_ip_address": "1.2.3.4",
  "smtp_mail_from": "someone@domain.com",
  "smtp_helo": "domain.com",
  "tls_protocol": "TLSv1.2",
  "tls_cipher_suite": "TLS_AES_256_GCM_SHA384",
  "recipients": ["me@mydomain.com", "you@mydomain.com", "they@mydomain.com"],
  "ingress_point_metadata": { // only applies to AUTH Ingress Endpoint
       "password_version": "",
       "secrets_manager_arn": ""
  }
}
```

> **Note**
>
> ログは、イングレスエンドポイントによって受け入れられるメッセージに対してのみ作成されます。すべての受信メッセージを拒否するイングレスエンドポイントは、ログを発行しません。

CloudWatch Logs Insights クエリの例

sender@domain.com からのクエリメッセージ：

```
fields @timestamp, @message, @logStream, @log
| filter smtp_mail_from like /sender@domain.com/
| sort @timestamp desc
| limit 10000
```

サイズが 5000 バイトを超えるメッセージをクエリします。

```
fields @timestamp, @message, @logStream, @log
| filter message_size_bytes > 5000
| sort @timestamp desc
| limit 10000
```

ルールセットログ

ログは、アクションごとにメッセージごとに生成されます。つまり、ルールセットのルールのアクションによってメッセージが処理されるたびにログレコードが生成されます。

```
{
   "resource_arn": "arn:aws:ses:us-east-1:1234567890:mailmanager-rule-set/rs-xxxx",
   "event_timestamp": 1732298258254,
   "message_id": "0000llcki1jmushh817gr586f963a5inhkvnh81",
   "rule_set_name": "MyRuleSet",
   "rule_name": "MyRule",
   "rule_index": 1,
   "recipients_matched": ["recipient1@domain.com", "recipient2@domain.com"],
   "action_metadata": {
       "action_name": "WRITE_TO_S3" | "DROP" | "RELAY" | "DELIVER_TO_MAILBOX" | etc.,
       "action_index": 2,
       "action_status": "SUCCESS" | "FAILURE" | "IN_PROGRESS",
       "action_failure": "Access denied"
```

```
    }
}
```

- recipients_matched – アクションが実行されているルールの条件に一致する受信者。
- rule_index – ルールセット内のルールの順序。
- action_index – ルール内のアクションの順序。
- action_status – 指定されたメッセージに対してアクションを実行した結果を示します。
- action_failure – アクションの失敗の詳細を示します (アクションが失敗した場合にのみ適用されます)。例えば、指定されたロールに アクションを実行するための十分なアクセス許可がない場合です。

さらに、ルール条件がメッセージと一致しなかった場合、つまり、メッセージがルールによって処理されない場合、メッセージがルールセットによって処理されたが、そのルールセットに対して実行されたアクションがないことを示す 1 つのログが発行されます。

```
{
   "resource_arn": "arn:aws:ses:us-east-1:1234567890:mailmanager-rule-set/rs-xxxx",
   "event_timestamp": 1732298258254,
   "message_id": "0000llcki1jmushh817gr586f963a5inhkvnh81",
   "rule_set_name": "MyRuleSet",
   "rule_name": "MyRule",
   "rule_index": 1,
   "recipients_matched": [],
}
```

CloudWatch Logs Insights クエリの例

特定の message-id をクエリする (ルールセットを介したメッセージフローを示します)。

```
fields @timestamp, @message, @logStream, @log
| filter message_id = 'message-id-123'
| sort @timestamp desc
| limit 10000
```

失敗した WRITE_TO_S3 アクションをクエリします。

```
fields @timestamp, @message, @logStream, @log
| filter action_metadata.action_name = 'WRITE_TO_S3'
    and action_metadata.action_status = 'FAILURE'
```

```
| sort @timestamp desc
| limit 10000
```

ルールセットの 2 番目のルールによって処理されなかったメッセージをクエリします (メッセージはルールの条件を満たしていません)。

```
fields @timestamp, @message, @logStream, @log
| filter recipients_matched = '[]'
    and rule_index = 2
| sort @timestamp desc
| limit 10000
```

Amazon Simple Email Service でのリストとサブスクリプションの管理

Amazon SES では、メーリングリストや定期購読のリスト、E メール抑制といった独自のリストを管理できます。送信者の評価を維持するために、SES では、無効な受信者への送信を防止し、送信者の評価を傷つけないようにする、アカウントレベルおよび設定セットレベルの抑制を提供しています。バウンスした E メールや苦情に対するもう 1 つの手段として、SES では、サブスクリプション管理を通じて、すべての送信メールにサブスクライブ解除リンクを自動的に追加できます。

これらのタイプのリストの各タイプについては、この章のトピックのセクションで詳しく説明していますが、ここではサプレッションリストの違いとグローバルサプレッションリスト管理での主な変更点を把握できるように、サプレッションリストの概要を説明します。この章で説明しているリストを使用する前に、この概要を一読することをお勧めします。

サプレッションリストとサプレッションオーバーライドメカニズムの概要

グローバルサプレッションリストの削除機能は、お客様向けでなくなり、お客様はサプレッション管理にこの機能を操作できなくなりました。グローバルサプレッションリストは、SES によりバックグラウンドで運用と管理が行われます。お客様は、アカウントレベルのサプレッションリストおよび設定セットレベルのサプレッションリストを利用できるようになりました。これにより、お客様のアカウントでの E メールサプレッションの処理方法について、よりカスタマイズされた管理ができるようになります。

さまざまなタイプのサプレッションリストとその範囲、それらが提供する利点について、次に説明します。

- グローバルサプレッションリスト – SES が所有して管理を行い、SES 共有 IP プール内のアドレスの評価を保護します。
- アカウントレベルのサプレッションリスト – お客様が所有および管理し、お客様のアカウントの評価を保護します。このリストは、グローバルサプレッションリストよりも優先されます。
  - 設定セットレベルのサプレッション – 設定セットで指定されたオーバーライドを使用して、アカウントレベルのサプレッションリストの条件付き制御やきめ細かな制御を提供するオーバーライドメカニズムです。

グローバルサプレッションリストは、新しい Amazon SES コンソールと API v2 で、アカウントレベルおよび設定セットレベルの抑制が導入されるまで、サプレッションリストの唯一のタイプでし

た。グローバルサプレッションリストは、SES が所有して管理し、SES の評価を保護します。これは SES の顧客すべてが同じ IP アドレスのプールを共有している (専用の IP アドレスでないかぎり) ために必要となります。SES にとっては、顧客がスパムを送信していないことや、SES が共有している IP プールの IP アドレスの評判に悪影響を与えるものを送信していないようにすることが重要です。グローバルサプレッションリストをお客様が直接操作することはなくなったとはいえ、グローバルサプレッションリストはバックグラウンドで引き続き動作しています。グローバルサプレッションリストの動作に関する一般的な原則は、他のタイプのサプレッションの動作に関する全体的な原則を理解するうえでも利用できます。「Amazon SES グローバルサプレッションリスト」を参照してください。

> **Note**
>
> アカウントレベルの抑制リストがこのセクションで説明するすべての利点に取って代わったため、グローバル抑制リスト削除リクエストフォームは Amazon SES コンソールからなくなりました。

アカウントレベルのサプレッションリストは、お客様が独自のサプレッションリストと評価を作成し、管理できるように導入されました。このため、アカウントレベルのサプレッションリストは、お客様のアカウントにのみ適用されます。新しいコンソールの、アカウントレベルのサプレッションリストのインターフェイスでは、アカウントレベルのサプレッションリストにあるアドレスを、簡単に管理できます。これには、アドレスの一括追加や一括削除のアクションが含まれます。アドレスがグローバルサプレッションリストに登録されてはいるが、カウントレベルのサプレッションリストには登録されていない (つまり、そのアドレスに送信したい) 場合に、そのアドレスに対して送信すると、 Amazon SES は送信を試みますが、もし送信したメールがバウンスした場合は、お客様自身の評価に悪影響を及ぼします。ただ、他の人については、独自にアカウントレベルのサプレッションリストを使用していなければ、誰もバウンスは発生しません。つまり、アカウントレベルのサプレッションリストがグローバルサプレッションリストをオーバーライドするのは、お客様のアカウントに関してのみとなります。「Amazon SESアカウントレベルのサプレッションリストの使用」を参照してください。

設定セットレベルのサプレッション自体は、リストではないとはいえ、さまざまな E メール送信シナリオそれぞれに専用に作成された設定リストを使用することにより、サプレッションのカスタマイズと、アカウントレベルのサプレッションリストのオーバーライドを設定できるメカニズムです。例えば、アカウントレベルのサプレッションリストが、バウンスアドレスと苦情アドレスの両方が追加されるように設定されているが、設定セットで定義された特定の E メール属性があり、追加される苦情アドレスのみに関心があるとします。この場合は、設定セットの抑制の上書きを有効にして、(アカウントレベルのサプレッションリストで設定されているバウンスや苦情ではなく) この設定セッ

トで送信された E メールからの苦情に対してのみ、アカウントレベルのサプレッションリストに E メールアドレスが追加されるようにします。設定セットレベルの抑制では、抑制をまったく使用しないなど、さまざまなレベルで、アカウントレベルの抑制を上書きします。「設定セットレベルの抑制を使用してアカウントレベルのサプレッションリストを上書きする」を参照してください。

Amazon SES グローバルサプレッションリスト

Amazon SES は内部のグローバルサプレッションリストを保持しています。これは SES によりバックグラウンドで運用と管理が行われています。SES のユーザーから送信された E メールがハードバウンスを起こすと、SES はバウンスを起こした E メールアドレスをグローバルサプレッションリストに追加します。グローバルサプレッションリストは、すべての SES のお客様に適用されるという意味で、グローバルです。つまり、別のカスタマーがグローバルサプレッションリストに登録されているアドレスに E メールを送信しようとした場合、SES はメッセージを受け付けますが、送信しません。これは、E メールアドレスが抑制されているためです。

グローバルサプレッションリストの E メールアドレス削除リクエスト機能は、お客様向けでなくなり、お客様はサプレッション管理にこの機能を操作できなくなりました。この機能の代わりに、Amazon SES ではアカウントレベルのサプレッションリストと設定セットレベルのサプレッションオーバーライドを利用できるようになりました。これにより、お客様のアカウントでの E メールサプレッションの処理方法について、よりカスタマイズされた管理ができるようになります。詳細については、Amazon SESアカウントレベルのサプレッションリストの使用および設定セットレベルの抑制を使用してアカウントレベルのサプレッションリストを上書きするを参照してください。

> **Important**
>
> グローバルサプレッションリストの E メールアドレス削除リクエストフォームは、アカウントレベルのサプレッションリストに置き換えられているため、Amazon SES のコンソールにはありません。アカウントレベルのサプレッションリストの使い方については、「Amazon SESアカウントレベルのサプレッションリストの使用」を参照してください。

グローバルサプレッションリストの考慮事項

グローバルサプレッションリストに関する主な要素

- グローバルサプレッションリストは、SES によりバックグラウンドで運用と管理が行われています - 直接操作することはできません。ただし、独自のアカウントレベルのサプレッションリストでサプレッションリストを上書きできます。。

- グローバルサプレッションリストは、すべての SES アカウントに対してデフォルトで有効になります。無効にすることはできません。
- SES はグローバルサプレッションリストをすべてのユーザーに適用するため、グローバルサプレッションリストにクエリを実行したり、このリストに手動でアドレスを追加したりすることはできません。
- E メールアドレスがハードバウンスを生成すると、SES はそのアドレスを短時間のグローバルサプレッションリストに追加します。この期間が経過すると、SES はリストからアドレスを削除します。アドレスが別のハードバウンスを生成した場合、SES はそれをより長い期間のグローバルサプレッションリストに追加し、その期間終了時に削除します。アドレスがグローバルサプレッションリストに残っている時間は、アドレスがハードバウンスを生成するたびに長くなります。アドレスは、グローバルサプレッションリストに最大 14 日間残ります。
- グローバルサプレッションリストに含まれているアドレスにメッセージを送信しようとすると、SES はメッセージを受け付けますが、送信はしません。SES はバウンス通知を生成します。通知には `Permanent` の `bounceType` 値および `Suppressed` の `bounceSubType` 値が含まれています。E メールアドレスがグローバルサプレッションリストに含まれているかどうかを確認するには、このタイプのバウンス通知を受信する方法しかありません。グローバルサプレッションリストをクエリすることはできません。
- SES は、グローバルサプレッションリストのアドレスに送信したメッセージを、アカウントのバウンス率と日ごとの送信クォータにカウントします。
- ハードバウンスを起こしたすべての E メールアドレスと同様、サプレッションリストのバウンスを起こしたアドレスは、それが有効なアドレスであることが確実である場合を除き、メーリングリストから削除してください。
- サプレッションリストのバウンスは、アカウントのバウンス率に反映されます。バウンス率が高すぎる場合、当社はお客様のアカウントを確認し、アカウントの E メール送信機能を一時停止することがあります。

> **Note**
>
> これらの SES サプレッションリストの相互関係と階層構造を理解しておくことが重要です。「サプレッションリストとサプレッションオーバーライドメカニズムの概要」を参照してください。

Amazon SESアカウントレベルのサプレッションリストの使用

Amazon SESアカウントレベルのサプレッションリストは、顧客が独自のサプレッションリストを作成および管理して評価を管理できるように導入されました。したがって、アカウントレベルのサプレッションリストはアカウントにのみ適用されます。SES コンソールのアカウントレベルのサプレッションリストインターフェイスを使用すると、アドレスを追加または削除するための一括アクションなど、アカウントレベルのサプレッションリスト内のアドレスを簡単に管理できます。

SES アカウントレベルのサプレッションリストは、現在の AWS アカウント の に適用されます AWS リージョン。SES API v2 または コンソールを使用して、アカウントレベルのサプレッションリストからアドレスを個別または削除できます。

> **Note**
>
> 住所を一括追加または削除するには、本番稼働用アクセスが必要です。サンドボックスの詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。

Amazon SES アカウントレベルのサプレッションリストに関する考慮事項

アカウントレベルのサプレッションリストを使用する場合は、次の要素を考慮する必要があります。

- 2019 年 11 月 25 SES日以降に Amazon の使用を開始した場合、アカウントはデフォルトでバウンスと苦情の両方にアカウントレベルのサプレッションリストを使用します。この日付SESより前に の使用を開始した場合は、 の SES `PutAccountSuppressionAttributes`オペレーションを使用してこの機能を有効にする必要がありますAPI。
- アカウントレベルのサプレッション設定で選択したのと同じサプレッション理由に一致するサプレッション理由があるアカウントレベルのサプレッションリストにあるアドレスにメッセージを送信しようとすると、 はメッセージSESを受け入れますが、送信しません。ただし、一致しない場合は、 SES はメッセージを送信します。これを明確にするために、以下の例を示します。
  - アカウントレベルのサプレッション設定をバウンスのみのサプレッション理由で設定した場合、SES はアカウントレベルのサプレッションリスト内のアドレスの配信を試行しません。サプレッション理由はバウンスです。ただし、 SESは、アカウントレベルのサプレッションリスト内のアドレスの配信を、抑制理由として苦情 (この場合は一致しないため) で試行します。

  - アカウントレベルのサプレッション設定をバウンスと苦情のサプレッション理由で設定した場合、 SES はアカウントレベルのサプレッションリスト内のアドレスの配信を試行せず、バウンスまたは苦情のサプレッション理由があります。
- SES は、アカウントレベルのサプレッションリストのアドレスに送信したメッセージを、アカウントの / 名前空間内の評価BounceRateまたは評価メトリクスComplaintRateにカウントしません。AWS SESこのようなメッセージは、 AWS/SES 名前空間のバウンスまたは苦情メトリクスでカウントされます。
- アドレスがグローバルサプレッションリストにあるが、アカウントレベルのサプレッションリストには含まれていない（つまり、送信する）場合、 SESは引き続き配信を試みます。ただし、バウンスしても、アカウントのバウンス率と毎日の送信クォータにカウントされます。
- SES は、アカウントレベルのサプレッションリストのアドレスに送信するメッセージを、毎日の送信クォータにカウントします。
- アカウントレベルのサプレッションリストの E メールアドレスは、削除するまで、リストに残ります。
- アカウントの E メール送信機能が一時停止されている場合、 は 90 日後にアカウントレベルのサプレッションリストのアドレスSESを自動的に削除します。この 90 日の期間が終了する前にアカウントのメール送信機能が復元された場合、アカウントレベルのサプレッションリストのアドレスは削除されません。
- Gmail では、SES に苦情データが提供されません。受取人が Gmail ウェブクライアントの [Spam] (スパム) ボタンを使用して、受信したメールを迷惑メールとして報告した場合、アカウントレベルのサプレッションリストには追加されません。
- アカウントがSESサンドボックスにある場合は、アカウントレベルのサプレッションリストを有効にできます。ただし、アカウントがサンドボックスから削除されるまで、PutSuppressedDestination または CreateImportJobオペレーションを使用することはできません。サンドボックスの詳細については、「本稼働アクセスのリクエスト (Amazon SES サンドボックスからの移行)」を参照してください。
- ハードバウンスのみが、アカウントレベルのサプレッションリストに追加されます。ソフトバウンドとハードバウンドの違いについては、「the section called “Amazon SES から E メールを送信した後”」を参照してください。
- アカウントレベルのサプレッションリストを使用すると、 は、ハードバウンスにつながるアドレスをグローバルサプレッションリストにもSES追加します。

Amazon SESアカウントレベルのサプレッションリストの有効化

Amazon v2 の SES API PutAccountSuppressionAttributesオペレーションを使用して、アカウントレベルのサプレッションリストを有効にしてセットアップできます。 AWS CLIを使用すると、この設定をすばやく簡単に設定できます。 AWS CLIのインストールおよび設定の詳細については、AWS Command Line Interface ユーザーガイドを参照してください。

を使用してアカウントレベルのサプレッションリストを設定するには AWS CLI

- コマンドラインで以下のコマンドを入力します。

  Linux, macOS, or Unix

  ```
  aws sesv2 put-account-suppression-attributes \
  --suppressed-reasons BOUNCE COMPLAINT
  ```

  Windows

  ```
  aws sesv2 put-account-suppression-attributes `
  --suppressed-reasons BOUNCE COMPLAINT
  ```

  アカウントレベルのサブレッションリストを有効にするには、suppressed-reasons パラメータに少なくとも 1 つの理由を指定する必要があります。前の例に示すように、BOUNCE もしくは COMPLAINT、または両方を指定することもできます。

SES コンソールを使用してアカウントレベルのサプレッションリストを設定するには：

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [Configuration] (設定)で、[Suppression list] (サプレッションリスト) を選択します。
3. [Account-level settings] (アカウントレベルの設定) ペインで [Edit] (編集) を選択します。
4. [Suppression list] (サプレッションリスト) で、[Enabled] (有効) ボックスにチェックをいれます。
5. [Suppression reasons] (サプレッションの理由) で、受信者の E メールアドレスをアカウントレベルのサプレッションリストに自動的に追加する必要のある理由を 1 つを選択します。

6. [Save changes] (変更の保存) をクリックします。

設定セットの Amazon SESアカウントレベルのサプレッションリストの有効化

Amazon SESアカウントレベルのサプレッションを設定して、特定の設定セットにのみ適用することもできます。この操作を行うと、バウンスイベントまたは苦情イベントの原因となった E メールを送信したときに設定セットを指定した場合にのみ、アドレスがサプレッションリストに追加されます。

> **Note**
>
> 次の手順では、 AWS CLIがインストール済みであるものとします。のインストールと設定の詳細については AWS CLI、「 AWS Command Line Interface ユーザーガイド」を参照してください。

を使用して設定セットのアカウントレベルのサプレッションリストを設定するには AWS CLI

- コマンドラインで以下のコマンドを入力します。

  Linux, macOS, or Unix

  ```
  aws sesv2 put-configuration-set-suppression-options \
  --configuration-set-name configSet \
  --suppressed-reasons BOUNCE COMPLAINT
  ```

  Windows

  ```
  aws sesv2 put-configuration-set-suppression-options `
  --configuration-set-name configSet `
  --suppressed-reasons BOUNCE COMPLAINT
  ```

  前の例では、 をアカウントレベルのサプレッションリストを使用する設定セットの名前*configSet*に置き換えます。

SES コンソールを使用して設定セットのアカウントレベルのサプレッションリストを設定するには：

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [Configuration] (設定)で、[Configuration sets] (設定セット) を選択します。
3. [Configuration sets] (設定セット) で、カスタマイズサプレッションで設定する設定セットの名前を選択します。
4. [Suppression list options] (サプレッションリストオプション) ペインで、[Edit] (編集) を選択します。
5. [Suppression list options] (サプレッションリスト) のオプションセクションには、この設定セットによりアカウントレベルの抑制を上書きするオプションなどを含む決定セットが用意されています。設定セットレベルのサプレッションロジックマップによって、オーバーライドが組み合わさった結果が、どのように影響するのかが分かります。これらの多層オーバーライドを選択し組み合わせることで、3 つの異なるレベルの抑制を実装できます。

   a. アカウントレベルのサプレッションを使用してください。アカウントレベルの抑制を上書きしたり、設定セットレベルの抑制を実装したりしないでください。基本的に、この設定セットを使用して送信される E メールは、アカウントレベルの抑制のみを使用します。これを実行するには:

      - [Suppression list settings] (サプレッションリストの設定) で、[Override account level settings] (アカウントレベルの設定を上書きする) ボックスのチェックを外します。

   b. 抑制は使用しないでください。設定セットレベルの抑制は有効にせずに、アカウントレベルの抑制を上書きします。この設定セットを使用して送信される E メールは、アカウントレベルの抑制を使用しないことになります。言い換えると、すべての抑制がキャンセルされます。これを実行するには:

      i. [Suppression list settings] (サプレッションリストの設定) で、[Override account level settings] (アカウントレベルの設定を上書きする) ボックスにチェックを付けます。

      ii. [Suppression list] (サプレッションリスト) で、[Enabled] (有効) ボックスのチェックを外します。

   c. 設定セットレベルの抑制を使用してください。この設定セットで定義されたカスタムのサプレッションリストの設定を使用して、アカウントレベルの抑制を上書きします。この設定

セットを使用して送信される E メールは、独自の抑制設定のみを使用することになり、アカウントレベルの抑制設定は無視されます。これを実行するには:

i. [Suppression list settings] (サプレッションリストの設定) で、[Override account level settings] (アカウントレベルの設定を上書きする) ボックスにチェックを付けます。
ii. [Suppression list] (サプレッションリスト) で [Enabled] (有効) にチェックを入れます。
iii. [Specify the reason(s)...] (理由を指定) で、この設定セットで使用する抑制の理由を 1 つ選択します。

6. [Save changes] (変更の保存) をクリックします。

Amazon SESアカウントレベルのサプレッションリストへの個別の E メールアドレスの追加

v2 の PutSuppressedDestinationオペレーションを使用して、Amazon SES API SESアカウントレベルのサプレッションリストに個別のアドレスを追加できます。アカウントレベルのサプレッションリストに追加できるアドレスの数に制限はありません。

> **Note**
>
> 次の手順では、 AWS CLIがインストール済みであるものとします。のインストールと設定の詳細については AWS CLI、「 AWS Command Line Interface ユーザーガイド」を参照してください。

AWS CLIを使用してアカウントレベルのサプレッションリストに個別のアドレスを追加するには

- コマンドラインで以下のコマンドを入力します。

  Linux, macOS, or Unix

  ```
  aws sesv2 put-suppressed-destination \
  --email-address recipient@example.com \
  --reason BOUNCE
  ```

  Windows

  ```
  aws sesv2 put-suppressed-destination `
  --email-address recipient@example.com `
  ```

```
--reason BOUNCE
```

前の例では、 をアカウントレベルのサプレッションリストに追加する E メールアドレス*recipient@example.com*に置き換え、 をサプレッションリストにアドレスを追加する*BOUNCE*理由に置き換えます (許容値は BOUNCEと ですCOMPLAINT) 。

SES コンソールを使用してアカウントレベルのサプレッションリストに個別のアドレスを追加するには：

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [Configuration] (設定)で、[Suppression list] (サプレッションリスト) を選択します。
3. [Suppression list] (サプレッションリスト) ペインで、[Add email address] (E メールアドレスの追加) を選択します。
4. [Email address] (E メールアドレス) フィールドに E メールアドレスを入力し、[Suppression reason] (抑制の理由) で理由を選択します。さらにアドレスを入力する必要がある場合は、[Add another address] (別のアドレスを追加) を選択し、追加のアドレスごとに繰り返します。
5. アドレスの入力が完了したら、入力内容が正確であるかを確認します。入力した一部が送信すべきではないと判断したときは、[Remove] (削除) ボタンを選択してください。
6. 入力したメールアドレスをアカウントレベルのサプレッションリストに追加するには、[Save changes] (変更を保存) を選択します。

Amazon SESアカウントレベルのサプレッションリストへの E メールアドレスの一括追加

アドレスを一括追加するには、まず連絡先リストを Amazon S3 オブジェクトにアップロードしてから、Amazon v2 の SES API CreateImportJobオペレーションを使用します。

> **Note**
>
> - アカウントレベルのサプレッションリストに追加できるアドレスの数に制限はありませんが、API呼び出しごとに Amazon S3 オブジェクトに 100,000 アドレスの一括追加制限があります。

- データソースが S3 バケットの場合は、インポート先と同じリージョンに存在する必要があります。

アカウントレベルのサプレッションリストにメールアドレスを一括追加するには、次の手順を実行します。

- アドレスリストを CSVまたは JSON形式で Amazon S3 オブジェクトにアップロードします。

  CSV アドレスを追加するための 形式の例：

  recipient1@example.com,*BOUNCE*

  recipient2@example.com,*COMPLAINT*

  改行で区切られたJSONファイルのみがサポートされます。この形式では、各行は個別のアドレス定義を含む完全なJSONオブジェクトです。

  JSON アドレスを追加する 形式の例：

  {"emailAddress":"*recipient1@example.com*","reason":"*BOUNCE*"}

  {"emailAddress":"*recipient2@example.com*","reason":"*COMPLAINT*"}

  前の例では、 *recipient1@example.com*と *recipient2@example.com*をアカウントレベルのサプレッションリストに追加する E メールアドレスに置き換えます。サプレッションリストにアドレスを追加する理由として受け入れられるのは、*BOUNCE* および *COMPLAINT* です。
- Amazon S3 オブジェクトを読み取るアクセスSES許可を付与します。

  Amazon S3 バケットに適用すると、次のポリシーにより、そのバケットを読み取るSESアクセス許可が に付与されます。Amazon S3 バケットの添付ポリシーの詳細については、Amazon Simple Storage Service ユーザーガイド の「バケットポリシーとユーザーポリシーの使用 」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSESGet",
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "ses.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::BUCKET-NAME/OBJECT-NAME",
            "Condition": {
                "StringEquals": {
                    "aws:Referer": "AWSACCOUNTID"
                }
            }
        }
    ]
}
```

- AWS KMS キーを使用するアクセスSES許可を付与します。

  Amazon S3 オブジェクトが AWS KMS キーで暗号化されている場合は、 AWS KMS キーを使用するためのアクセスSES許可を Amazon に付与する必要があります。 は、デフォルトキーではなく、カスタマーマネージドKMSキーからのみアクセス許可を取得SESできます。ステートメントをキーのポリシーに追加して、カスタマーマネージドキーを使用するためのSESアクセス許可をに付与する必要があります。

  次のポリシーステートメントをキーポリシーに貼り付けて、 SESがカスタマーマネージドキーを使用できるようにします。

```
{
   "Sid": "AllowSESToDecrypt",
   "Effect": "Allow",
   "Principal": {
       "Service":"ses.amazonaws.com"
   },
   "Action": [
       "kms:Decrypt",
   ],
   "Resource": "*"
}
```

- v2 SES API で CreateImportJobオペレーションを使用します。

> **Note**
>
> 次の例では、 AWS CLIがインストール済みであるものとします。のインストールと設定の詳細については AWS CLI、「AWS Command Line Interface ユーザーガイド」を参照してください。

コマンドラインで、以下のコマンドを入力します。を Amazon S3 バケットの名前*s3bucket*に、 を Amazon S3 オブジェクトの名前*s3object*に置き換えます。

```
aws sesv2 create-import-job --import-destination
 SuppressionListDestination={SuppressionListImportAction=PUT} --import-data-source
 S3Url=s3://s3bucket/s3object,DataFormat=CSV
```

SES コンソールを使用して E メールアドレスをアカウントレベルのサプレッションリストに一括で追加するには：

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [Configuration] (設定)で、[Suppression list] (サプレッションリスト) を選択します。
3. [Suppression list] (サプレッションリスト) テーブルで、[Bulk actions] (一括アクション) ボタンを展開し、[Add email addresses in bulk] (E メールアドレスを一括で追加する) を選択します。
4. Bulk action specifications (一括アクションの仕様) では、(a) [Choose file from S3 bucket] (S3 バケットからファイルを選択) するか (b) [Import from file] (ファイルからインポート) を選択します。

    a. S3 バケットからファイルを選択 - ソースファイルがすでに Amazon S3 バケットに保存されている場合:

        i. 使用する Amazon S3 バケットURIの がわかっている場合は、Amazon S3 URIフィールドに入力します。それ以外の場合は、S3 の参照を選択します。

            A. [Buckets] (バケット) で、S3 バケットの名前を選択します。

            B. [Objects] (オブジェクト) で、ファイル名を選択し、次に [Choose] (選択)を選択します。[Bulk action specifications] (一括アクションの仕様) に戻ります 。

C. (オプション) Amazon S3 コンソールに移動して、S3 オブジェクトの詳細を表示するには、[View] (表示) を選択します。

ii. [File format] (ファイル形式) で、Amazon S3 バケットからインポートするために選択したファイルの形式を選択します。

iii. [Add email addresses] (E メールアドレスの追加)を選択して、ファイルからのアドレスのインポートを開始します。[Bulk actions] (一括操作) タブの下にテーブルが表示されます。

b. ファイルからのインポート - 新規または既存の Amazon S3 バケットにアップロードするローカルソースファイルがある場合:

i. [Import source file] (ソースファイルのインポート) で、[Choose file] (ファイルの選択) を選択します。

ii. CSV ファイルブラウザで JSONまたは ファイルを選択し、開く を選択します。ファイルの名前、サイズ、日付がファイルの選択ボタンの下に表示されます。

iii. [Amazon S3 bucket] (Amazon S3 バケット) を展開し、S3 バケットを選択します。

- 新しいバケットにファイルをアップロードするには、[Create S3 bucket] (S3 バケットの作成) を選択し、[Bucket name] (バケット名) に名前を入力し、[Create bucket] (バケットの作成) を選択します。

iv. [Add email addresses] (E メールアドレスの追加)を選択して、ファイルからのアドレスのインポートを開始します。[Bulk actions] (一括操作) タブの下にテーブルが表示されます。

5. 使用したインポート方法に関係なく、ジョブ ID が [Bulk actions] (一括アクション) にインポートの種類、ステータス、日付とともに一覧表示されます。ジョブの詳細を表示するには、ジョブIDを選択します。

6. Suppression list (サプレッションリスト) タブを選択すると、正常にインポートされたすべての E メールアドレスが、抑制の理由と日付が追加されて表示されます。次のオプションを使用できます。

a. E メールアドレスを選択するか、対応するチェックボックスを選択して[View report] (レポートの表示) をクリックして、詳細を表示します。 (バウンスや苦情が原因で自動的にサプレッションリストに追加されたアドレスである場合は、トリガーイベントを生成した E メールメッセージの詳細など、追加の原因となったフィードバックイベントに関する情報が表示されます。)

b. アカウントサプレッションリストから削除する 1 つ以上のE メールアドレスの対応するチェックボックスをオンにし、[Remove] (削除) を選択します。

Amazon SESアカウントレベルのサプレッションリストにあるアドレスのリストの表示

v2 の ListSuppressedDestinationsオペレーションを使用して、アカウントのアカウントレベルのサプレッションリストにあるすべての E SES API メールアドレスのリストを表示できます。

> **Note**
>
> 次の手順では、 AWS CLIがインストール済みであるものとします。のインストールと設定の詳細については AWS CLI、「 AWS Command Line Interface ユーザーガイド」を参照してください。

アカウントレベルのサプレッションリストにあるすべての E メールアドレス一覧表示するには

- コマンドラインで以下のコマンドを入力します。

```
aws sesv2 list-suppressed-destinations
```

上記のコマンドは、アカウントのアカウントレベルのサプレッションリストにあるすべての E メールアドレスを返します。出力は以下の例のようになります。

```
{
    "SuppressedDestinationSummaries": [
        {
            "EmailAddress": "recipient2@example.com",
            "Reason": "COMPLAINT",
            "LastUpdateTime": "2020-04-10T21:03:05Z"
        },
        {
            "EmailAddress": "recipient0@example.com",
            "Reason": "COMPLAINT",
            "LastUpdateTime": "2020-04-10T21:04:26Z"
        },
        {
```

```
            "EmailAddress": "recipient1@example.com",
            "Reason": "BOUNCE",
            "LastUpdateTime": "2020-04-10T22:07:59Z"
        }
    ]
}
```

- 注 – 出力に文字列値を含むNextToken「」フィールドが含まれている場合、アカウントのサプレッションリストに追加の E メールアドレスがあることを示します。追加の抑制アドレスを表示するには、ListSuppressedDestinations へ別のリクエストを発行し、返された文字列値を以下のように --next-token パラメータに渡します :

  ```
  aws sesv2 list-suppressed-destinations --next-token string
  ```

  前述のコマンドで、 を返された NextToken 値*string*に置き換えます。

  詳細については、「How to list over 1000 email addresses from account-level suppression list」を参照してください。

StartDate オプションを使用すると、特定の日付以降にリストに追加された E メールアドレスのみを表示できます。

特定の日付の後にアカウントレベルのサプレッションリストに追加されたアドレスを一覧表示するには

- コマンドラインで以下のコマンドを入力します。

  ```
  aws sesv2 list-suppressed-destinations --start-date 1604394130
  ```

  前述のコマンドで、 を開始日の Unix タイムスタンプ*1604394130*に置き換えます。

EndDate オプションを使用して、特定の日付以前にリストに追加された E メールアドレスのみを表示することもできます。

特定の日付の前にアカウントレベルのサプレッションリストに追加されたアドレスを一覧表示するには

- コマンドラインで以下のコマンドを入力します。

```
aws sesv2 list-suppressed-destinations --end-date 1611126000
```

前述のコマンドで、 を終了日の Unix タイムスタンプ*1611126000*に置き換えます。

Linux、macOS、または Unix コマンドラインで、組み込みの grep ユーティリティを使用して、特定のアドレスまたはドメインを検索することもできます。

アカウントレベルのサプレッションリストで特定のアドレスを検索するには

- コマンドラインで以下のコマンドを入力します。

```
aws sesv2 list-suppressed-destinations | grep -A2 'example.com'
```

前述のコマンド*example.com*で、 を、検索するテキストの文字列 (アドレスやドメインなど) に置き換えます。

SES コンソールを使用して、アカウントレベルのサプレッションリストにあるすべての E メールアドレスのリストを表示するには：

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [Configuration] (設定)で、[Suppression list] (サプレッションリスト) を選択します。
3. [Suppression list] (サプレッションリスト) ペインでは、アカウントレベルのサプレッションリストのすべての E メールアドレスに、抑制の理由と日付が追加された状態で表示されます。次のオプションを使用できます。

   a. E メールアドレスを選択するか、対応するチェックボックスを選択して[View report] (レポートの表示) をクリックして、詳細を表示します。 (バウンスや苦情が原因で自動的にサプレッションリストに追加されたアドレスである場合は、トリガーイベントを生成した E メールメッセージの詳細など、追加の原因となったフィードバックイベントに関する情報が表示されます。)

   b. サプレッションリストテーブルは、歯車アイコンを選択することでカスタマイズできます。モーダルが表示され、ページサイズ、行の折り返し、および表示する列をカスタマイズできます。選択した後、 [Confirm] (確認) をクリックします。サプレッションリストテーブルには、表示の選択が反映されます。

Amazon SESアカウントレベルのサプレッションリストから個々の E メールアドレスを削除する

アドレスがアカウントのサプレッションリストにあるが、そのアドレスがリストに含まれていないことがわかっている場合は、v2 SES API の DeleteSuppressedDestinationオペレーションを使用してアドレスを削除できます。

> **Note**
>
> 次の手順では、 AWS CLIがインストール済みであるものとします。のインストールと設定の詳細については AWS CLI、「 AWS Command Line Interface ユーザーガイド」を参照してください。

AWS CLIを使用してアカウントレベルのサプレッションリストから個別のアドレスを削除するには

- コマンドラインで以下のコマンドを入力します。

  Linux, macOS, or Unix

  ```
  aws sesv2 delete-suppressed-destination \
  --email-address recipient@example.com
  ```

  Windows

  ```
  aws sesv2 delete-suppressed-destination `
  --email-address recipient@example.com
  ```

  前の例では、 をアカウントレベルのサプレッションリストから削除する E メールアドレス*recipient@example.com*に置き換えます。

SES コンソールを使用してアカウントレベルのサプレッションリストから個々のアドレスを削除するには：

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。

2. ナビゲーションペインの [Configuration] (設定)で、[Suppression list] (サプレッションリスト) を選択します。
3. (a)テーブル選択または(b)入力された内容によって個別の E メールアドレスを削除します。

   a. テーブルから選択: [Suppression list] (サプレッションリスト) テーブルで、1 つ以上の E メールアドレスの対応するチェックボックスを選択し、[Remove] (削除) をクリックします。

   b. フィールドに入力:

      i. [Suppression list] (サプレッションリスト) で、[Remove email address] (E メールアドレスの削除) を選択します。

      ii. [Email address] (メールアドレス) フィールドに E アドレスを入力します。さらに、アドレスを入力する必要がある場合は、[Enter another address] (別のアドレスを入力) を選択して入力し、追加のアドレスごとにこの操作を繰り返します。

      iii. アドレスの入力が完了したら、入力内容が正確であるかを確認します。入力した一部が送信すべきではないと判断したときは、[Remove] (削除) ボタンを選択してください。

      iv. [Save changes] (変更の保存) を選択して、入力したメールアドレスをアカウントレベルのサプレッションリストから削除します。

Amazon SESアカウントレベルのサプレッションリストから E メールアドレスを一括削除する

アドレスを一括で削除するには、まず連絡先リストを Amazon S3 オブジェクトにアップロードしてから、v2 の SES API CreateImportJobオペレーションを使用します。

> **Note**
>
> - アカウントレベルのサプレッションリストから削除できるアドレスの数に制限はありませんが、API呼び出しごとに Amazon S3 オブジェクトに 10,000 個のアドレスの一括削除制限があります。
> - データソースが S3 バケットの場合は、インポート先と同じリージョンに存在する必要があります。

アカウントレベルのサプレッションリストからメールアドレスを一括で削除するには、次の手順を実行します。

- アドレスリストを CSVまたは JSON形式で Amazon S3 オブジェクトにアップロードします。

  CSV アドレスを削除するための 形式の例：

  recipient3@example.com

  改行で区切られたJSONファイルのみがサポートされます。この形式では、各行は個別のアドレス定義を含む完全なJSONオブジェクトです。

  JSON アドレスを追加する 形式の例：

  {"emailAddress":"*recipient3@example.com*"}

  前の例では、 をアカウントレベルのサプレッションリストから削除する E メールアドレス*recipient3@example.com*に置き換えます。

- Amazon S3 オブジェクトを読み取るアクセスSES許可を付与します。

  Amazon S3 バケットに適用すると、次のポリシーにより、そのバケットを読み取るSESアクセス許可が に付与されます。Amazon S3 バケットの添付ポリシーの詳細については、Amazon Simple Storage Service ユーザーガイド の「バケットポリシーとユーザーポリシーの使用 」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSESGet",
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::BUCKET-NAME/OBJECT-NAME",
            "Condition": {
                "StringEquals": {
                    "aws:Referer": "AWSACCOUNTID"
                }
            }
        }
    ]
}
```

- AWS KMS キーを使用するアクセスSES許可を付与します。

  Amazon S3 オブジェクトが AWS KMS キーで暗号化されている場合は、 AWS KMS キーを使用するためのアクセスSES許可を Amazon に付与する必要があります。 は、デフォルトキーではなく、カスタマーマネージドKMSキーからのみアクセス許可を取得SESできます。ステートメントをキーのポリシーに追加して、カスタマーマネージドキーを使用するためのSESアクセス許可をに付与する必要があります。

  次のポリシーステートメントをキーポリシーに貼り付けて、 SESがカスタマーマネージドキーを使用できるようにします。

```
{
   "Sid": "AllowSESToDecrypt",
   "Effect": "Allow",
   "Principal": {
       "Service":"ses.amazonaws.com"
   },
   "Action": [
       "kms:Decrypt",
   ],
   "Resource": "*"
}
```

- v2 SES API で CreateImportJobオペレーションを使用します。

> **Note**
>
> 次の例では、 AWS CLIがインストール済みであるものとします。のインストールと設定の詳細については AWS CLI、「 AWS Command Line Interface ユーザーガイド」を参照してください。

コマンドラインで、以下のコマンドを入力します。を Amazon S3 バケットの名前*s3bucket*に、 を Amazon S3 オブジェクトの名前*s3object*に置き換えます。

```
aws sesv2 create-import-job --import-destination
 SuppressionListDestination={SuppressionListImportAction=DELETE} --import-data-source
 S3Url="s3://s3bucket/s3object",DataFormat=CSV
```

SES コンソールを使用してアカウントレベルのサプレッションリストから E メールアドレスを一括削除するには：

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [Configuration] (設定)で、[Suppression list] (サプレッションリスト) を選択します。
3. [Suppression list] (サプレッションリスト) テーブルで、[Bulk actions] (一括アクション) ボタンを展開し、[Remove email addresses in bulk] (E メールアドレスを一括で削除) をクリックします。
4. Bulk action specifications (一括アクションの仕様) では、(a) [Choose file from S3 bucket] (S3 バケットからファイルを選択) または (b) [Import from file] (ファイルからインポート) を選択します。手順は、インポート方法ごとに示されています。

   a. S3 バケットからファイルを選択 - ソースファイルがすでに Amazon S3 バケットに保存されている場合:

      i. 使用する Amazon S3 バケットURIの がわかっている場合は、Amazon S3 URIフィールドに入力します。それ以外の場合は、S3 の参照を選択します。

         A. [Buckets] (バケット) で、S3 バケットの名前を選択します。

         B. [Objects] (オブジェクト) で、ファイル名を選択し、次に [Choose] (選択)を選択します。[Bulk action specifications] (一括アクションの仕様) に戻ります 。

         C. (オプション) Amazon S3 コンソールに移動して、S3 オブジェクトの詳細を表示するには、[View] (表示) を選択します。

      ii. [File format] (ファイル形式) で、Amazon S3 バケットからインポートするために選択したファイルの形式を選択します。

      iii. [Remove email addresses] (E メールアドレスの削除) を選択して、ファイルからのアドレスのインポートを開始します。[Bulk actions] (一括操作) タブの下にテーブルが表示されます。

   b. ファイルからのインポート - 新規または既存の Amazon S3 バケットにアップロードするローカルソースファイルがある場合:

      i. [Import source file] (ソースファイルのインポート) で、[Choose file] (ファイルの選択) を選択します。

ii. CSV ファイルブラウザで JSONまたは ファイルを選択し、開く を選択します。ファイルの名前、サイズ、日付がファイルの選択ボタンの下に表示されます。

iii. [Amazon S3 bucket] (Amazon S3 バケット) を展開し、S3 バケットを選択します。

- 新しいバケットにファイルをアップロードするには、[Create S3 bucket] (S3 バケットの作成) を選択し、[Bucket name] (バケット名) に名前を入力し、[Create bucket] (バケットの作成) を選択します。

iv. [Remove email addresses] (E メールアドレスの削除) を選択して、ファイルからのアドレスのインポートを開始します。[Bulk actions] (一括操作) タブの下にテーブルが表示されます。

5. 使用したインポート方法に関係なく、ジョブ ID が [Bulk actions] (一括アクション) にインポートの種類、ステータス、日付とともに一覧表示されます。ジョブの詳細を表示するには、ジョブIDを選択します。
6. [Suppression list] (サプレッションリスト) タブを選択し、サプレッションリストから削除された、正常にインポートされたすべての E メールアドレスが表示されなくなります。

アカウントのインポートジョブのリストの表示

Amazon v2 の ListImportJobsオペレーションを使用して、アカウントのアカウントレベルのサプレッションリストにあるすべての E SES API メールアドレスのリストを表示できます。

> **Note**
>
> 次の手順では、 AWS CLIがインストール済みであるものとします。のインストールと設定の詳細については AWS CLI、AWS Command Line Interface 「 ユーザーガイド」を参照してください。

アカウントのすべてのインポートジョブのリストを表示するには

- コマンドラインで以下のコマンドを入力します。

```
aws sesv2 list-import-jobs
```

上記のコマンドは、アカウントのすべてのインポートジョブを返します。出力は以下の例のようになります。

```
{
    "ImportJobs": [
        {
            "CreatedTimestamp": 2020-07-31T06:06:55Z",
            "ImportDestination": {
                "SuppressionListDestination": {
                    "SuppressionListImportAction": "PUT"
                }
            },
            "JobStatus": "COMPLETED",
            "JobId": "755380d7-fbdb-4ed2-a9a3-06866220f5b5"
        },
        {
            "CreatedTimestamp": "2020-07-30T18:45:32Z",
            "ImportDestination": {
                "SuppressionListDestination": {
                    "SuppressionListImportAction": "DELETE"
                }
            },
            "JobStatus": "COMPLETED",
            "JobId": "076683bd-a7ee-4a40-9754-4ad1161ba8b6"
        },
        {
            "CreatedTimestamp": "2020-08-05T16:45:18Z",
            "ImportDestination": {
                "SuppressionListDestination": {
                    "SuppressionListImportAction": "PUT"
                }
            },
            "JobStatus": "COMPLETED",
            "JobId": "6e261869-bd30-4b33-b1f2-9e035a83a395"
        }
    ]
}
```


SES コンソールを使用してアカウントのすべてのインポートジョブのリストを表示するには：

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [Configuration] (設定)で、[Suppression list] (サプレッションリスト) を選択します。

3. [Suppression list] (サプレッションリスト) ペインで、[Bulk actions] (一括アクション) タブを選択します。
4. すべてのインポートジョブが、インポートタイプ、ステータス、および日付とともに [Bulk actions] (一括アクション) テーブルに一覧表示されます。
5. ジョブの詳細を表示するには、ジョブ ID を選択すると、次のペインが表示されます。

    a. 一括アクション: ジョブの全体的なステータス、完了した日時、インポートされたレコード数、およびインポートに失敗したレコード数を示します。

    b. 一括アクションの詳細: は、ジョブ ID、アドレスの追加または削除に使用されたかどうか、ファイル形式が JSONか であったかどうかCSV、一括ファイルが保存された Amazon S3 バケットURIの 、一括アクションが作成された日時を示します。

アカウントのインポートジョブに関する情報を取得する

Amazon SESAPIv2 の GetImportJobオペレーションを使用して、アカウントのインポートジョブに関する情報を取得できます。

> **Note**
>
> 次の手順では、 AWS CLIがインストール済みであるものとします。のインストールと設定の詳細については AWS CLI、「 AWS Command Line Interface ユーザーガイド」を参照してください。

アカウントのインポートジョブに関する情報を取得するには

- コマンドラインで以下のコマンドを入力します。

```
aws sesv2 get-import-job --job-id JobId
```

前述のコマンドは、アカウントのインポートジョブに関する情報を返します。出力は以下の例のようになります。

```
{
    "ImportDataSource": {
        "S3Url": "s3://bucket/object",
        "DataFormat": "CSV"
```

```
    },
    "ProcessedRecordsCount": 2,
    "FailureInfo": {
        "FailedRecordsS3Url": "s3presignedurl"
    },
    "JobStatus": "COMPLETED",
    "JobId": "jobid",
    "CreatedTimestamp": "2020-08-12T17:05:15Z",
    "FailedRecordsCount": 1,
    "ImportDestination": {
        "SuppressionListDestination": {
            "SuppressionListImportAction": "PUT"
        }
    },
    "CompletedTimestamp": "2020-08-12T17:06:42Z"
}
```

SES コンソールを使用してアカウントのインポートジョブに関する情報を取得するには：

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [Configuration] (設定)で、[Suppression list] (サプレッションリスト) を選択します。
3. [Suppression list] (サプレッションリスト) ペインで、[Bulk actions] (一括アクション) タブを選択します。
4. すべてのインポートジョブが、インポートタイプ、ステータス、および日付とともに [Bulk actions] (一括アクション) テーブルに一覧表示されます。
5. ジョブの詳細を表示するには、ジョブ ID を選択すると、次のペインが表示されます。

    a. 一括アクション: ジョブの全体的なステータス、完了した日時、インポートされたレコード数、およびインポートに失敗したレコード数を示します。

    b. 一括アクションの詳細: は、ジョブ ID、アドレスの追加または削除に使用されたかどうか、ファイル形式が JSONまたは であったかどうかCSV、一括ファイルが保存された Amazon S3 バケットURIの 、一括アクションが作成された日時を示します。

Amazon SESアカウントレベルのサプレッションリストの無効化

v2 SES API の PutAccountSuppressionAttributesオペレーションを使用して、 `suppressed-reasons` 属性から値を削除することで、アカウントレベルのサプレッションリストを効果的に無効にできます。

> **Note**
>
> 次の手順では、 AWS CLIがインストール済みであるものとします。のインストールと設定の詳細については AWS CLI、「 AWS Command Line Interface ユーザーガイド」を参照してください。

を使用してアカウントレベルのサプレッションリストを無効にするには AWS CLI

- コマンドラインで以下のコマンドを入力します。

```
aws sesv2 put-account-suppression-attributes --suppressed-reasons
```

SES コンソールを使用してアカウントレベルのサプレッションリストを無効にするには：

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [Configuration] (設定)で、[Suppression list] (サプレッションリスト) を選択します。
3. [Account-level settings] (アカウントレベルの設定) ペインで [Edit] (編集) を選択します。
4. [Suppression list] (サプレッションリスト) で、[Enabled] (有効) ボックスのチェックを外します。
5. [Save changes] (変更の保存) をクリックします。

設定セットレベルの抑制を使用してアカウントレベルのサプレッションリストを上書きする

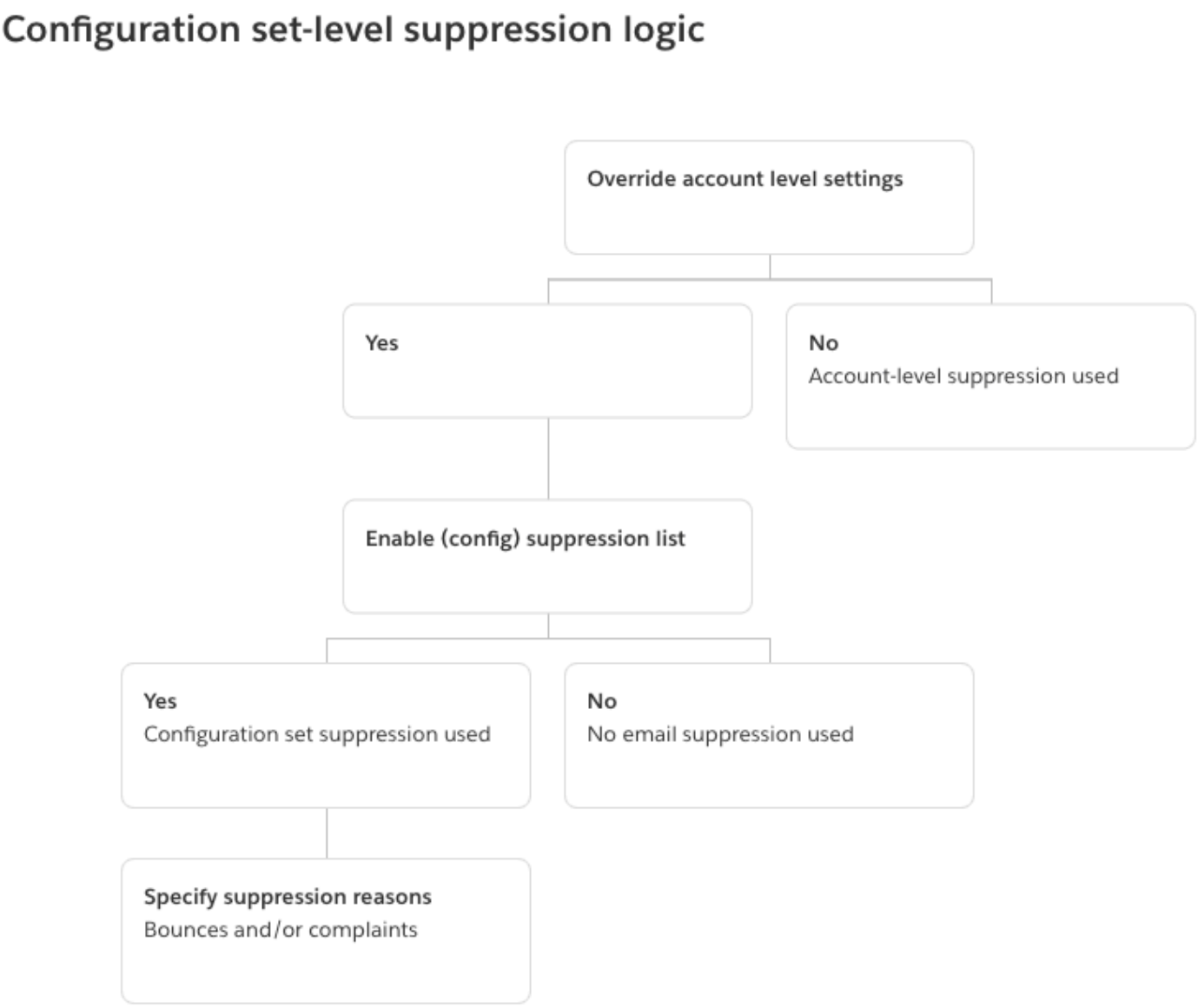
アカウント全体に対してアカウントレベルのサプレッションリストが設定されている場合、設定セットレベルのサプレッションリストを上書きすることで、異なる設定セットに対して個別にカスタマイ

ズできます。このきめ細かさにより、独自の設定セットに割り当てた異なる E メール送信グループに対して、カスタマイズされた抑制設定を使用できます。例えば、アカウントレベルのサプレッションリストが、バウンスアドレスと苦情アドレスの両方が追加されるように設定されているが、設定セットで定義された特定の E メール属性があり、追加される苦情アドレスのみに関心があるとします。この場合は、設定セットの抑制の上書きを有効にして、(アカウントレベルのサプレッションリストで設定されているバウンスや苦情ではなく) この設定セットで送信された E メールからの苦情に対してのみ、アカウントレベルのサプレッションリストに E メールアドレスが追加されるようにします。

設定セットレベルの抑制では、抑制をまったく使用しないなど、さまざまなレベルで、アカウントレベルの抑制を上書きします。次のコンソール手順で設定できるこれらのさまざまな抑制レベルを理解するために、次のリレーションシップマップでは、さまざまなレベルの上書きを有効または無効にできる選択肢の決定セットをモデル化します。これらの組み合わせに応じて、3 つのレベルの抑制の実装に使用できます。

- No overrides (上書きなし) (デフォルト) – 設定セットでは、アカウントレベルのサプレッションリスト設定が使用されます。
- Override account level settings (アカウントレベルの設定を上書きする) – これにより、アカウントレベルのサプレッションリストの設定が無効になります。この設定セットで送信された E メールでは、抑制設定はまったく使用されません。
- Override account level settings with configuration set-level suppression enabled (設定セットレベルの抑制を有効にしたアカウントレベルの設定を上書きする) – この設定セットで送信された E メールでは、その設定に対して有効にした抑制条件 (バウンス、苦情、またはバウンスと苦情) のみが使用されます。アカウントレベルのサプレッションリストの設定に関わらず、設定は上書きされます。

> **Note**
>
> 設定セットレベルで指定されていない抑制条件の場合、アカウントレベルの設定が上書きされたため、抑制動作はグローバル抑制リストにフォールバックします。



設定セットレベルの抑制は実際のサプレッションリストではないことに注意してください。これは、この設定セットで定義されたカスタム抑制設定を使用して、アカウントレベルの抑制を上書きするメカニズムにすぎません。この設定セットを使用して送信される E メールは、独自の抑制設定のみを使用することになり、アカウントレベルの抑制設定は無視されます。つまり、設定セットレベルの抑制は、アカウントレベルのサプレッションリストに追加される E メールアドレスを決定する抑制理由を変更 (上書き) するだけで、アカウントレベルのサプレッションリストと相互に作用します。

設定セットレベルのサプレッションの有効化

Amazon SESの新しいコンソールを使用して設定セットレベルの抑制を有効にするには：

1. にサインイン AWS Management Console し、 で Amazon SESコンソールを開きますhttps://console.aws.amazon.com/ses/。
2. ナビゲーションペインの [Configuration] (設定)で、[Configuration sets] (設定セット) を選択します。
3. [Configuration sets] (設定セット) で、カスタマイズサプレッションで設定する設定セットの名前を選択します。
4. [Suppression list options] (サプレッションリストオプション) ペインで、[Edit] (編集) を選択します。
5. [Suppression list options] (サプレッションリスト) のオプションセクションには、この設定セットによりアカウントレベルの抑制を上書きするオプションなどを含む決定セットが用意されています。設定セットレベルのサプレッションロジックマップによって、オーバーライドが組み合わさった結果が、どのように影響するのかが分かります。これらの多層オーバーライドを選択し組み合わせることで、3 つの異なるレベルの抑制を実装できます。

   a. アカウントレベルのサプレッションを使用してください。アカウントレベルの抑制を上書きしたり、設定セットレベルの抑制を実装したりしないでください。基本的に、この設定セットを使用して送信される E メールは、アカウントレベルの抑制のみを使用します。これを実行するには:

      - [Suppression list settings] (サプレッションリストの設定) で、[Override account level settings] (アカウントレベルの設定を上書きする) ボックスのチェックを外します。

   b. 抑制は使用しないでください。設定セットレベルの抑制は有効にせずに、アカウントレベルの抑制を上書きします。この設定セットを使用して送信される E メールは、アカウントレベルの抑制を使用しないことになります。言い換えると、すべての抑制がキャンセルされます。これを実行するには:

      i. [Suppression list settings] (サプレッションリストの設定) で、[Override account level settings] (アカウントレベルの設定を上書きする) ボックスにチェックを付けます。

      ii. [Suppression list] (サプレッションリスト) で、[Enabled] (有効) ボックスのチェックを外します。

   c. 設定セットレベルの抑制を使用してください。この設定セットで定義されたカスタム抑制設定を使用して、アカウントレベルの抑制を上書きします。この設定セットを使用して送信さ

れる E メールは、独自の抑制設定のみを使用することになり、アカウントレベルの抑制設定は無視されます。これを実行するには:

i. [Suppression list settings] (サプレッションリストの設定) で、[Override account level settings] (アカウントレベルの設定を上書きする) ボックスにチェックを付けます。

ii. [Suppression list] (サプレッションリスト) で [Enabled] (有効) にチェックを入れます。

iii. [Specify the reason(s)...] (理由を指定) で、この設定セットで使用する抑制の理由を 1 つ選択します。

6. [Save changes] (変更の保存) をクリックします。

リスト管理の使用

Amazon SES にはリスト管理機能があります。つまり、カスタマーは連絡先リストと呼ばれる独自のメーリングリストを管理できます。連絡先リストは、特定の 1 つ以上のトピックを購読したすべての連絡先を保存できるリストです。連絡先は、E メールを受信しているエンドユーザーです。トピックは、リスト内の関心グループ、テーマ、またはラベルです。リストには、複数のトピックを含めることができます。

Amazon SES API v2の `ListContacts` オペレーションを使用して、特定のトピックを購読している連絡先のリストを取得し、`SendEmail` オペレーションを使用して、 連絡先に E メールを送信することができます。

サブスクリプション管理の詳細については、「サブスクリプションの使用」を参照してください。

リスト管理の概要

リスト管理を使用する場合は、次の要素を考慮する必要があります。

- リストの作成中にリストトピックを指定できます。
- AWS アカウント ごとに許可される連絡先リストは 1 つだけです。
- リストには、最大 20 個のトピックを設定できます。
- 既存の連絡先リストを更新できます。これには、リストへの新しいトピックの追加、リストからの連絡先の追加と削除、リストまたはトピックの連絡先設定の更新が含まれます。
- トピックの表示名や説明などのトピックメタデータを更新できます。
- 連絡先リストの連絡先、トピックを購読した連絡先、1つのトピックの購読を取り消した連絡先、およびリスト内のすべてのトピックの購読を取り消した連絡先のリストを取得できます。

- 既存の連絡先リストを Amazon SES にインポートするには、CreateImportJob APIを使用します。
- Amazon SES は、お客様の連絡先リストに登録されていない連絡先にメールを送信した場合、E メールをバウンスします。詳細については、「サブスクリプションの使用」を参照してください。
- 各連絡先には、その連絡先に関する情報を格納するための属性を関連付けることができます。

リスト管理の設定

次の操作を使用して、リスト管理機能を設定できます。連絡先リストおよび連絡先操作の完全なリストは、Amazon SES API v2 リファレンスを参照してください。

連絡先リストを作成する

Amazon SES API v2 の CreateContactList オペレーションを使用して、連絡先リストを作成できます。AWS CLI を使用すると、この設定をすばやく簡単に設定できます。AWS CLI のインストールおよび設定の詳細については、「AWS Command Line Interface ユーザーガイド」を参照してください。

AWS CLI を使用してコンタクトリストを作成するには

- コマンドラインで以下のコマンドを入力します。

```
aws sesv2 create-contact-list --cli-input-json file://CONTACT-LIST-JSON
```

上記のコマンドで、*CONTACT-LIST-JSON* を CreateContactList リクエストの JSON ファイルへのパスに置き換えます。

リクエストのための CreateContactList input JSON ファイルの例を次に示します。

```
{
    "ContactListName": "ExampleContactListName",
    "Description": "Creating a contact list example",
    "Topics": [
     {
         "TopicName": "Sports",
         "DisplayName": "Sports Newsletter",
         "Description": "Sign up for our free newsletter to receive updates on all
 sports.",
         "DefaultSubscriptionStatus": "OPT_OUT"
```

```
    },
    {
        "TopicName": "Cycling",
        "DisplayName": "Cycling newsletter",
        "Description": "Never miss a cycling update by subscribing to our
 newsletter.",
        "DefaultSubscriptionStatus": "OPT_IN"
    },
    {
        "TopicName": "NewProducts",
        "DisplayName": "New products",
        "Description": "Hear about new products by subscribing to this mailing
 list.",
        "DefaultSubscriptionStatus": "OPT_IN"
    },
    {
        "TopicName": "DailyUpdates",
        "DisplayName": "Daily updates",
        "Description": "Start your day with sport updates, Monday through
 Friday.",
        "DefaultSubscriptionStatus": "OPT_OUT"
    }
   ]
}
```

連絡先を作成

Amazon SES API v2 の `CreateContact` オペレーションを使用して、連絡先を作成できます。AWS CLI を使用すると、この設定をすばやく簡単に設定できます。AWS CLI のインストールおよび設定の詳細については、「AWS Command Line Interface ユーザーガイド」を参照してください。

AWS CLI を使用して接続を作成するには

- コマンドラインで以下のコマンドを入力します。

  ```
  aws sesv2 create-contact --cli-input-json file://CONTACT-JSON
  ```

  上記のコマンドで、*CONTACT-JSON* を `CreateContact` リクエストの JSON ファイルへのパスに置き換えます。

リクエストのための CreateContact input JSON ファイルの例を次に示します。

```
{
    "ContactListName": "ExampleContactListName",
    "EmailAddress": "example@amazon.com",
    "UnsubscribeAll": false,
    "TopicPreferences": [
        {
            "TopicName": "Sports",
            "SubscriptionStatus": "OPT_IN"
        }
    ],
    "AttributesData": "{\"Name\": \"John\", \"Location\": \"Seattle\"}"
}
```

上の例で、false の UnsubscribeAll 値は、連絡先がすべてのトピックの購読を解除していないことを示し、true の値は、連絡先がすべてのトピックの購読を解除したことを意味します。

TopicPreferences には、連絡先のトピックへのサブスクリプションステータスに関する情報が含まれています。前述の例では、連絡先は「スポーツ」トピックにオプトインしており、「スポーツ」トピックへのすべての E メールを受信します。

AttributesData は JSON フィールドで、連絡先に関するメタデータを格納できます。有効な JSON オブジェクトである必要があります。

連絡先リストへの連絡先の一括インポート

アドレスを手動で一括追加するには、最初に、連絡先をAmazon S3 オブジェクトにアップロードしてから、Amazon SES API v2の CreateImportJob オペレーションまたは SES コンソールを使用します。詳細については、「アカウントレベルのサプレッションリストに E メールアドレスを一括で追加」を参照してください。

連絡先をインポートする前に、連絡先リストを作成する必要があります。

> **Note**
>
> ImportJob ごとに、最大 100 万件の連絡先を連絡先リストに追加できます。

連絡先リストに連絡先を一括で追加するには、次の手順を実行します。

- 連絡先を CSV 形式または JSON 形式で Amazon S3 オブジェクトにアップロードします。

  CSV 形式

  Amazon S3 にアップロードされるファイルの最初の行は、ヘッダー行である必要があります。

  `topicPreferences` オブジェクトは、CSV 形式用にフラット化する必要があります。`topicPreferences` のすべてのトピックには、個別のヘッダーフィールドがあります。

  連絡先リストに一括で連絡先を追加するための CSV 形式の例：

```
emailAddress,unsubscribeAll,attributesData,topicPreferences.Sports,topicPreferences.Cycling
example1@amazon.com,false,{"Name": "John"},OPT_IN,OPT_OUT
example2@amazon.com,true,,OPT_OUT,OPT_OUT
```

  JSON 形式

  newline-delimited JSON ファイルのみがサポートされています。この形式で、各行は 1 つの連絡先情報を含む完全な JSON オブジェクトです。

  連絡先リストに一括で連絡先を追加するための JSON 形式の例：

```
{
     "emailAddress": "example1@amazon.com",
     "unsubscribeAll": false,
     "attributesData": "{\"Name\":\"John\"}",
     "topicPreferences": [
      {
          "topicName": "Sports",
          "subscriptionStatus": "OPT_IN"
      },
      {
          "topicName": "Cycling",
          "subscriptionStatus": "OPT_OUT"
      }
     ]
}
{
```

```
    "emailAddress": "example2@amazon.com",
    "unsubscribeAll": true,
    "topicPreferences": [
     {
         "topicName": "Sports",
         "subscriptionStatus": "OPT_OUT"
     },
     {
         "topicName": "Cycling",
         "subscriptionStatus": "OPT_OUT"
     }
    ]
}
```

前述の例では*example1@amazon.com*および*example2@amazon.com*を、連絡先リストに追加する E メールアドレスに置き換えます。attributesData の値を、連絡先の特定の値に置き換えます。また、*####*および*######*を、連絡先に適用する topicName に置き換えます。使用可能な topicPreferences は *OPT_IN* および *OPT_OUT* です。

連絡先を CSV 形式または JSON 形式で Amazon S3 オブジェクトにアップロードする場合、次の属性がサポートされます。

属性	説明
emailAddress	連絡先の E メールアドレスです。これは必須フィールドです。
unsubscribeAll	連絡先がすべての連絡先リストのトピックの購読を解除しているかどうかを通知するブール値のステータス。
topicPreferences	トピックへのオプトインまたはオプトアウトに関する連絡先の設定。
attributesData	連絡先に添付された属性データ。

- Amazon S3 オブジェクトを読み取るアクセス許可を Amazon SES に付与します。

Amazon S3 バケットに適用すると、Amazon SES にそのバケットを読み込む許可を与えます。Amazon S3 バケットの添付ポリシーの詳細については、Amazon Simple Storage Service ユーザーガイド の「バケットポリシーとユーザーポリシーの使用 」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSESGet",
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::BUCKET-NAME/OBJECT-NAME",
            "Condition": {
                "StringEquals": {
                    "aws:Referer": "AWSACCOUNTID"
                }
            }
        }
    ]
}
```

- AWS KMS キーを使用する許可を Amazon SES に与える

  Amazon S3 オブジェクトが AWS KMS キーを暗号化する場合は、KMS キーを使用するための許可を Amazon SES に付与する必要があります。Amazon SES は、カスタマー管理キーからのみ許可を得ることができ、デフォルトの KMS キーからは許可を得られません。キーのポリシーにステートメントを追加することで、カスタマー管理キーを使用するための許可を Amazon SES に付与する必要があります。

  Amazon SES がカスタマー管理キーを使用することを許可するために Amazon SES に次のポリシーステートメントを貼り付けます。

```
{
   "Sid": "AllowSESToDecrypt",
   "Effect": "Allow",
   "Principal": {
       "Service":"ses.amazonaws.com"
   },
```

```
    "Action": [
        "kms:Decrypt",
    ],
    "Resource": "*"
}
```

- CreateImportJob オペレーションを Amazon SES API v2 で使用します。

> **Note**
>
> 次の例では、AWS CLI がインストール済みであるものとします。AWS CLI のインストールおよび設定の詳細については、AWS Command Line Interface ユーザーガイドを参照してください。

コマンドラインで、以下のコマンドを入力します。*s3####* をAmazon S3 バケットの名前と置き換え、*s3object* をAmazon S3 オブジェクト名に置き換えます。

```
aws sesv2 create-import-job --import-destination
 ContactListDestination={ContactListName=ExampleContactListName,ContactListImportAction=PUT}
 --import-data-source S3Url="s3://s3bucket/s3object",DataFormat=CSV
```

例によるリスト管理のウォークスルー

次のウォークスルーでは、リスト管理を使用して連絡先を一覧表示し、ListManagementOptions を使用して E メールで連絡先リストとトピック名を指定する方法、およびサブスクリプションの解除リンクを挿入する方法の例を示します。

1. AWS CLI を使用して連絡先を一覧表示する – ListContacts オペレーションを使用して、特定のトピックを購読している連絡先リストを取得すると共に、SendEmail オペレーションで、リストメンバーに E メールを送信できます。

   コマンドラインで以下のコマンドを入力します。

   ```
   aws sesv2 list-contacts --cli-input-json file://LIST-CONTACTS-JSON
   ```

   上記のコマンドで、*LIST-CONTACTS-JSON* を ListContacts リクエストの JSON ファイルへのパスに置き換えます。

リクエストのための ListContacts input JSON ファイルの例を次に示します。

```
{
    "ContactListName": "ExampleContactListName",
    "Filter": {
        "FilteredStatus": "OPT_IN",
        "TopicFilter": {
            "TopicName": "Cycling",
            "UseDefaultIfPreferenceUnavailable": true
        }
    },
    "PageSize": 50
}
```

FilteredStatus には、フィルタリングする購読ステータス (OPT_IN または OPT_OUT) が表示されます。

TopicFilter は、結果が欲しいトピックを指定するオプションのフィルターです。上記の例では「Cycling」です。

UseDefaultIfPreferenceUnavailable は、true または false の値を持つことができます。true の場合、連絡先にトピックに対する明示的な設定がなければ、トピックのデフォルト設定が使用されます。false の場合、明示的に定めた設定を持つ連絡先のみがフィルタリングの対象となります。

2. **ListManagementOptions を有効にしてメールを送信** — 上記の ListContactsオペレーションを使用してリスト内の連絡先を一覧表示した後、ListManagementOptions ヘッダーを利用して、連絡先リストとトピック名を指定し、SendEmail オペレーションを利用して各連絡先にメールを送信できます。

   SendEmail オペレーションで ListManagementOptions を使用するには、メールが属している contactListName と topicName を含めます (topicName はオプションです):

```
ListManagementOptions:
    String contactListName
    String topicName
```

連絡先リストにない受信者の E メールアドレスへの SendEmail リクエストに ListManagementOptions を含めると、連絡先がリストに自動的に作成されます。

連絡先リストにある登録解除済み連絡先にメールを送信した場合、Amazon SES は E メールをバウンスします。つまり、登録解除した連絡先に送信されないようにするために SendEmail リクエストを更新する必要はありません。

3. サブスクリプション解除のリンクの場所を示す — ListManagementOptions を利用する場合、Amazon SES を有効にし、{{amazonSESUnsubscribeUrl}} プレースホルダを使用して、SES でサブスクリプションの解除 URL を挿入する必要がある場所を指定して、E メールにサブスクリプションの解除フッターリンクを追加できるオプションがあります。プレースホルダの置き換えは、HTML および TEXT コンテンツタイプでのみサポートされます。プレースホルダは、最大 2 倍含めることができます。2 回以上使用した場合、最初の 2 つのオカレンスのみが置き換えられます。詳細については、「サブスクリプションの使用」を参照してください。

   または、SMTPインターフェイスで E メールを送信しながら、X-SES-LIST-MANAGEMENT-OPTIONS ヘッダーを使用してリストとトピック名を指定します。

   SMTP インターフェイスを使用して E メールを送信するときにリストとトピック名を指定するには、次の E メールヘッダーをメッセージに追加します。

   ```
   X-SES-LIST-MANAGEMENT-OPTIONS: {contactListName}; topic={topicName}
   ```

サブスクリプションの使用

Amazon SES には購読管理機能があります。この機能では、SendEmail オペレーションリクエストの ListManagementOptions 内で contactListName と topicName を指定すると、Amazon SES はすべての送信 E メールで購読取り消しリンクを自動的に有効にします。

連絡先が特定のトピックまたはリストから登録を解除した場合、Amazon SES はそのトピックまたはリストに関連して連絡先に E メールを送信することを許可しません。

> **Note**
>
> - Amazon SES サブスクリプション管理は、多くの E メールサービスプロバイダーが適用している一括送信者の要件をサポートしています。詳細については、「バルク送信者についての変更の概要」の「セクション 2」を参照してください。

- Amazon DKIMで簡単 SES を使用しているユーザーは購読管理を利用できますが、Amazon SES は、Amazon SES を呼び出す前に E メールに署名している送信者に対して、購読取り消しリンクを E メールに追加することはできません。

リスト管理と、特定のトピックを購読している連絡先リストを取得するなど、リストの管理の詳細については、「リスト管理の使用」を参照してください。

購読管理の概要

購読管理を使用する場合は、次の要素を考慮する必要があります。

- 購読管理は Amazon SES によってすべて管理されます。つまり、Amazon SES は、購読取り消しのウェブページから購読取り消しのメールとリクエストを受信し、リストの連絡先の設定を更新します。設定セット通知を使用して、購読取り消し通知を受信できます。設定セットの詳細については、「Amazon SES の設定セットの使用」を参照してください。
- E メール送信時に連絡先リストを指定する必要があります。`List-Unsubscribe` ヘッダーと `ListManagementOptions` フッターのリンクによる購読管理は、それに応じて処理されます。
- Amazon SES では、`List-Unsubscribe` ヘッダー標準がサポートされます。これにより、サポートされている場合、E メールクライアントおよび受信トレイプロバイダーは、E メールの上部に購読取り消しリンクを表示できるようになります。これらのヘッダーをサポートしていない E メールサービスプロバイダーもあります。
- `List-Unsubscribe` ヘッダーは、次の動作に従います。
  - 連絡先リストとトピックの両方が指定されている E メールの サブスクリプション解除のリンクをクリックすると、連絡先は特定トピックからのみの購読が解除されます。
  - トピックが指定されていない場合、連絡先はリスト内のすべてのトピックから配信停止になります。
- 連絡先が、メールフッター内の購読取り消しリンクをクリックすると、購読取り消しのランディングページに移動します。
- 購読取り消しのランディングページでは、連絡先の設定を更新するオプションが表示されます。特定のリスト内のすべてのトピックについて `OPT_IN` または `OPT_OUT` を設定します。ランディングページには、リスト内のすべてのトピックの購読を解除するオプションもあります。
- `ListManagementOptions` を使用する場合、Amazon SES で購読取り消し URL を挿入するべき場所を示す `{{amazonSESUnsubscribeUrl}}` プレースホルダを E メールに含める必要があります。プレースホルダは、最大 2 倍含めることができます。2 回以上使用した場合、最初の 2 つのオカレンスのみが置き換えられます。

- List-Unsubscribe ヘッダーと ListManagementOptions フッターリンクは、E メールが単一の受信者に送信されている場合にのみ追加されます。
- 取引先担当者が登録を解除できないようにするメールの場合、SendEmail リクエストで ListManagementOptions フィールドを排除できます。

購読取り消しヘッダーの考慮事項

購読解除リンクによるサブスクリプション管理は、E メールに次のヘッダーが含まれている場合に有効になります。

List-Unsubscribe

List-Unsubscribe-Post

Amazon SES のサブスクリプション管理 ListManagementOptions を使用する場合、Amazon SES では、これらのヘッダーが E メールに存在する場合、これらのヘッダーをオーバーライドします。

List-Unsubscribe や List-Unsubscribe-Postヘッダーを認識しないプロバイダーもあるため、そのようなプロバイダーを使用して受信者に送信される E メールには、サブスクリプション解除のリンクが表示されないため、これらのヘッダーにより生成されたリンクをクリックしてサブスクリプション解除した受信者は、E メールクライアントまたは受信トレイのプロバイダーによって経験が異なることになります。

受信者の E メールクライアントがこれらのヘッダーを認識する場合、サブスクリプションの解除リンクが表示され、リンクを介してサブスクリプションを解除できますが、どのトピックからサブスクリプションを解除するかを選択することはできず、E メールが送信されたトピックからサブスクリプションが解除されます。

List-Unsubscribe ヘッダーの詳細については、RFC 2369 を参照してください。また、List-Unsubscribe-Postヘッダーについては、RFC 8058 を参照してください。

> **Note**
>
> Amazon SES は、多くの E メールサービスプロバイダーが適用している一括送信者の要件に従って、ワンクリックのサブスクライブ解除をサポートしています。詳細については、「Amazon SES を使用したワンクリックでのサブスクライブ解除の使用」を参照してください。

購読取り消しフッターリンクの追加

テンプレートおよびテンプレート無しの E メールで {{amazonSESUnsubscribeUrl}} プレースホルダを使用して、Amazon SES が購読取り消し URL を挿入する必要がある場所を指定します。

プレースホルダの置き換えは、HTML および TEXT コンテンツタイプでのみサポートされます。

プレースホルダは、最大 2 倍含めることができます。2 回以上使用した場合、最初の 2 つのオカレンスのみが置き換えられます。

> **Note**
>
> {{amazonSESUnsubscribeUrl}}プレースホルダは、SendEmail オペレーションを使用しているときに ListManagementOptions がヘッダーとして指定された場合、または SMTPインターフェイスを使用しているときに X-SES-LIST-MANAGEMENT-OPTIONS をヘッダーとして指定した場合のみ使用できます (ListManagementOptions に依存せず、単独で使用できる、List-Unsubscribe または List-Unsubscribe-Post ヘッダーと混同しないようにご注意ください)。

Amazon SES 送信アクティビティのモニタリング

Amazon SES には、イベント、メトリクスおよび統計を使用して送信アクティビティをモニタリングするための方法が用意されています。イベントは、メトリクスとして追跡するように指定した送信アクティビティに関連して発生するものです。メトリクスは、統計を生成するモニタリング対象イベントタイプの値を表す時系列のデータポイントのセットを表します。統計とは、指定した期間で集約されたメトリクスデータであり、現在までを含みます。

これらのモニタリング方法は、アカウントのバウンス率、苦情率、拒否率などの重要な指標を追跡するのに役立ちます。バウンス率や苦情率が高すぎると、SES を使用した E メール送信に支障が生じる場合があります。また、これらの方法を使用して、設定セットに関連付けられたイベント公開およびカスタムドメインを利用して、全体的なオープン率とクリックスルー率を特定し、送信するメールに対する顧客エンゲージメント率を測定することもできます - カスタムドメインを設定してオープンとクリックの追跡を処理します を参照してください。

モニタリングを設定する最初のステップは、SES を使用して測定およびモニタリングする送信アクティビティに関連する E メールイベントの種類を特定することです。SES でモニタリングするイベントタイプとして、次を選択できます :

- 送信数— 送信リクエストが成功すると、Amazon SES はそのメッセージを受信者のメールサーバーに配信しようと試行します。(アカウントレベルまたはグローバル抑制が使用されている場合でも、SES により送信済みとしてカウントされますが、配信は抑制されます)。
- レンダリング失敗 – テンプレートのレンダリング問題により、E メールが送信されませんでした。このイベントタイプは、テンプレートデータが見つからない場合や、テンプレートのパラメータとデータが一致しない場合に発生します。(このイベントタイプは、`SendTemplatedEmail` または `SendBulkTemplatedEmail` API オペレーションを使用して E メールを送信する場合にのみ発生します。)
- 拒否 - Amazon SESは E メールを受け取りましたが、この E メールにウイルスが含まれていると判断して拒否したため、受信者のメールサーバーに E メール配信を試みませんでした。
- 配信 - Amazon SES は、受取人のメールサーバーにメールを正常に配信しました。
- バウンス – ハードバウンスにより、受信者のメールサーバーが E メールを完全に拒否しました。(ソフトバウンスは、SES が E メールの配信を再試行しない場合にのみ含まれます。通常、これらのソフトバウンスは配信障害を示しますが、メールが受信者の受信トレイに正常に届いた場合でも、場合によってはソフトバウンスが返されることがあります。これは通常、受信者が不在の自動返信を送信した場合に発生します。ソフトバウンスの詳細については、この AWS re:Post 記事を参照してください)。

- 苦情— Eメールは受信者のメールサーバーに正常に配信されましたが、受信者はスパムとしてマークしました。
- 配信の遅延 – 一時的な問題が発生したため、メールを受信者のメールサーバーに配信できませんでした。配信の遅延は、受信者の受信トレイがいっぱいになった場合や、受信側の電子メールサーバーで一時的な問題が発生した場合などに発生します。
- サブスクリプション— メールは正常に配信されましたが、受信者がEメールヘッダーの `List-Unsubscribe` またはフッターの `Unsubscribe` リンクをクリックし、サブスクリプションの設定を更新しました。
- オープン – 受信者がメッセージを受け取り、E メールクライアントで開きました。
- クリック – 受信者はメール内の 1 つ以上のリンクをクリックしました。

E メール送信イベントはいくつかの方法でモニタリングできます。選択する方法は、モニタリングするイベントタイプ、モニタリングする粒度と詳細レベル、SES でデータを発行する場所によって異なります。バウンスと苦情のイベントを追跡するには、フィードバック通知またはイベント発行のいずれかを使用する必要があります。複数のモニタリング方法の使用を選択することもできます。各方法の特徴を次の表に示します。

モニタリング方法	モニタリング可能なイベント	データへのアクセス方法	詳細レベル	細分性
SES コンソール	アカウントヘルス、送信されたメール、使用された クォータ、成功した送信リクエスト、拒否、バウンス、苦情 (最近の履歴から現在の評価まで)	SES コンソールのアカウントダッシュボードページ	計算と割合	AWS アカウント全体
SES コンソール	アカウントヘルス、送信された E メール、バウ	SES コンソールの評価メトリクスページ	計算された率のみ	AWS アカウント全体

モニタリング方法	モニタリング可能なイベント	データへのアクセス方法	詳細レベル	細分性
	ンスおよび苦情 (現在の評判)			
Virtual Deliverability Manager	アカウント統計、ISP、送信 ID、設定セット、送信済み、配信済み、苦情、一時的バウンスと永続的なバウンス、開封とクリック、配信可能性、評価	SES コンソールの the section called “ダッシュボード” SES コンソールの the section called “アドバイザー”	計算と割合	AWS アカウント全体
SES API	配信、バウンス、苦情、拒否	`GetSendStatistics` API オペレーション	カウントのみ	AWS アカウント全体

モニタリング方法	モニタリング可能なイベント	データへのアクセス方法	詳細レベル	細分性
Amazon CloudWatchコンソール	送信、配信、開封、クリック、バウンス、バウンス率、苦情、苦情率、拒否、レンダリング失敗、およびブラックリストIP。	CloudWatch コンソール **Note** 一部のメトリクスは、関連付けられたイベントが発生するまでCloudWatch に表示されません。たとえば、バウンスのメトリクスは、送信した E メールの少なくとも 1 つがバウンスするまでは、またはメールボックスシミュレーターを使用して、シミュ	カウントのみ	AWS アカウント全体

モニタリング方法	モニタリング可能なイベント	データへのアクセス方法	詳細レベル	細分性
		レートされたバウンスイベントを作成するまでは、CloudWatch に表示されません。。		
フィードバック通知	配信、バウンス、および苦情	Amazon SNS 通知 (配信、バウンス、苦情) または E メール (バウンスと苦情のみ)。「イベント通知の設定」を参照してください。	各イベントの詳細	AWS アカウント全体

モニタリング方法	モニタリング可能なイベント	データへのアクセス方法	詳細レベル	細分性
イベントの発行	送信、配信、オープン、クリック、バウンス、苦情、拒否、レンダリングの失敗。	Amazon CloudWatch または Amazon Data Firehose、または Amazon SNS 通知経由。「イベント発行を使用して E メール送信をモニタリングする」を参照。 (追加料金が適用されます。「CloudWatch のメトリクスあたりの料金」を参照してください。)	各イベントの詳細	細かい (ユーザー定義可能なメールの特性に基づく)
設定セットに関連付けられたカスタムドメインを使用したイベントの発行 - 詳細情報	追跡を開きクリックします。	Amazon CloudWatch、Amazon Data Firehose、または Amazon SNS 通知経由。 (追加料金が適用されます。「CloudWatch のメトリクスあたりの料金」を参照してください。)	各イベントの詳細	細かい (ユーザー定義可能なメールの特性に基づく)

> **Note**
>
> E メール送信イベントで測定したメトリクスは、送信クォータと完全には一致しない場合があります。このような不一致は、E メールのバウンスや拒否、または SES 受信箱シミュレーターの使用が原因で発生する場合があります。どれだけ送信クォータに近付いているかを確認するには、「送信クォータのモニタリング」を参照してください。

各モニタリング方法を使用する方法については、以下のトピックを参照してください。

- Amazon SES コンソールを使用した送信統計情報のモニタリング
- Amazon SES API を使用した使用統計のモニタリング
- Amazon SES イベント発行を使用して E メール送信をモニタリングする

Amazon SES コンソールを使用した送信統計情報のモニタリング

Amazon SES コンソールの [アカウントダッシュボード]、[評価メトリクス]、および [SMTP 設定] ページから、すべての E メール送信、使用状況、統計、SMTP 設定、アカウントの全体的なヘルス、評価メトリクスをモニタリングできます。以下のセクションでは、これらの各コンソールページに表示されるメトリクスと統計について説明します。

the section called “アカウントダッシュボード” および the section called “評価メトリクス” の両方のコンソールページにはバウンスと苦情のメトリクスが含まれていますが、以下に説明するように、これら 2 つのバウンス率と苦情率の間には微妙な違いがあります。

- アカウントダッシュボードページ — 選択した日付範囲に基づいて、過去のバウンス率と苦情率が表示され、現在までの変化のメトリクスの進行状況が示されます。
- 評価メトリクスページ – 高レベルで全体的な過去の平均を計算することで受け取った最新のデータポイントに基づくバウンス率と苦情率 (アカウントダッシュボードページに表示されるリアルタイムの正確なバウンス/苦情イベントに該当する、通常のバウンス/苦情率と混同しないでください)。

評価メトリクスページとアカウントダッシュボードページのバウンス率または苦情率を比較するシンプルな例として、昨日のレートが 2% で、現在は 1% とします。評価メトリクスページには現在のレートである 1% しか表示されませんが、アカウントダッシュボードページのグラフには進行がプロットされ、昨日は 2%、今日は 1% のレートが示されます。

アカウントダッシュボード

アカウントから送信された E メールの数および使用された送信クォータの割合は、SES コンソールのアカウントダッシュボードページの [Daily email usage] (日次 E メール使用量) ペインで直接モニタリングできます。アカウントの配信率と拒否率をモニタリングするには、[Sending Statistics] (送信統計) ペインを使用できます。E メールの送信に関するその他の主な要素も、以下のペインでモニタリングできます。

- Sending limits (送信制限) – SES を通じたメールの送信に適用される次のクォータが含まれています。
  - Daily sending quota (日次送信クォータ) – 24 時間あたりに送信できる E メールの最大数。
  - Maximum send rate (最大送信レート) – 1 秒あたりにアカウントから送信できる E メールの最大数。
- Account health (アカウントのヘルス) – SES アカウントのステータス:
  - `Healthy` – 現在、アカウントに影響するような、評価に関連する問題はありません。
  - `Under review` – SES アカウントで潜在的な問題が特定されました。問題の修正作業中は、アカウントがレビュー中になります。
  - `Paused` – アカウントから送信された E メールに関する問題により、アカウントの E メール送信機能が現在一時停止されています。問題が修正されたら、アカウントの E メール送信機能の再開をリクエストできます。
- Daily email usage (毎日の E メール使用量) – 毎日の使用状況をチェックして、送信制限に近づいていないことを確認します。
  - Emails sent (送信された E メール) – 24 時間以内に送信された E メールの合計数。
  - Remaining sends (残りの送信数) -24 時間に送信できる残りのメールの合計数
  - Sending quota used (使用された送信クォータ) – 毎日の送信クォータの使用割合。
- Sending statistics (送信の統計) – モニタリング対象イベントタイプの値を表す時系列データポイント内の 4 つの重要なメトリクスの進行を示すグラフで構成されます。このグラフは、1 時間の集計期間を使用して、選択した日付範囲の統計を生成します。`Last 1 day` から `Last 14 days` までの開始値を使用してデータ範囲を選択し、以下のチャートをフィルタリングします :
  - Sends (送信数) – 選択した日付範囲で成功した E メール送信リクエストの合計。
  - Rejects (拒否) – 選択した日付範囲の `Rejects/Sends * 100` に基づき、SES によって拒否された送信リクエストの平均レート。
  - Bounces (バウンス) – 選択した日付範囲の進行状況を示す、全体的な過去の送信者評価メトリクスから得られた平均レート。

- Complaints (苦情) – 選択した日付範囲の進行状況を示す、全体的な過去の送信者評価メトリクスから得られた平均レート。

これらの各チャートには、[View in CloudWatch] (CloudWatch で表示する) ボタンがあり、Amazon CloudWatch コンソールでそれぞれのメトリクスを開き、詳細なデータの表示、カスタマイズされたメトリクスの計算の実行、CloudWatch でのアラームの作成 ができます。

評価メトリクス

バウンス率と苦情率に加えて、評価メトリクスページは、他のペインに属する評価に影響を与える主要因について、その他の大まかな可視性も提供します。

- Summary (概要) – 評価のヘルスの概要が示されます。
  - Status (ステータス) – 過去のバウンス率と苦情率に基づく全体的な評価のヘルス:
    - `Healthy` – 両方のメトリクスは通常のレベル内です。
    - `Under review` – 一方または両方のメトリクスにより、自動的にアカウントがレビュー対象になっています。
    - `At risk` – 一方または両方のメトリクスが正常でないレベルに達しており、アカウントの E メール送信機能が危険な状態になっている可能性があります。
  - 送信された E メール (過去 24 時間) – 過去 24 時間に送信された E メールの合計数。
  - 残りの送信数 – 24 時間に送信できる残りのメールの合計数
  - 使用された送信クォータ – 毎日の送信クォータの使用割合。
- アカウントレベルのタブの内容:
  - バウンス率
    - Status (ステータス) – [Summary] (概要) ペインで説明されているのと同じ値を使用して、バウンス率のヘルスを示します。
    - Historic bounce rate (過去のバウンス率) – 一般的な送信方法を表す代表ボリュームに基づいて、全体的な履歴平均から計算され、ハードバウンスが発生したアカウントからの E メールの割合。
  - Complaint rate
    - Status (ステータス) – [Summary] (概要) ペインで説明されている値と同じ値を使用して、苦情率のヘルスを示します。

    - Historic bounce rate (過去のバウンス率) – 一般的な送信方法を表す代表ボリュームに基づいて、全体的な履歴平均から計算された、受信者がスパムとして報告したアカウントからの E メールの割合。
- [Confituration set] (設定セット) タブの内容:
    - 設定セット別の評価
        - Configuration set (設定セット) – 評価メトリクスが有効な設定セットを入力または選択し、選択した設定セットを使用して送信された E メールに基づいて、概要、バウンス、および苦情データを表示できます。設定セットを選択した後に表示される結果のペインの説明は、評価メトリクスページに関する上記の説明と同じですが、アカウントレベルの送信メトリクス全体に適用される選択された設定セットで送信される E メールのみに基づく点が異なります。

SMTP 設定

このページでは、Amazon SES SMTP インターフェイスを SES API またはプログラムで使用するために必要な SMTP 設定を一覧表示し、SMTP 認証情報を作成および管理するためのリンクを提供します。

- SMTP settings (SMTP 設定) – SMTP 対応のプログラミング言語、E メールサーバー、またはアプリケーションを使用して Amazon SES の SMTP インターフェイスに接続する場合は、次の情報が提供されます。
    - SMTP エンドポイント
    - STARTTLS ポート
    - Transport Layer Security (TLS)
    - TLS ラッパー
    - SMTP および IAM 認証情報の作成および管理に提供される認証リンク

コンソールを使用して送信メトリクスと評価メトリクスをモニタリングする

以下の手順では、アカウントダッシュボードページで最近の履歴 (最大 14 日間) に基づくメトリクスを使用するか、評価メトリクスページで現在までの全体的な履歴に基づくメトリクスを使用して、送信メトリクスと評価メトリクスの詳しい調査を開始します。

送信済みEメールおよび送信クォータを表示するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. ナビゲーションペインで、[アカウントダッシュボード] を選択します。使用量の統計は、[Daily email usage] (毎日の E メール使用量) セクションに表示されます。

送信数、拒否率、バウンス率、および苦情率を表示するには

1. ナビゲーションペインで、[アカウントダッシュボード] を選択します。
2. [Sending statistics] (統計を送信する) セクションで、[Date range] (日付範囲) ドロップダウンで日付範囲の開始値を選択し、[Sending statistics] (統計を送信する) セクション直下の 4 つのチャートをフィルタリングします。
3. 選択した日付範囲に基づいて、現在までの変化のメトリクスの進行状況を示す過去のカウントとレートを表示できます。
4. いずれかのチャートで、[View in CloudWatch] (CloudWatch で表示する) ボタンをクリックして、それぞれのメトリクスを Amazon CloudWatch コンソールで開くと、詳細データの表示、カスタマイズされたメトリクス計算の実行、およびCloudWatch でモニタリングアラームを作成することができます。

過去の全体的なバウンス率と苦情率を表示するには

1. 左のナビゲーションペインで [Reputation metrics] (評価のメトリクス) を選択します。
2. [Bounce rate] (バウンス率) ペインでは、アカウントから送信され、ハードバウンスが発生した E メールの割合を表示できます。[Complaint rate] (苦情率) ペインでは、アカウントから送信され、受信者によってスパムと報告された E メールの割合を表示できます。どちらのメトリクスも、E メールの一般的な送信方法に基づいて計算されています。
3. いずれかのペインで、[View in CloudWatch] (CloudWatch で表示する) ボタンを選択して、それぞれのメトリクスを Amazon CloudWatch コンソールで開くと、詳細データの表示、カスタマイズされたメトリクス計算の実行、および CloudWatch でのモニタリングアラームの作成を行うことができます。

設定セットごとに評価メトリクスを表示するには

1. 左のナビゲーションペインで [Reputation metrics] (評価のメトリクス) を選択します。

2. 評価メトリクスページで、[Configuration set] (設定セット) タブを選択します。
3. [Reputation by configuration set] (設定セット別の評価) ペインで、[Configuration set] (設定セット) フィールド内をクリックします。評価メトリクスが有効になっている設定セットの入力を開始するか、選択します。
4. 設定セットを選択すると、[Summary] (概要)、[Bounce] (バウンス)、および [Complaint] (苦情) ペインがロードされ、選択した設定セットと共に送信された E メールのみに基づくメトリクスが表示されます。

Amazon SES API を使用した使用統計のモニタリング

Amazon SES API は、 `GetSendStatistics` オペレーションを使用してサービスの使用に関する情報を返します。送信統計を定期的にチェックし、必要に応じて調整を行うようお勧めします。

`GetSendStatistics` オペレーションを呼び出すと、過去 2 週間の送信アクティビティを示すデータポイントのリストが返されます。このリストの各データポイントは、15 分間のアクティビティを示し、その期間に関する以下の情報が含まれています:

- ハードバウンス数
- 苦情数
- 配信試行回数 (送信した E メールの数に相当します)
- 送信試行の拒否数
- 分析期間のタイムスタンプ

`GetSendStatistics` オペレーションの詳細については、Amazon Simple Email Service API リファレンスを参照してください。

このセクションでは、以下のトピックについて説明します。

- the section called “`GetSendStatistics` を使用した AWS CLI API オペレーションの呼び出し”
- the section called “プログラムを使用した `GetSendStatistics` オペレーションの呼び出し”

GetSendStatistics を使用した AWS CLI API オペレーションの呼び出し

GetSendStatistics API オペレーションを最も簡単に呼び出す方法は AWS Command Line Interface (AWS CLI) を使用することです。

GetSendStatistics を使用して AWS CLI API オペレーションを呼び出すには

1. AWS CLI をインストールします (まだの場合)。詳細については、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interface のインストール」を参照してください。
2. まだ行っていない場合は、AWS認証情報を使用するためにはAWS CLIを設定します。詳細については、『AWS Command Line Interface ユーザーガイド』の「AWS CLI の設定」を参照してください。
3. コマンドラインから、以下のコマンドを実行します。

   ```
   aws ses get-send-statistics
   ```

   AWS CLI を正常に設定すると、送信統計が JSON 形式で一覧表示されます。JSON オブジェクトごとに 15 分間の集約された送信統計が含まれています。

プログラムを使用した GetSendStatistics オペレーションの呼び出し

AWS SDK を使用して GetSendStatistics オペレーションを呼び出すこともできます。このセクションでは、AWS SDK のコード例を Go、PHP、Python および Ruby で示します。以下のいずれかのリンクを選択すると、その言語でコード例が表示されます。

- AWS SDK for Go のコード例
- AWS SDK for PHP のコード例
- AWS SDK for Python (Boto) のコード例
- AWS SDK for Ruby のコード例

> **Note**
>
> これらのコード例では、AWS アクセスキー ID、AWS シークレットアクセスキー、および使用する AWS 地域を含む AWS 共有認証情報ファイルが作成済みであるものとします。詳細については、「共有認証情報ファイルと設定ファイル」を参照してください。

`GetSendStatistics` を使用した AWS SDK for Go の呼び出し

```
package main

import (
    "fmt"

    //go get github.com/aws/aws-sdk-go/...
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/ses"
    "github.com/aws/aws-sdk-go/aws/awserr"
)

const (
    // Replace us-west-2 with the AWS Region you're using for Amazon SES.
    AwsRegion = "us-west-2"
)

func main() {

    // Create a new session and specify an AWS Region.
    sess, err := session.NewSession(&aws.Config{
        Region:aws.String(AwsRegion)},
    )

    // Create an SES client in the session.
    svc := ses.New(sess)
    input := &ses.GetSendStatisticsInput{}

    result, err := svc.GetSendStatistics(input)

    // Display error messages if they occur.
    if err != nil {
        if aerr, ok := err.(awserr.Error); ok {
```

```
            switch aerr.Code() {
            default:
                fmt.Println(aerr.Error())
            }
        } else {
            // Print the error, cast err to awserr.Error to get the Code and
            // Message from an error.
            fmt.Println(err.Error())
        }
        return
    }

    fmt.Println(result)
}
```

GetSendStatistics を使用した AWS SDK for PHP の呼び出し

```
<?php

// Replace path_to_sdk_inclusion with the path to the SDK as described in
// http://docs.aws.amazon.com/aws-sdk-php/v3/guide/getting-started/basic-usage.html
define('REQUIRED_FILE','path_to_sdk_inclusion');

// Replace us-west-2 with the AWS Region you're using for Amazon SES.
define('REGION','us-west-2');

require REQUIRED_FILE;

use Aws\Ses\SesClient;

$client = SesClient::factory(array(
    'version'=> 'latest',
    'region' => REGION
));

try {
     $result = $client->getSendStatistics([]);
  echo($result);
} catch (Exception $e) {
     echo($e->getMessage()."\n");
}

?>
```

`GetSendStatistics` を使用した AWS SDK for Python (Boto) の呼び出し

```
import boto3 #pip install boto3
import json
from botocore.exceptions import ClientError

client = boto3.client('ses')

try:
    response = client.get_send_statistics(
)
except ClientError as e:
    print(e.response['Error']['Message'])
else:
    print(json.dumps(response, indent=4, sort_keys=True, default=str))
```

`GetSendStatistics` を使用した AWS SDK for Ruby の呼び出し

```
require 'aws-sdk' # gem install aws-sdk
require 'json'

# Replace us-west-2 with the AWS Region you're using for Amazon SES.
awsregion = "us-west-2"

# Create a new SES resource and specify a region
ses = Aws::SES::Client.new(region: awsregion)

begin

  resp = ses.get_send_statistics({
  })
  puts JSON.pretty_generate(resp.to_h)

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts error

end
```

Amazon SES イベント発行を使用して E メール送信をモニタリングする

メール送信をきめ細かく追跡できるようにするには、定義した特性に基づいて E メール送信イベントを、Amazon CloudWatch、Amazon Data Firehose、Amazon Pinpoint、Amazon Simple Notification Service、または Amazon EventBridge に発行するように、Amazon SES を設定できます。

送信、配信、オープン、クリック、バウンス、苦情、拒否、レンダリングの失敗、配信遅延など、さまざまな種類の E メール送信イベントを追跡できます。この情報は、運用と分析の目的で役立ちます。例えば、E メール送信データを CloudWatch に公開し、E メールキャンペーンのパフォーマンスを追跡するダッシュボードを作成できます。または、Amazon SNS を使用して、特定のイベントが発生したときに通知を送信できます。

イベント発行が設定セットとメッセージタグと連携する方法

イベント発行を使用するには、まず 1 つ以上の設定セットをセットアップします。設定セットは、イベントの発行先の場所と発行するイベントを指定します。その後、メールを送信するたびに、設定セットの名前と 1 つ以上のメッセージタグを名前と値のペアの形式で指定し、メールを分類します。たとえば、書籍を宣伝する場合、関連するキャンペーンのメールを送信するときに、メッセージタグに genre という名前を付け、値 sci-fi または western を割り当てることができます。

使用する E メール送信インターフェースに応じて、`SendEmail` API オペレーションの `EmailTags` フィールドにパラメータとしてメッセージタグを指定するか、SES 固有の E メールヘッダーである `X-SES-MESSAGE-TAGS` にメッセージタグを追加します。設定セットの詳細については、「Amazon SES の設定セットの使用」を参照してください。

ユーザーが指定するメッセージタグに加え、SES は送信するメッセージに自動タグも追加します。自動タグを使用するために追加のステップを実行する必要はありません。

SES を使用して送信するメッセージに自動的に適用される自動タグの一覧は、次の表のとおりです。

SES 自動タグ

自動タグ名	説明
`ses:caller-identity`	E メールを送信した SES ユーザーの IAM ID

自動タグ名	説明
ses:configuration-set	E メールに関連付けられた設定セットの名前
ses:from-domain	「From」アドレスのドメイン
ses:outgoing-ip	SES が E メールの送信に使用した IP アドレス
ses:source-ip	呼び出し元がメールの送信に使用した IP アドレス
ses:source-tls-version	発信者が E メールの送信に使用した TLS プロトコルバージョン
ses:outgoing-tls-version	SES が E メールの送信に使用した TLS プロトコルバージョン

E メールキャンペーンのきめ細かなフィードバック

ses:feedback-id-<*a* or *b*> タグは、オプションのメッセージのタグであり、ハイブリッドタグまたは半自動タグと考えることができます。前のセクションで説明した自動タグと似ていますが、手動で追加して、プレフィックスキーの ses: を使用するという点で異なります。これらのタグは、ses:feedback-id-a および ses:feedback-id-b として定義して、最大 2 つを使用できます。

これらのタグを指定すると、SES は、フィードバックループ (FBL) の一環として苦情やスパム率などの配信統計を提供するために使用される標準の Feedback-ID ヘッダーにこのようなタグを自動的に追加します。「フィードバックループ」を参照してください。Feedback-ID ヘッダーは、SES が苦情情報を収集するために使用する識別子である SESInternalID と、以下に示すような、SES を送信プラットフォームとして識別する静的タグ AmazonSES で構成されます。

FeedBackId:feedback-id-a:feedback-id-b:((SESInternalID):(AmazonSES))

これらのオプションのフィードバック ID タグは、E メールキャンペーンの一環として送信するメッセージなど、きめ細かいフィードバックを生成する方法として提供されています。次の例で説明されるとおり、SendEmail オペレーションリクエストの EmailTags フィールドにメッセージタグとして指定することで、ses:feedback-id-<*a* or *b*> を使用できます。

```
{
```

```
  "FromEmailAddress": "noreply@example.com",
  "Destination": {
    "ToAddresses": [
      "customer@example.net"
    ]
  },
  "Content": {
    "Simple": {
      "Subject": {
        "Data": "Hello and welcome"
      },
      "Body": {
        "Text": {
          "Data": "Lorem ipsum dolor sit amet."
        },
        "Html": {
          "Data": "Lorem ipsum dolor sit amet."
        }
      }
    }
  },
  "EmailTags": [
    {
      "Name": "ses:feedback-id-a",
      "Value": "new-members-campaign"
    },
    {
      "Name": "ses:feedback-id-b",
      "Value": "football-campaign"
    }
  ],
  "ConfigurationSetName": "football-club"
}
```

raw 形式で送信する場合は、SES 固有のヘッダーである X-SES-MESSAGE-TAGS に ses:feedback-id-<*a* or *b*> をメッセージタグとして追加します。

メッセージタグ ses:feedback-id-<*a* or *b*> は、その他のメッセージタグと同様、CloudWatch 値のソースとして指定すると、Amazon CloudWatch で追跡することもできます。「the section called “CloudWatch Event 送信先の詳細を追加する”」を参照してください (追加料金が発生します。「CloudWatch のメトリクスあたりの料金」を参照してください)。

イベント発行を使用する方法

以下のセクションでは、SES イベント発行のセットアップと使用に必要な情報が記載されています。

- イベント発行のセットアップ
- イベントデータの使用

イベント発行の用語

SES イベント発行に関連する用語の定義は、以下のリストのとおりです。

メール送信イベント

SES に送信した E メールの結果に関連付けられた情報。以下に示しているのは、送信イベントです。

- 送信数— 送信リクエストが成功すると、Amazon SES はそのメッセージを受信者のメールサーバーに配信しようと試行します。(アカウントレベルまたはグローバル抑制が使用されている場合でも、SES により送信済みとしてカウントされますが、配信は抑制されます)。
- レンダリング失敗 – テンプレートのレンダリング問題により、E メールが送信されませんでした。このイベントタイプは、テンプレートデータが見つからない場合や、テンプレートのパラメータとデータが一致しない場合に発生します。(このイベントタイプは、`SendTemplatedEmail` または `SendBulkTemplatedEmail` API オペレーションを使用して E メールを送信する場合にのみ発生します。)
- 拒否 - Amazon SESは E メールを受け取りましたが、この E メールにウイルスが含まれていると判断して拒否したため、受信者のメールサーバーに E メール配信を試みませんでした。
- 配信 - Amazon SES は、受取人のメールサーバーにメールを正常に配信しました。
- バウンス – ハードバウンスにより、受信者のメールサーバーが E メールを完全に拒否しました。(ソフトバウンスは、SES が E メールの配信を再試行しない場合にのみ含まれます。通常、これらのソフトバウンスは配信障害を示しますが、メールが受信者の受信トレイに正常に届いた場合でも、場合によってはソフトバウンスが返されることがあります。これは通常、受信者が不在の自動返信を送信した場合に発生します。ソフトバウンスの詳細については、この AWS re:Post 記事を参照してください)。
- 苦情— Eメールは受信者のメールサーバーに正常に配信されましたが、受信者はスパムとしてマークしました。

- 配信の遅延 – 一時的な問題が発生したため、メールを受信者のメールサーバーに配信できませんでした。配信の遅延は、受信者の受信トレイがいっぱいになった場合や、受信側の電子メールサーバーで一時的な問題が発生した場合などに発生します。
- サブスクリプション— メールは正常に配信されましたが、受信者がEメールヘッダーの `List-Unsubscribe` またはフッターの `Unsubscribe` リンクをクリックし、サブスクリプションの設定を更新しました。
- オープン – 受信者がメッセージを受け取り、E メールクライアントで開きました。
- クリック – 受信者はメール内の 1 つ以上のリンクをクリックしました。

設定セット

SES が発行する E メール送信イベントの送信先と、発行する E メール送信イベントのタイプを定義するルールセット。イベント発行で使用するメールを送信するとき、メールに関連付ける設定セットを指定します。

イベント送信先

SES の E メール送信イベントを発行する AWS サービス。セットアップする各イベントの宛先は、設定セット 1 つだけに帰属しています。

メッセージタグ

イベント発行の目的でメールの分類に使用する名前と値のペア。たとえば、campaign/book や campaign/clothing などです。メッセージタグは、メールを送信する際に、API コールへのパラメータとして、または SES 固有のメールヘッダーとして指定します。

自動タグ

イベント発行レポートに自動的に含まれるメッセージタグ。設定セット名、「送信元」アドレスのドメイン、発信者の送信 IP アドレス、SES 送信 IP アドレス、発信者の IAM ID 向けの自動タグがあります。

Amazon SES でのイベント発行のセットアップ

このセクションでは、次のAWSサービスに E メール送信イベントを発行するよう、Amazon SES を設定するための必要事項について説明します。

- Amazon CloudWatch
- Amazon Data Firehose
- Amazon Pinpoint

- Amazon Simple Notification Service (Amazon SNS)

イベント公開の設定に必要な次の手順については、以下のトピックで説明します。

1. 最初に、Amazon SES コンソールまたは API を使用して、作成する設定セット を作成する必要があります。
2. 設定セットに単数または複数のイベント送信先 (CloudWatch、Firehose、Pinpoint、または SNS) を追加して、そのイベント送信先に一意のパラメータを設定します。
3. E メールを送信するときに、イベントの送信先を含めるために使用する設定セットを指定します。

このセクションのトピック

ステップ 1: 設定セットを作成する

イベント公開を設定するには、まず設定セットが必要です。まだ設定セットをお持ちでない場合、または新しい設定セットを作成したい場合は、SES での設定セットの作成を参照してください。

Amazon SES API V2 のCreateConfigurationSet オペレーションまたはAmazon SES CLI V2を使用して、設定セットを作成する方法については、設定セット (AWS CLI) を作成します。を参照してください。

ステップ 2: イベント送信先を追加する

イベントの送信先は、Amazon SES イベントを発行する場所です。セットアップする各イベントの宛先は、設定セット 1 つだけに帰属しています。Amazon SES を使用してイベント送信先をセットアップする場合、送信先のAWSのサービスを選択し、その送信先に関連付けられたパラメータを指定します。

イベントの送信先を設定する場合、次のいずれかの AWSサービスを使用して、イベントを送信するように選択できます。

- Amazon CloudWatch
- Amazon Data Firehose

- Amazon EventBridge
- Amazon Pinpoint
- Amazon Simple Notification Service (Amazon SNS)

どのイベント送信先を選択するかは、イベントに関して必要な詳細のレベルとイベント情報を受け取る方法によって異なります。各タイプのイベントの合計数のみ必要な場合は (たとえば、合計数が高くなりすぎた場合にアラームを設定できるようにするなど)、CloudWatch を使用します。

分析のために Amazon OpenSearch Service や Amazon Redshift などの別のサービスに出力できる詳細なイベントレコードが必要な場合は、Firehose を使用します。

特定のイベントが発生したときに通知を受け取る場合は、Amazon SNS を選択します。

このセクションは、以下のトピックで構成されます。

- イベント発行の CloudWatch Event 送信先のセットアップ
- Amazon SES イベント発行向けに Data Firehose イベント送信先を設定する
- イベント発行のために Amazon EventBridge イベント送信先を設定する
- イベント発行の Amazon Pinpoint イベント送信先のセットアップ
- イベント発行の Amazon SNS イベント送信先のセットアップ

イベント発行の CloudWatch Event 送信先のセットアップ

Amazon CloudWatch メトリクスでは、イベント送信先を使用して、Amazon SES E メール送信イベントを CloudWatch に発行することができます。CloudWatch イベント送信先をセットアップできるのは、設定セット内でのみであるため、まず設定セットを作成してから、その設定セットにイベント送信先を追加する必要があります。

CloudWatch Event 送信先を設定セットに追加したら、E メールの送信時に使用するメッセージタグに対応する CloudWatch ディメンションを 1 つ以上選択する必要があります。メッセージタグと同様、CloudWatch ディメンションは、メトリクスを一意に識別できるようにする名前と値のペアです。

たとえば、メッセージタグと、メールキャンペーンの識別に使用する `campaign` と呼ばれるディメンションを選択できます。メール送信イベントを CloudWatch に発行するとき、メッセージタグとディメンションを選択することが重要です。その選択内容によって、CloudWatch の請求に影響が及び、CloudWatch でのメール送信イベントデータのフィルタ方法が決まるからです。

このセクションでは、ディメンションの選択に役立つ情報を示し、CloudWatch Event 送信先を設定セットに追加する方法を示します。

このセクションのトピック

- CloudWatch Event 送信先の追加
- CloudWatch ディメンションの選択

CloudWatch Event 送信先の追加

このセクションの手順では、作成した設定セットに CloudWatch Event 送信先の詳細を追加する方法を示します。ただし、この前に、「イベント送信先の作成」のステップ 1～6 を完了する必要があります。

また、Amazon SES API V2 で、UpdateConfigurationSetEventDestination オペレーションを使用して、イベントの送信先を作成および変更します。

コンソールを使用して CloudWatch Event 送信先の詳細を設定セットに追加するには

1. ステップ 7 でイベント送信先タイプとして CloudWatch を選択する際の詳細な手順は以下のとおりです。ここでは、「イベント送信先の作成」の前のステップがすべて完了していることを前提としています。CloudWatch の [送信先タイプ] を選択して、送信先の [名前] を入力し、[イベント発行] を有効にすると、[Amazon CloudWatch ディメンション] ペインが表示されます。このペインのフィールドは、以下の手順で指定します。(追加料金が適用されます。「CloudWatch のメトリクスあたりの料金」を参照してください。)
2. [値ソース] で、Amazon SES が CloudWatch に渡すデータを取得する方法を指定します。次の値ソースを使用できます。

   - メッセージタグ - Amazon SES は、`X-SES-MESSAGE-TAGS` ヘッダーまたは `EmailTags` API のパラメータを使用して、指定するタグからディメンション名と値を取得します。メッセージタグの使用方法の詳細については、「the section called “ステップ 3: 送信時に設定セットを指定する”」を参照してください。

     > **Note**
     >
     > メッセージタグには、0～9 の数字、A～Z の文字 (大文字と小文字)、ハイフン (-)、およびアンダースコア (_) を使用できます。

[メッセージタグ] 値ソースを使用し、Amazon SES 自動タグに基づいてディメンションを作成することもできます。自動タグを使用するには、自動タグの完全な名前を [Dimension Name] として入力します。例えば、設定セット自動タグに基づいてディメンションを作成するには、[ディメンション名] に `ses:configuration-set` を使用し、[デフォルト値] に設定セットの名前を使用します。自動タグの詳しい一覧については、「イベント発行が設定セットとメッセージタグと連携する方法」を参照してください。

- Email Header – Amazon SES が、E メール内のヘッダーからディメンション名と値を取得します。

> **Note**
>
> 以下の E メールヘッダーをディメンション名として使用することはできません: `Received`、`To`、`From`、`DKIM-Signature`、`CC`、`message-id`、または `Return-Path`。

- リンクタグ - Amazon SES は、リンクで指定したタグからディメンション名と値を取得します。リンクへのタグの追加の詳細については、「リンクに一意の識別子をタグ付けできますか?」を参照してください。

3. [ディメンション名] で、CloudWatch に渡すディメンションの名前を入力します。

> **Note**
>
> ディメンション名は、ASCII 文字 (a ~ z、A ~ Z)、数字 (0 ~ 9)、アンダースコア (_)、ダッシュ (-) のみを含みます。スペース、アクセント文字、非ラテン文字、およびその他の特殊文字は使用できません。

4. [Default Value] に、ディメンションの値を入力します。

> **Note**
>
> ディメンション値は、ASCII 文字 (a ~ z、A ~ Z)、数字 (0 ~ 9)、アンダースコア (_)、ダッシュ (-)、アットマーク (@)、ピリオド (.) のみを含みます。スペース、アクセント文字、非ラテン文字、およびその他の特殊文字は使用できません。

5. さらにディメンションを追加する場合は、[Add Dimension] を選択します。それ以外の場合は、次へ を選択します。

6. レビュー画面で、イベント目的地の定義に問題がなければ、送信先を追加するを選択します。

CloudWatch ディメンションの選択

CloudWatch ディメンションとして使用する名前と値を選択するときは、次の要素を考慮してください。

- メトリクスあたりの料金 – CloudWatch のベーシック Amazon SES メトリクスを無料で表示できます。ただし、イベント発行を使用してメトリクスを収集する場合は、CloudWatch 詳細モニタリング のコストが発生します。イベントタイプ、ディメンション名、ディメンション値それぞれの独自の組み合わせによって、CloudWatch でさまざまなカスタムメトリクスが作成されます。CloudWatch の詳細モニタリングを使用する場合、メトリクスごとに料金が発生します。このため、多くの異なる値を取得する可能性があるディメンションの選択を回避できます。例えば、「From」ドメインごとに E メール送信イベントを追跡することが特に必要な場合を除き、Amazon SES 自動タグ `ses:from-domain` のディメンションを定義する必要はありません。多くの異なる値が使用されるためです。詳細については、「CloudWatch 料金表」をご覧ください。
- メトリクスフィルタリング – メトリクスに複数のディメンションがある場合、CloudWatch では各ディメンションに基づいて別個にメトリクスにアクセスすることはできません。そのため、1 つの CloudWatch Event 送信先に複数のディメンションを追加する場合は、事前によく検討してください。たとえば、`campaign` あたりのメトリクスと、`campaign` と `genre` の組み合わせあたりのメトリクスが必要な場合、2 つのイベント送信先を追加する必要があります。ディメンションとして `campaign` のみ使用するイベント送信先と、`campaign` および `genre` の両方をディメンションとして使用するイベント送信先です。
- ディメンション値ソース – Amazon SES 固有のヘッダーまたは API へのパラメータを使用してディメンション値を指定する代わりに、Amazon SES が独自の MIME メッセージヘッダーからディメンション値を取得するように選択することもできます。このオプションは、すでにカスタムヘッダーを使用しており、ヘッダー値に基づいてメトリクスを収集するために、メールやメール送信 API 呼び出しを変更したくない場合に使用できます。Amazon SES イベント発行に独自の MIME メッセージヘッダーを使用する場合、Amazon SES イベント発行に使用するヘッダー名と値には、文字 A ～ Z、数字 0 ～ 9、アンダーバー (_)、アットマーク (@)、ハイフン (-)、およびピリオド (.) のみを含めることができます。その他の文字を含む名前または値を指定した場合、E メールの送信には成功しますが、イベントメトリクスは Amazon CloudWatch に出力されません。

CloudWatch コンセプトの詳細については、Amazon CloudWatch ユーザーガイドの「Amazon CloudWatch の概念」を参照してください。

Amazon SES イベント発行向けに Data Firehose イベント送信先を設定する

Amazon Data Firehose イベント送信先は、特定の Amazon SES E メール送信イベントを Firehose に発行するエンティティを指します。Firehose のイベント送信先を設定できるのは、設定セット内でのみであるため、まず設定セットを作成する必要があります。次に、設定セットにイベントの送信先を追加します。

このセクションの手順では、「イベント送信先の作成」のステップ 1～6 を既に実行したことを前提として、Firehose のイベント送信先の詳細を設定セットに追加する方法を説明しています。

また、Amazon SES API V2 で、UpdateConfigurationSetEventDestination オペレーションを使用して、イベントの送信先を作成および更新できます。

コンソールを使用して、Firehose イベント送信先を設定セットに追加するには

1. これらは、「ステップ 7」でイベントの送信先タイプとして Firehose を選択するための詳細な手順であり、既に「イベント送信先の作成」のそれ以前のステップをすべて完了していることを前提としています。Firehose の [送信先タイプ] を選択したら、送信先の [名前] を入力し、[イベント発行] を有効にすると、[Amazon Data Firehose 配信ストリーム] ペインが表示されます。このペインのフィールドは、以下の手順で指定します。
2. [配信ストリーム] では、既存の Firehose 配信ストリームを選択するか、[新しいストリームの作成] をクリックして、Firehose コンソールで新しいストリームを作成します。

   Firehose コンソールを使用したストリーム作成の詳細については、「Amazon Data Firehose デベロッパーガイド」の「Amazon Kinesis Firehose 配信ストリームの作成」を参照してください。
3. [Identity and Access Management (IAM) ロール] では、Amazon SES がお客様に代わって Firehose に発行するアクセス許可を付与されている IAM ロールを選択します。ロールは、既存のものを選択するか、Amazon SES で自動作成するか、独自に作成することができます。

   既存のロールを選択するか、独自のロールを作成する場合、ロールのポリシーを手動で変更し、Firehose 配信ストリームへのアクセス許可をロールに付与して、ロールを引き受けるアクセス許可を Amazon SES に付与する必要があります。エンドポイントポリシーの例については、「Amazon SES に対する Firehose 配信ストリームへの発行アクセス許可の付与」を参照してください。
4. [Next] を選択します。
5. レビュー画面で、イベント送信先の定義に問題がなければ、送信先を追加するを選択します。

UpdateConfigurationSetEventDestination API を使用して Firehose イベント送信先を追加する方法の詳細については、「Amazon Simple Email Service API リファレンス」を参照してください。

Amazon SES に対する Firehose 配信ストリームへの発行アクセス許可の付与

Amazon SES が Firehose 配信ストリームにレコードを発行できるようにするには、AWS Identity and Access Management (IAM) ロールを使用して、ロールのアクセス許可ポリシーと信頼ポリシーをアタッチするか、ポリシーを変更する必要があります。アクセス許可ポリシーのロールを使用すると、レコードを Firehose 配信ストリームに発行できます。信頼ポリシーは、Amazon SES がロールを引き受けることができるようにします。

このセクションでは、両方のポリシーの例を示します。IAM ロールへの添付ポリシーについての詳細は、IAM ユーザーガイドの「ロールの修正」を参照してください。

アクセス許可ポリシー

以下のアクセス許可ポリシーにより、ロールがデータレコードを Firehose 配信ストリームに発行できるようになります。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": [
        "firehose:PutRecordBatch"
      ],
      "Resource": [
        "arn:aws:firehose:delivery-region:111122223333:deliverystream/delivery-stream-name"
      ]
    }
  ]
}
```

上のポリシー例に、以下の変更を加えます。

- *delivery-region* は、Firehose 配信ストリームを作成した AWS リージョンに置き換えます。
- *111122223333* を自分の AWS アカウント ID に置き換えます。

- *delivery-stream-name* は、Firehose 配信ストリーム名に置き換えます。

信頼ポリシー

次の信頼ポリシーにより、Amazon SES はロールを引き受けることができます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ses.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "111122223333",
          "AWS:SourceArn": "arn:aws:ses:delivery-region:111122223333:configuration-
set/configuration-set-name"
        }
      }
    }
  ]
}
```

上のポリシー例に、以下の変更を加えます。

- *delivery-region* は、Firehose 配信ストリームを作成した AWS リージョンに置き換えます。
- *111122223333* を自分の AWS アカウント ID に置き換えます。
- *configuration-set-name* は、この Firehose 配信ストリームに関連付けられている設定セット名に置き換えます。

イベント発行のために Amazon EventBridge イベント送信先を設定する

Amazon EventBridge イベント送信先は、設定セットで指定したメール送信イベントについて通知します。SES は、イベント送信先の作成時に定義したイベントを生成して、EventBridge のデフォルトイベントバスに送信します。イベントバスとは、イベントを受信するルーターであり、イベントを複数の送信先に配信できます。E メール送信イベントと Amazon EventBridge の統合の詳細について

は、「EventBridge を使用したモニタリング」を参照してください。EventBridge イベント送信先をセットアップできるのは、設定セット内でのみであるため、設定セットにイベント送信先を追加する前に、設定セットを作成する必要があります。

このセクションの手順では、「イベント送信先の作成」のステップ 1～6 を既に実行したことを前提として、EventBridge のイベント送信先の詳細を設定セットに追加する方法を説明しています。

また、Amazon SES API V2 で、UpdateConfigurationSetEventDestination オペレーションを使用して、イベントの送信先を作成および変更します。

コンソールを使用して、EventBridge イベント送信先を設定セットに追加するには

1. これらは、「ステップ 7」でイベントの送信先タイプとして EventBridge を選択するための詳細な手順であり、既に「イベント送信先の作成」のそれ以前のステップをすべて完了していることを前提としています。Amazon EventBridge の [送信先タイプ] を選択して、送信先の [名前] を入力し、[イベント発行] を有効にすると、[Amazon EventBridge イベントバス] 情報ページが表示されます。
2. [Next] を選択します。
3. レビュー画面で、イベント送信先の定義に問題がなければ、送信先を追加するを選択します。これによりイベント送信先のサマリーページが開き、イベント送信先が正常に作成または変更されたかどうかを、成功バナーで確認できます。

イベント発行の Amazon Pinpoint イベント送信先のセットアップ

Amazon Pinpoint イベント送信先は、設定セットで指定したメール送信イベントについて通知します。Amazon Pinpoint イベント送信先をセットアップできるのは、設定セット内でのみであるため、設定セットにイベント送信先を追加する前に、設定セットを作成する必要があります。

このセクションの手順では、Amazon Pinpoint イベント送信先の詳細を設定セットに追加する方法を示します。ただし、その前に、「イベント送信先の作成」のステップ 1～6 を完了する必要があります。

また、Amazon SES API V2 で、UpdateConfigurationSetEventDestination オペレーションを使用して、イベントの送信先を作成および変更します。

Amazon Pinpoint プロジェクトで設定したチャネルのタイプには追加料金がかかります。詳細については、Amazon Pinpoint 料金を参照してください。

コンソールを使用して Amazon Pinpoint イベント送信先を設定セットに追加するには

1. ステップ 7でイベント送信先タイプとして Amazon Pinpoint を選択する際の詳細な手順は次のとおりです。「イベント送信先の作成」の前の手順がすべて完了していることを前提としています。

   > **Note**
   >
   > Amazon Pinpoint は、配送の遅延またはサブスクリプションのイベントタイプをサポートしません。

   Amazon Pinpoint の [送信先タイプ] を選択して、送信先の [名前] を入力し、[イベント発行] を有効にすると、[Amazon Pinpoint プロジェクトの詳細] ペインが表示されます。このペインのフィールドは、以下の手順で指定します。

2. プロジェクトについては、既存の Amazon Pinpoint プロジェクトを選択するか、Amazon Pinpoint で新しいプロジェクトを作成するか、新規作成します。

   プロジェクトの作成の詳細については、Amazon Pinpoint ユーザーガイドのプロジェクトの作成を参照してください。

3. 次へを選択します。

4. レビュー画面で、イベント送信先の定義に問題がなければ、送信先を追加するを選択します。これによりイベント送信先のサマリーページが開き、イベント送信先が正常に作成または変更されたかどうかを、成功バナーで確認できます。

イベント発行の Amazon SNS イベント送信先のセットアップ

Amazon SNS イベント送信先は、設定セットで指定したメール送信イベントについて通知します。Amazon SNS イベント送信先をセットアップできるのは、設定セット内でのみであるため、設定セットにイベント送信先を追加する前に、設定セットを作成する必要があります。

このセクションの手順では、Amazon SNS イベント送信先の詳細を設定セットに追加する方法を示します。ただし、その前に、「イベント送信先の作成」のステップ 1～6 を完了する必要があります。

また、Amazon SES API V2 で、UpdateConfigurationSetEventDestination オペレーションを使用して、イベントの送信先を作成および変更します。

> **Note**
>
> バウンス、苦情、配信に関するフィードバック通知は、Amazon SNS を通じて、検証済みのいずれかの送信 ID に対して設定することもできます。詳細については、「the section called "Amazon SNS通知の設定"」を参照してください。

Amazon SNS トピックにサブスクライブしているエンドポイントにメッセージを送信するには、追加料金が発生します。詳細については、「Amazon SNS の料金」を参照してください。

コンソールを使用してAmazon SNSイベント送信先を設定セットに追加するには

1. ステップ 7でイベント送信先タイプとして Amazon SNS を選択する際の詳細な手順は次のとおりです。「イベント送信先の作成」の前の手順がすべて完了していることを前提としています。Amazon SNS の [送信先タイプ] を選択したら、送信先の [名前] を入力し、[イベント発行] を有効にすると、[Simple Notification Service (SNS) トピック] ペインが表示されます。このペインのフィールドは、以下の手順で指定します。
2. [SNSトピック] で、既存の Amazon SNS トピックを選択するか、[SNSピックの作成] を選択して新しいトピックを作成します。

   トピックの作成の詳細については、Amazon Simple Notification Service デベロッパーガイドの「トピックの作成」を参照してください。

   > **Important**
   >
   > Amazon SNS を使用してトピックを作成する場合、タイプはスタンダードを選択します。(SES は FIFO タイプのトピックをサポートしていません。)

3. [Next] を選択します。
4. レビュー画面で、イベント送信先の定義に問題がなければ、送信先を追加するを選択します。これによりイベント送信先のサマリーページが開き、イベント送信先が正常に作成または変更されたかどうかを、成功バナーで確認できます。
5. 新しい SNS トピックを作成した場合も、既存のトピックを選択した場合も、トピックに通知を発行するために SES へのアクセスを許可する必要があります。前のステップで表示された、イベント送信先のサマリーページで、[Destination type] (送信先タイプ) カラムから Amazon SNS を選択します。これにより Amazon Simple Notification Service コンソールの [Topics] (トピック) リストが 表示されます。Amazon SNS コンソールで、次のステップを実行してください。

a. 前のステップで作成または変更した、SNS トピックの名前を選択します。

b. トピックの詳細画面で、[Edit] (編集) を選択します。

c. トピックに通知を発行するためのアクセス許可を SES に付与するには、SNS コンソールの [Edit topic] (トピックの編集) 画面で [Access policy] (アクセスポリシー) を展開し、[JSON editor] (JSON エディタ) に次の許可ポリシーを追加します。

```
{
  "Version": "2012-10-17",
  "Id": "notification-policy",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ses.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:topic_region:111122223333:topic_name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "111122223333",
          "AWS:SourceArn":
 "arn:aws:ses:topic_region:111122223333:configuration-set/configuration-set-
name"
        }
      }
    }
  ]
}
```

上のポリシー例に、以下の変更を加えます。

- *topic_region* を、SNS トピックを作成した AWS リージョンに置き換えます。
- *111122223333*を自分のAWSアカウント ID に置き換えます。
- *topic_name*は、SNS トピックの名前に置き換えます。
- *configuration-set-name* は、SNS イベントに関連付けられた設定セットの名前に置き換えます。

d. [Save changes] (変更の保存) をクリックします。

ステップ 3: E メールを送信するときに設定セットを指定する

設定セットを作成し、イベント発行先を追加したら、イベント発行の最後のステップとして E メールを送信します。

メールに関連付けられているイベントを発行するには、メールに関連付ける設定セットの名前を指定する必要があります。必要に応じて、メールを分類するためのメッセージタグを指定することもできます。

この情報を Amazon SES に提供する方法として、E メール送信 API へのパラメータ、Amazon SES 固有の E メールヘッダー、または MIME メッセージのカスタムヘッダーがあります。選択する方法は、次の表に示すように、使用するメール送信インターフェイスによって決まります。

メール送信インターフェイス	イベント発行方法
`SendEmail`	API パラメータ
`SendTemplatedEmail`	API パラメータ
`SendBulkTemplatedEmail`	API パラメータ
`SendCustomVerificationEmail`	API パラメータ
`SendRawEmail`	API パラメータ、Amazon SES 固有のメールヘッダー、またはカスタム MIME ヘッダー **Important** ヘッダーと API パラメータの両方を使用してメッセージタグを指定すると、Amazon SES では API パラメータで指定されたメッセージタグのみが使用されます。Amazon SES は、API パラメータとヘッダーで指定されたメッセージタグを結合しません。
SMTP インターフェイス	Amazon SES 固有の E メールヘッダー

以下のセクションでは、ヘッダーおよび API パラメータを使用して設定セットとメッセージタグを指定する方法について説明します。

- Amazon SES API パラメータの使用
- Amazon SES 固有の E メールヘッダーの使用
- カスタムメールヘッダーの使用

> **Note**
>
> 必要に応じて、E メールのヘッダーにメッセージタグを含めることができます。メッセージタグには、0〜9 の数字、A〜Z の文字 (大文字と小文字)、ハイフン (-)、およびアンダースコア (_) を使用できます。

Amazon SES API パラメータの使用

イベント発行で SendEmail、SendTemplatedEmail、SendBulkTemplatedEmail、SendCustomVerificationEmail、または SendRawEmail を使用するには、ConfigurationSet および MessageTag というデータ構造を API コールに渡すことで設定セットとメッセージタグを指定します。

Amazon SES API の使用については、Amazon Simple Email Service API リファレンス を参照してください。

Amazon SES 固有の E メールヘッダーの使用

SendRawEmail または SMTP インターフェイスを使用する場合、Amazon SES 固有のヘッダーを E メールに追加することで設定セットとメッセージタグを指定できます。Amazon SES は、E メールを送信する前にヘッダーを削除します。次の表に、使用するヘッダー名を示します。

イベント発行情報	ヘッダー
設定セット	`X-SES-CONFIGURATION-SET`
メッセージタグ	`X-SES-MESSAGE-TAGS`

次の例で、Amazon SES に送信した raw E メールにヘッダーがどのように表示されるかを示します。

```
X-SES-MESSAGE-TAGS: tagName1=tagValue1, tagName2=tagValue2
X-SES-CONFIGURATION-SET: myConfigurationSet
From: sender@example.com
To: recipient@example.com
Subject: Subject
Content-Type: multipart/alternative;
 boundary="----=_boundary"

------=_boundary
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 7bit

body
------=_boundary
Content-Type: text/html; charset=UTF-8
Content-Transfer-Encoding: 7bit

body
------=_boundary--
```

カスタムメールヘッダーの使用

設定セット名は Amazon SES 固有のヘッダー X-SES-CONFIGURATION-SET を使用して指定する必要がありますが、メッセージタグは独自の MIME ヘッダーを使用して指定できます。

> **Note**
>
> Amazon SES イベント発行に使用するヘッダー名と値は、ASCII にする必要があります。Amazon SES イベント発行のために ASCII 以外のヘッダー名または値を指定した場合でも、E メール送信コールに成功しますが、イベントメトリクスは Amazon CloudWatch に出力されません。

Amazon SES イベントデータの使用

イベント発行をセットアップし、メール送信の設定セットを指定すると、メールに関連付けられた設定セットのセットアップ時に指定したイベント送信先からメール送信イベントを取得できます。

このセクションでは、Amazon CloudWatch および Amazon Data Firehose から E メール送信イベントを取得する方法と、Amazon SNS が提供するイベントデータを解釈する方法について説明します。

- CloudWatch から Amazon SES イベントデータの取得
- Firehose からの Amazon SESイベントデータの取得
- Amazon SNS からの Amazon SES イベントデータの解釈

CloudWatch から Amazon SES イベントデータの取得

Amazon SES は、Amazon CloudWatch への E メール送信イベントに関するメトリクスを発行することができます。イベントデータを CloudWatch に発行する場合、これらのメトリクスは時系列データのセットとして提供されます。これらのメトリクスを使用して、メール送信のパフォーマンスをモニタリングできます。例えば、苦情メトリクスをモニタリングし、メトリクスが一定の値を超えたときに CloudWatch アラームがトリガーされるように設定できます。

Amazon SES がこれらのイベントを CloudWatch に発行する際の詳細度には、2 つのレベルがあります。

- アカウント全体AWS アカウント - これらの粗いメトリクス (Amazon SES コンソールおよびGetSendStatisticsAPI を使用してモニタリングするメトリクスに対応) は、AWS アカウント全体における合計です。Amazon SES は、これらのメトリクスを自動的に CloudWatch に発行します。
- 細かい – これらのメトリクスは、メッセージタグを使用して定義する E メール特性によって分類されます。これらのメトリクスを CloudWatch に公開するには、 CloudWatch Event 送信先を含めてイベント発行をセットアップし、E メールを送信時に設定セットを指定する必要があります。また、メッセージタグを指定したり、Amazon SES が自動的に提供する自動タグを使用できます。

このセクションでは、使用可能なメトリクスと、CloudWatch でメトリクスを表示する方法について説明します。

使用可能なメトリクス

以下の Amazon SES E メール送信メトリクスを CloudWatch に発行することができます。

- 送信数— 送信リクエストが成功すると、Amazon SES はそのメッセージを受信者のメールサーバーに配信しようと試行します。(アカウントレベルまたはグローバル抑制が使用されている場合でも、SES により送信済みとしてカウントされますが、配信は抑制されます)。
- レンダリング失敗 – テンプレートのレンダリング問題により、E メールが送信されませんでした。このイベントタイプは、テンプレートデータが見つからない場合や、テンプレートのパラメータとデータが一致しない場合に発生します。(このイベントタイプは、SendTemplatedEmail また

は SendBulkTemplatedEmail API オペレーションを使用して E メールを送信する場合にのみ発生します。)

- 拒否 - Amazon SESは E メールを受け取りましたが、この E メールにウイルスが含まれていると判断して拒否したため、受信者のメールサーバーに E メール配信を試みませんでした。
- 配信 - Amazon SES は、受取人のメールサーバーにメールを正常に配信しました。
- バウンス – ハードバウンスにより、受信者のメールサーバーが E メールを完全に拒否しました。(ソフトバウンスは、SES が E メールの配信を再試行しない場合にのみ含まれます。通常、これらのソフトバウンスは配信障害を示しますが、メールが受信者の受信トレイに正常に届いた場合でも、場合によってはソフトバウンスが返されることがあります。これは通常、受信者が不在の自動返信を送信した場合に発生します。ソフトバウンスの詳細については、この AWS re:Post 記事を参照してください)。
- 苦情— Eメールは受信者のメールサーバーに正常に配信されましたが、受信者はスパムとしてマークしました。
- 配信の遅延 – 一時的な問題が発生したため、メールを受信者のメールサーバーに配信できませんでした。配信の遅延は、受信者の受信トレイがいっぱいになった場合や、受信側の電子メールサーバーで一時的な問題が発生した場合などに発生します。
- サブスクリプション— メールは正常に配信されましたが、受信者がEメールヘッダーの List-Unsubscribe またはフッターの Unsubscribe リンクをクリックし、サブスクリプションの設定を更新しました。
- オープン – 受信者がメッセージを受け取り、E メールクライアントで開きました。
- クリック – 受信者はメール内の 1 つ以上のリンクをクリックしました。

使用できるディメンション

CloudWatch は、Amazon SES でユーザーが CloudWatch Event 送信先を設定セットに追加する際に指定したディメンション名を使用します。詳細については、「イベント発行の CloudWatch Event 送信先のセットアップ」を参照してください。

CloudWatch コンソールでの Amazon SES メトリクスの表示

次の手順では、CloudWatch コンソールを使用して Amazon SES イベント発行メトリクスを表示する方法について説明します。

CloudWatch コンソールを使用してメトリクスを表示するには

1. AWS Management Console にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。

2. 必要に応じてリージョンを変更します。ナビゲーションバーから、AWS リソースがあるリージョンを選択します。詳細については、「リージョンとエンドポイント」を参照してください。
3. ナビゲーションペインで [すべてのメトリクス] を選択します。
4. [メトリクス] ペインで [SES] を選択します。
5. 表示するメトリクスを選択します。細かいイベント発行メトリクスを表示するには、CloudWatch Event 発行のセットアップ時に指定したディメンションの組み合わせを選択します。CloudWatch メトリクスを使用したメトリクス表示の詳細については、「Amazon CloudWatch を使用する」を参照してください。

AWS CLI を使ってメトリクスを表示するには

- コマンドプロンプトで、次のコマンドを使用します。

```
aws cloudwatch list-metrics --namespace "AWS/SES"
```

Firehose からの Amazon SESイベントデータの取得

Amazon SES は、E メール送信イベントをJSONレコードとして Firehose に発行します。次に、Firehose は、Firehose で配信ストリームをセットアップしたときに選択した AWS サービス送信先にレコードを発行します。Firehose 配信ストリームの設定についての詳細は、「Amazon Data Firehose デベロッパーガイド」の「Firehose 配信ストリームの作成」を参照してください。

このセクションのトピック:

- Amazon が Firehose SES に発行するイベントデータの内容
- Amazon SES が Firehose に発行するイベントデータの例

Amazon が Firehose SES に発行するイベントデータの内容

Amazon SES は、E メール送信イベントレコードを JSON形式で Amazon Data Firehose に発行します。Firehose にイベントを発行する場合、Amazon は各JSONレコードに改行文字をSES付けます。

これらのすべての通知タイプのレコード例については、「Amazon SES が Firehose に発行するイベントデータの例」を参照してください。

このセクションのトピック

- 最上位JSONオブジェクト

- Mail オブジェクト
- Bounce オブジェクト
- 苦情のオブジェクト
- 配信オブジェクト
- Send オブジェクト
- Reject オブジェクト
- Open オブジェクト
- Click オブジェクト
- レンダリング失敗オブジェクト
- DeliveryDelay オブジェクト
- サブスクリプションオブジェクト

最上位JSONオブジェクト

E メール送信イベントレコードの最上位JSONオブジェクトには、次のフィールドが含まれます。

フィールド名	説明
eventType	イベントのタイプを説明する文字列。可能な値: Bounce、Complaint 、Delivery、Send、Reject、Open、Click、Rend Failure、DeliveryDelay 、または Subscription 。 もしイベント発行のセットアップをしなかった場合は、このフィールドは notificationType と名付けられます。
mail	イベントを生成した E メールに関する情報を含むJSONオブジェクト。
bounce	このフィールドは、eventType が Bounce の場合のみ表示されます。このフィールドには、バウンスに関する情報が含まれています。

フィールド名	説明
complaint	このフィールドは、eventType が Complaint の場合のみ表示されます。このフィールドには、苦情に関する情報が含まれています。
delivery	このフィールドは、eventType が Delivery の場合のみ表示されます。このフィールドには、配信に関する情報が含まれています。
send	このフィールドは、eventType が Send の場合のみ表示されます。
reject	このフィールドは、eventType が Reject の場合のみ表示されます。このフィールドには、拒否に関する情報が含まれています。
open	このフィールドは、eventType が Open の場合のみ表示されます。このフィールドには、オープンイベントに関する情報が含まれています。
click	このフィールドは、eventType が Click の場合のみ表示されます。このフィールドには、クリックイベントに関する情報が含まれています。
failure	このフィールドは、eventType が Rendering Failure の場合のみ表示されます。このフィールドには、レンダリング失敗イベントに関する情報が含まれています。
deliveryDelay	このフィールドは、eventType が DeliveryDelay の場合のみ表示されます。E メールの遅延配信に関する情報が含まれています。

フィールド名	説明
subscription	このフィールドは、eventType が Subscription の場合のみ表示されます。このフィールドには、サブスクリプション設定に関する情報が含まれています。

Mail オブジェクト

各メール送信イベントレコードには、mail オブジェクトの元のメールに関する情報が含まれています。JSON オブジェクトに関する情報を含む mail オブジェクトには、次のフィールドがあります。

フィールド名	説明
timestamp	メッセージが送信された ISO8601 形式 (YYYY-MM-DDThh:mm:ss.sZ) の日時。
messageId	Amazon がメッセージにSES割り当てた一意の ID。Amazon は、メッセージを送信したときにこの値をSES返しました。 **Note** このメッセージ ID は Amazon によって割り当てられましたSES。元の E メールのメッセージ ID は、mail オブジェクトの headers および commonHeaders フィールドにあります。
source	メッセージが送信された E メールアドレス (エンベロープMAILFROMアドレス) 。
sourceArn	E メールの送信に使用された ID の Amazon リソースネーム (ARN) 。送信承認の場合、sourceArn は、ID 所有者が代理送信者が E メールの送信に使用することを承認した ID

フィールド名	説明
	ARNの です。送信承認の詳細については、「Eメールの認証方法」を参照してください。
sendingAccountId	E メールの送信に使用された AWS アカウントのアカウント ID。送信承認の場合、sendingAccountId は代理送信者のアカウント ID です。
destination	元のメールの受取人の E メールアドレスのリスト。
headersTruncated	通知でヘッダーが切り捨てられたかどうかを示す文字列。切り捨ては、ヘッダーが 10 KB を超える場合に発生します。指定できる値は true および false です。
headers	E メールの元のヘッダーの一覧。リスト内の各ヘッダーには、name フィールドと value フィールドがあります。 **Note** headers フィールド内のメッセージ ID は、Amazon に渡した元のメッセージからのものですSES。Amazon が SESその後メッセージに割り当てたメッセージ ID は、 mail オブジェクトの messageId フィールドにあります。

フィールド名	説明
commonHeaders	E メールの元の一般的に使用されるヘッダーのマッピング。 **Note** commonHeaders フィールド内のメッセージ ID は、Amazon がSESその後mailオブジェクトの messageId フィールドのメッセージに割り当てたメッセージ ID です。
tags	E メールに関連付けられたタグのリストです。

Bounce オブジェクト

Bounce イベントに関する情報を含むJSONオブジェクトには、常に次のフィールドがあります。

フィールド名	説明
bounceType	Amazon によって決定されるバウンスのタイプSES。
bounceSubType	Amazon によって決定されるバウンスのサブタイプSES。
bouncedRecipients	バウンスとなった元のメールの受取人についての情報を含むリスト。
timestamp	がバウンス通知ISPを送信した日時ISO86（YYYY-MM-DDThh:mm:ss.sZ）。
feedbackId	バウンスの一意の ID。
reportingMTA	からの Reporting-MTA フィールドの値DSN。これは、 で説明されている配信、リレー、またはゲートウェイオペレーションを実行しよ

フィールド名	説明
	うとした Message Transfer Authority (MTA) の値ですDSN。 **Note** このフィールドは、配信ステータス通知 (DSN) がバウンスにアタッチされた場合にのみ表示されます。

バウンスとなった受取人

バウンスイベントには、1 人の受信者に関するものと複数の受信者に関するものがあります。bouncedRecipients フィールドにはオブジェクトのリストが含まれ (バウンスイベントが関係する受信者ごとに 1 つのオブジェクト)、常に以下のフィールドが含まれています。

フィールド名	説明
`emailAddress`	受取人の E メールアドレス。DSN が使用可能な場合、これは の `Final-Recipient` フィールドの値ですDSN。

オプションで、 DSNがバウンスにアタッチされている場合、次のフィールドが存在する場合もあります。

フィールド名	説明
`action`	からの `Action`フィールドの値DSN。これは、この受信者にメッセージを配信しようとMTAした結果、レポートによって実行されたアクションを示します。
`status`	からの `Status`フィールドの値DSN。これは、メッセージの配信状態を示す、受取人ごとに個

フィールド名	説明
	別の、トランスポート独立型ステータスコードです。
diagnosticCode	レポート によって発行されたステータスコードMTA。これは、 の Diagnostic-Codeフィールドの値ですDSN。このフィールドは、 には存在しない可能性があります DSN (したがって、 にも存在しない) JSON。

バウンスのタイプ

各バウンスイベントは、以下の表に示すいずれかのタイプになります。

イベント発行システムは、Amazon によって再試行されなくなるハードバウンスとソフトバウンスのみを公開しますSES。とマークされたバウンスを受け取ったらPermanent、対応する E メールアドレスをメーリングリストから削除する必要があります。今後送信することはできません。Transient バウンスは、メッセージのソフトバウンスが複数回発生し、Amazon SES が再配信を停止したときに送信されます。最初は Transient バウンスとして返されても、将来的には同じアドレスに正常に再送できる場合があります。

bounceType	bounceSubType	説明
Undetermined	Undetermined	Amazon SES は特定のバウンス理由を特定できませんでした。
Permanent	General	Amazon は一般的なハードバウンスSESを受け取りました。このタイプのバウンスを受信した場合は、メーリングリストからこの受信者のメールアドレスを削除する必要があります。
Permanent	NoEmail	ターゲット E メールアドレスが存在しないため、Amazon は永続的なハードバウンスSESを受け取りました。このタイプのバウンスを受信した場合は、メーリングリストからこの受信者のメールアドレスを削除する必要があります。

bounceType	bounceSubType	説明
Permanent	Suppressed	Amazon SESは、バウンスした最近の履歴が無効なアドレスであるため、このアドレスへの送信を抑制しました。グローバルサプレッションリストを上書きするには、「Amazon SESアカウントレベルのサプレッションリストの使用」を参照してください。
Permanent	OnAccountSuppressionList	Amazon SESは、アカウントレベルのサプレッションリストに含まれているため、このアドレスへの送信を抑制しました。これは、バウンス率のメトリクスに対してはカウントされません。
Transient	General	Amazon は一般的なバウンスSESを受け取りました。今後、この受信者に正常に再送できる可能性があります。
Transient	MailboxFull	Amazon はメールボックスのフルバウンスSESを受け取りました。今後、この受信者に正常に再送できる可能性があります。
Transient	MessageTooLarge	Amazon は、バウンスが大きすぎるというメッセージSESを受信しました。メッセージサイズを小さくすることで、この受信者に正常に再送できる可能性があります。
Transient	CustomTimeoutExceeded	Amazon SESは、E メール送信者が指定した時間内に E メールを正常に配信できませんでした。(バウンスメッセージは、定義された 内で配信試行が失敗する可能性がある理由を指定します) TTL。
Transient	ContentRejected	Amazon はコンテンツ拒否バウンスSESを受け取りました。メッセージのコンテンツを変更することで、この受信者に正常に再送できる可能性があります。

bounceType	bounceSubType	説明
Transient	AttachmentRejected	Amazon が添付ファイルの拒否バウンスSESを受け取りました。添付ファイルを削除または変更することで、この受信者に正常に再送できる可能性があります。

苦情のオブジェクト

Complaint イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
complainedRecipients	苦情を送信した可能性がある受取人に関する情報を含むリスト。
timestamp	が苦情通知ISPを送信した日時 (ISO86-YYYYMM-DDThh:mm:ss.sZ) 。
feedbackId	苦情の一意の ID。
complaintSubType	Amazon によって決定される苦情のサブタイプ SES。

また、フィードバックレポートが苦情に添付されている場合、以下のフィールドが示される場合があります。

フィールド名	説明
userAgent	フィードバックレポートの User-Agent フィールドの値。これは、レポートを生成したシステムの名前とバージョンを示します。
complaintFeedbackType	から受け取ったフィードバックレポートの Feedback-Type フィールドの値ISP。これには、フィードバックのタイプが含まれます。

フィールド名	説明
arrivalDate	フィードバックレポートの Arrival-Date または Received-Date フィールドの値は ISO8601 形式 (YYYY-MM-DDThh:mm:ss.sZ) です。このフィールドはレポートにない場合があります (したがって、 にもないJSON)。

苦情を申告した受取人

complainedRecipients フィールドには、苦情の送信元と思われる受信者のリストが含まれます。

> **Important**
>
> ほとんどの は苦情通知から苦情を送信した受信者の E メールアドレスISPsを編集するため、このリストには、元のメッセージの受信者と苦情を受信した ISP に基づいて、苦情を送信した可能性のある受信者に関する情報が含まれています。Amazon は元のメッセージに対して検索SESを実行して、この受信者リストを決定します。

JSON このリストの オブジェクトには、次のフィールドが含まれます。

フィールド名	説明
emailAddress	受取人の E メールアドレス。

苦情のタイプ

Internet Assigned Numbers Authority のウェブサイトによるとISP、レポート によって割り当てられた complaintFeedbackTypeフィールドに次の苦情タイプが表示される場合があります。

フィールド名	説明
abuse	迷惑メールまたはその他のメール不正使用。
auth-failure	E メールの認証障害レポート

フィールド名	説明
fraud	なんらかの詐欺またはフィッシング行為を示します。
not-spam	レポートの提供者がこのメッセージをスパムではないと見なしていることを示します。このタイプは、誤ってスパムとしてタグ付けまたは分類されたメッセージを修正するために使用される場合があります。
other	その他の登録されたタイプに該当しないフィードバックを示します。
virus	元のメッセージでウイルスが見つかったことを示します。

配信オブジェクト

Delivery イベントに関する情報を含むJSONオブジェクトには、常に次のフィールドがあります。

フィールド名	説明
timestamp	Amazon が受信者のメールサーバーに E メールをSES配信した日時を ISO8601 形式 (YYYY-MM-DDThh:mm:ss.sZ) で表します。
processingTimeMillis	Amazon が送信者からのリクエストSESを受け入れてから、Amazon が受信者のメールサーバーにメッセージをSES渡したまでの時間をミリ秒単位で表します。
recipients	配信イベントの適用対象となる受信者のリスト。
smtpResponse	Amazon からの E メールを受け入れISPたリモート のSMTPレスポンスメッセージSES。こ

フィールド名	説明
	のメッセージは、E メール、メールサーバー、および の受信によって異なりますISP。
reportingMTA	SES メールを送信した Amazon メールサーバーのホスト名。

Send オブジェクト

send イベントに関する情報を含むJSONオブジェクトは常に空です。

Reject オブジェクト

Reject イベントに関する情報を含むJSONオブジェクトには、常に次のフィールドがあります。

フィールド名	説明
reason	メールが拒否された理由。唯一の可能な値はです。これはBad content、Amazon が E メールにウイルスが含まれていることをSES検出したことを意味します。メッセージが拒否されると、Amazon はメッセージの処理SESを停止し、受信者のメールサーバーへの配信を試みません。

Open オブジェクト

Open イベントに関する情報を含むJSONオブジェクトには、常に次のフィールドが含まれます。

フィールド名	説明
ipAddress	受信側の IP アドレス
timestamp	オープンイベントが ISO8601 形式で発生した日時 (YYYY-MM-DDThh:mm:ss.sZ) 。

フィールド名	説明
userAgent	受取人がメールを開くために使用したしたデバイスまたは E メールクライアントのユーザーエージェント。

Click オブジェクト

Click イベントに関する情報を含むJSONオブジェクトには、常に次のフィールドが含まれます。

フィールド名	説明
ipAddress	受信側の IP アドレス
timestamp	クリックイベントが発生した日時は ISO8601 形式 (YYYY-MM-DDThh:mm:ss.sZ) です。
userAgent	受信者が E メールのリンクをクリックするために使用したクライアントのユーザーエージェント。
link	受信者がクリックしたリンクURLの 。
linkTags	ses:tags 属性を使用してリンクに追加されたタグのリスト。メール内のリンクにタグを追加する方法については、「Amazon SES E メール送信メトリクスに関するよくある質問」の「Q5. リンクに一意の識別子をタグ付けできますか?」を参照してください。

レンダリング失敗オブジェクト

Rendering Failure イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
templateName	E メールの送信に使用されるテンプレートの名前。
errorMessage	レンダリング失敗に関する詳細情報を提供するメッセージ。

DeliveryDelay オブジェクト

DeliveryDelay イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
delayType	遅延のタイプ。可能な値は以下のとおりです。 • InternalFailure – Amazon の内部SESの問題により、メッセージが遅延しました。 • 全般 — SMTP会話中に一般的な障害が発生しました。 • MailboxFull – 受信者のメールボックスがいっぱいで、追加のメッセージを受信できません。 • SpamDetected – 受信者のメールサーバーが、アカウントから大量の未承諾 E メールを検出しました。 • RecipientServerError – 受信者の E メールサーバーに一時的な問題が発生し、メッセージの配信が妨げられています。 • IPFailure – メッセージを送信している IP アドレスが、受信者の E メールプロバイダーによってブロックまたはスロットリングされています。

フィールド名	説明
	• TransientCommunicationFailure – 受信者の E メールプロバイダーとのSMTP会話中に一時的な通信障害が発生しました。 • BYOIPHostNameLookupUnavailable – Amazon SES は IP アドレスのDNSホスト名を検索できませんでした。このタイプの遅延は、Bring Your Own IP を使用する場合にのみ発生します。 • Undetermined – Amazon SESは配信遅延の理由を特定できませんでした。 • SendingDeferral – Amazon SESは、メッセージを内部的に延期することが適切であると判断しました。
`delayedRecipients`	E メールの受取人に関する情報を含むオブジェクト。
`expirationTime`	Amazon がメッセージの配信SESを停止する日時。この値は 8601 ISO 形式で表示されます。
`reportingMTA`	遅延を報告した Message Transfer Agent (MTA) の IP アドレス。
`timestamp`	遅延が発生した日時。8601 ISO 形式で表示されます。

遅延受取人

`delayedRecipients` オブジェクトには、以下の値が含まれています。

フィールド名	説明
`emailAddress`	メッセージの配信が遅れる原因となった E メールアドレス。

フィールド名	説明
status	配信遅延に関連付けられたSMTPステータスコード。
diagnosticCode	受信メッセージ転送エージェントによって提供される診断コード (MTA)。

サブスクリプションオブジェクト

Subscription イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
contactList	連絡先が含まれているリストの名前。
timestamp	がサブスクリプション通知ISPを送信したときの ISO8601 形式 (YYYY-MM-DDThh:mm:ss.sZ) の日時。
source	メッセージが送信された E メールアドレス (エンベロープMAILFROMアドレス)。
newTopicPreferences	データ構造 (マップ) JSON。連絡先リスト内のすべてのトピックのサブスクリプションステータスを指定し、変更後のステータスを示します (連絡先がサブスクライブまたはサブスクライブ解除)。
oldTopicPreferences	データ構造 (マップ) JSON。連絡先リスト内のすべてのトピックのサブスクリプションステータスを指定し、変更前のステータス (連絡先がサブスクライブまたはサブスクライブ解除) を示します。

新旧トピックの設定

newTopicPreferences および oldTopicPreferences オブジェクトには、以下の値が含まれています。

フィールド名	説明
unsubscribeAll	連絡先リストのすべてのトピックから、連絡先がサブスクライブ解除されているかどうかを指定します。
topicSubscriptionStatus	指定されたイベントタイプの から通知を受信するように現在サブスクライブされているかどうかを示す topicName フィールドで、トピックSESのサブスクリプションステータスを指定します。指定できる値は、 subscriptionStatus フィールドの OptIn (サブスクライブ) または OptOut (サブスクライブ解除) です。
topicDefaultSubscriptionStatus	イベント送信先に追加された新しいトピックをデフォルトでサブスクライブするか、サブスクライブ解除するかを決定する topicName フィールドで、トピックのデフォルトのサブスクリプションステータスを指定します。指定できる値は、 subscriptionStatus フィールドで OptIn (デフォルトでサブスクライブ) または OptOut (デフォルトでサブスクライブ解除) です。

Amazon SES が Firehose に発行するイベントデータの例

このセクションでは、Amazon が Firehose に発行する E SES メール送信イベントレコードのタイプの例を示します。

このセクションのトピック:

- バウンスレコード

- 苦情レコード
- 配信レコード
- Send レコード
- Reject レコード
- Open レコード
- Click レコード
- レンダリング失敗レコード
- DeliveryDelay レコード
- サブスクリプションレコード

> **Note**
>
> 次の例では、 tagフィールドが使用されていますが、すべてのイベントタイプのタグの発行SESをサポートする設定セットによるイベント発行を使用しています。アイデンティティでフィードバック通知を直接使用する場合、 SESはタグを発行しません。設定セットの作成または設定セットの変更をする場合、タグの追加について参照してください。

バウンスレコード

Amazon が Firehose SES に発行するBounceイベントレコードの例を次に示します。

```
{
  "eventType":"Bounce",
  "bounce":{
    "bounceType":"Permanent",
    "bounceSubType":"General",
    "bouncedRecipients":[
      {
        "emailAddress":"recipient@example.com",
        "action":"failed",
        "status":"5.1.1",
        "diagnosticCode":"smtp; 550 5.1.1 user unknown"
      }
    ],
    "timestamp":"2017-08-05T00:41:02.669Z",
    "feedbackId":"01000157c44f053b-61b59c11-9236-11e6-8f96-7be8aexample-000000",
    "reportingMTA":"dsn; mta.example.com"
```

```
  },
  "mail":{
    "timestamp":"2017-08-05T00:40:02.012Z",
    "source":"Sender Name <sender@example.com>",
    "sourceArn":"arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId":"123456789012",
    "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination":[
      "recipient@example.com"
    ],
    "headersTruncated":false,
    "headers":[
      {
        "name":"From",
        "value":"Sender Name <sender@example.com>"
      },
      {
        "name":"To",
        "value":"recipient@example.com"
      },
      {
        "name":"Subject",
        "value":"Message sent from Amazon SES"
      },
      {
        "name":"MIME-Version",
        "value":"1.0"
      },
      {
        "name":"Content-Type",
        "value":"multipart/alternative; boundary=\"----
=_Part_7307378_1629847660.1516840721503\""
      }
    ],
    "commonHeaders":{
      "from":[
        "Sender Name <sender@example.com>"
      ],
      "to":[
        "recipient@example.com"
      ],
      "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject":"Message sent from Amazon SES"
    },
```

```
    "tags":{
      "ses:configuration-set":[
        "ConfigSet"
      ],
      "ses:source-ip":[
        "192.0.2.0"
      ],
      "ses:from-domain":[
        "example.com"
      ],
      "ses:caller-identity":[
        "ses_user"
      ]
    }
  }
}
```

苦情レコード

Amazon が Firehose SES に発行するComplaintイベントレコードの例を次に示します。

```
{
  "eventType":"Complaint",
  "complaint": {
    "complainedRecipients":[
      {
        "emailAddress":"recipient@example.com"
      }
    ],
    "timestamp":"2017-08-05T00:41:02.669Z",
    "feedbackId":"01000157c44f053b-61b59c11-9236-11e6-8f96-7be8aexample-000000",
    "userAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/60.0.3112.90 Safari/537.36",
    "complaintFeedbackType":"abuse",
    "arrivalDate":"2017-08-05T00:41:02.669Z"
  },
  "mail":{
    "timestamp":"2017-08-05T00:40:01.123Z",
    "source":"Sender Name <sender@example.com>",
    "sourceArn":"arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId":"123456789012",
    "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination":[
      "recipient@example.com"
```

```
    ],
    "headersTruncated":false,
    "headers":[
      {
        "name":"From",
        "value":"Sender Name <sender@example.com>"
      },
      {
        "name":"To",
        "value":"recipient@example.com"
      },
      {
        "name":"Subject",
        "value":"Message sent from Amazon SES"
      },
      {
        "name":"MIME-Version","value":"1.0"
      },
      {
        "name":"Content-Type",
        "value":"multipart/alternative; boundary=\"----
=_Part_7298998_679725522.1516840859643\""
      }
    ],
    "commonHeaders":{
      "from":[
        "Sender Name <sender@example.com>"
      ],
      "to":[
        "recipient@example.com"
      ],
      "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject":"Message sent from Amazon SES"
    },
    "tags":{
      "ses:configuration-set":[
        "ConfigSet"
      ],
      "ses:source-ip":[
        "192.0.2.0"
      ],
      "ses:from-domain":[
        "example.com"
      ],
```

```
      "ses:caller-identity":[
        "ses_user"
      ]
    }
  }
}
```

配信レコード

Amazon が Firehose SES に発行するDeliveryイベントレコードの例を次に示します。

```
{
  "eventType": "Delivery",
  "mail": {
    "timestamp": "2016-10-19T23:20:52.240Z",
    "source": "sender@example.com",
    "sourceArn": "arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId": "123456789012",
    "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination": [
      "recipient@example.com"
    ],
    "headersTruncated": false,
    "headers": [
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
        "value": "text/html; charset=UTF-8"
      },
```

```
      {
        "name": "Content-Transfer-Encoding",
        "value": "7bit"
      }
    ],
    "commonHeaders": {
      "from": [
        "sender@example.com"
      ],
      "to": [
        "recipient@example.com"
      ],
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject": "Message sent from Amazon SES"
    },
    "tags": {
      "ses:configuration-set": [
        "ConfigSet"
      ],
      "ses:source-ip": [
        "192.0.2.0"
      ],
      "ses:from-domain": [
        "example.com"
      ],
      "ses:caller-identity": [
        "ses_user"
      ],
      "ses:outgoing-ip": [
        "192.0.2.0"
      ],
      "myCustomTag1": [
        "myCustomTagValue1"
      ],
      "myCustomTag2": [
        "myCustomTagValue2"
      ]
    }
  },
  "delivery": {
    "timestamp": "2016-10-19T23:21:04.133Z",
    "processingTimeMillis": 11893,
    "recipients": [
      "recipient@example.com"
```

```
    ],
    "smtpResponse": "250 2.6.0 Message received",
    "reportingMTA": "mta.example.com"
  }
}
```

Send レコード

Amazon が Firehose SES に発行するSendイベントレコードの例を次に示します。

```
{
  "eventType": "Send",
  "mail": {
    "timestamp": "2016-10-14T05:02:16.645Z",
    "source": "sender@example.com",
    "sourceArn": "arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId": "123456789012",
    "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination": [
      "recipient@example.com"
    ],
    "headersTruncated": false,
    "headers": [
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
        "value": "multipart/mixed;  boundary=\"----=_Part_0_716996660.1476421336341\""
      },
      {
```

```
        "name": "X-SES-MESSAGE-TAGS",
        "value": "myCustomTag1=myCustomTagValue1, myCustomTag2=myCustomTagValue2"
      }
    ],
    "commonHeaders": {
      "from": [
        "sender@example.com"
      ],
      "to": [
        "recipient@example.com"
      ],
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject": "Message sent from Amazon SES"
    },
    "tags": {
      "ses:configuration-set": [
        "ConfigSet"
      ],
      "ses:source-ip": [
        "192.0.2.0"
      ],
      "ses:from-domain": [
        "example.com"
      ],
      "ses:caller-identity": [
        "ses_user"
      ],
      "myCustomTag1": [
        "myCustomTagValue1"
      ],
      "myCustomTag2": [
        "myCustomTagValue2"
      ]
    }
  },
  "send": {}
}
```

Reject レコード

Amazon が Firehose SES に発行するRejectイベントレコードの例を次に示します。

```
{
  "eventType": "Reject",
```

```
  "mail": {
    "timestamp": "2016-10-14T17:38:15.211Z",
    "source": "sender@example.com",
    "sourceArn": "arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId": "123456789012",
    "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination": [
      "sender@example.com"
    ],
    "headersTruncated": false,
    "headers": [
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
        "value": "multipart/mixed; boundary=\"qMm9M+Fa2AknHoGS\""
      },
      {
        "name": "X-SES-MESSAGE-TAGS",
        "value": "myCustomTag1=myCustomTagValue1, myCustomTag2=myCustomTagValue2"
      }
    ],
    "commonHeaders": {
      "from": [
        "sender@example.com"
      ],
      "to": [
        "recipient@example.com"
      ],
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
```

```
      "subject": "Message sent from Amazon SES"
    },
    "tags": {
      "ses:configuration-set": [
        "ConfigSet"
      ],
      "ses:source-ip": [
        "192.0.2.0"
      ],
      "ses:from-domain": [
        "example.com"
      ],
      "ses:caller-identity": [
        "ses_user"
      ],
      "myCustomTag1": [
        "myCustomTagValue1"
      ],
      "myCustomTag2": [
        "myCustomTagValue2"
      ]
    }
  },
  "reject": {
    "reason": "Bad content"
  }
}
```

Open レコード

以下は、Amazon が Firehose SES に発行するOpenイベントレコードの例です。

```
{
  "eventType": "Open",
  "mail": {
    "commonHeaders": {
      "from": [
        "sender@example.com"
      ],
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject": "Message sent from Amazon SES",
      "to": [
        "recipient@example.com"
      ]
```

```
    },
    "destination": [
      "recipient@example.com"
    ],
    "headers": [
      {
        "name": "X-SES-CONFIGURATION-SET",
        "value": "ConfigSet"
      },
      {
        "name":"X-SES-MESSAGE-TAGS",
        "value":"myCustomTag1=myCustomValue1, myCustomTag2=myCustomValue2"
      },
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
        "value": "multipart/alternative; boundary=\"XBoundary\""
      }
    ],
    "headersTruncated": false,
    "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "sendingAccountId": "123456789012",
    "source": "sender@example.com",
    "tags": {
      "myCustomTag1":[
        "myCustomValue1"
      ],
      "myCustomTag2":[
        "myCustomValue2"
```

```
      ],
      "ses:caller-identity": [
        "IAM_user_or_role_name"
      ],
      "ses:configuration-set": [
        "ConfigSet"
      ],
      "ses:from-domain": [
        "example.com"
      ],
      "ses:source-ip": [
        "192.0.2.0"
      ]
    },
    "timestamp": "2017-08-09T21:59:49.927Z"
  },
  "open": {
    "ipAddress": "192.0.2.1",
    "timestamp": "2017-08-09T22:00:19.652Z",
    "userAgent": "Mozilla/5.0 (iPhone; CPU iPhone OS 10_3_3 like Mac OS X)
 AppleWebKit/603.3.8 (KHTML, like Gecko) Mobile/14G60"
  }
}
```

Click レコード

Amazon が Firehose SES に発行するClickイベントレコードの例を次に示します。

```
{
  "eventType": "Click",
  "click": {
    "ipAddress": "192.0.2.1",
    "link": "http://docs.aws.amazon.com/ses/latest/DeveloperGuide/send-email-
smtp.html",
    "linkTags": {
      "samplekey0": [
        "samplevalue0"
      ],
      "samplekey1": [
        "samplevalue1"
      ]
    },
    "timestamp": "2017-08-09T23:51:25.570Z",
```

```
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/60.0.3112.90 Safari/537.36"
  },
  "mail": {
    "commonHeaders": {
      "from": [
        "sender@example.com"
      ],
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject": "Message sent from Amazon SES",
      "to": [
        "recipient@example.com"
      ]
    },
    "destination": [
      "recipient@example.com"
    ],
    "headers": [
      {
        "name": "X-SES-CONFIGURATION-SET",
        "value": "ConfigSet"
      },
      {
        "name":"X-SES-MESSAGE-TAGS",
        "value":"myCustomTag1=myCustomValue1, myCustomTag2=myCustomValue2"
      },
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
```

```
        "value": "multipart/alternative; boundary=\"XBoundary\""
      },
      {
        "name": "Message-ID",
        "value": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000"
      }
    ],
    "headersTruncated": false,
    "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "sendingAccountId": "123456789012",
    "source": "sender@example.com",
    "tags": {
      "myCustomTag1":[
        "myCustomValue1"
      ],
      "myCustomTag2":[
        "myCustomValue2"
      ],
      "ses:caller-identity": [
        "ses_user"
      ],
      "ses:configuration-set": [
        "ConfigSet"
      ],
      "ses:from-domain": [
        "example.com"
      ],
      "ses:source-ip": [
        "192.0.2.0"
      ]
    },
    "timestamp": "2017-08-09T23:50:05.795Z"
  }
}
```

レンダリング失敗レコード

Amazon が Firehose SES に発行するRendering Failureイベントレコードの例を次に示します。

```
{
  "eventType":"Rendering Failure",
  "mail":{
    "timestamp":"2018-01-22T18:43:06.197Z",
    "source":"sender@example.com",
```

```
    "sourceArn":"arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId":"123456789012",
    "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination":[
      "recipient@example.com"
    ],
    "headersTruncated":false,
    "tags":{
      "ses:configuration-set":[
        "ConfigSet"
      ]
    }
  },
  "failure":{
    "errorMessage":"Attribute 'attributeName' is not present in the rendering data.",
    "templateName":"MyTemplate"
  }
}
```

DeliveryDelay レコード

Amazon が Firehose SES に発行するDeliveryDelayイベントレコードの例を次に示します。

```
{
  "eventType": "DeliveryDelay",
  "mail":{
    "timestamp":"2020-06-16T00:15:40.641Z",
    "source":"sender@example.com",
    "sourceArn":"arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId":"123456789012",
    "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination":[
      "recipient@example.com"
    ],
    "headersTruncated":false,
    "tags":{
      "ses:configuration-set":[
        "ConfigSet"
      ]
    }
  },
  "deliveryDelay": {
    "timestamp": "2020-06-16T00:25:40.095Z",
    "delayType": "TransientCommunicationFailure",
```

```
    "expirationTime": "2020-06-16T00:25:40.914Z",
    "delayedRecipients": [{
      "emailAddress": "recipient@example.com",
      "status": "4.4.1",
      "diagnosticCode": "smtp; 421 4.4.1 Unable to connect to remote host"
    }]
  }
}
```

サブスクリプションレコード

Amazon が Firehose SES に発行するSubscriptionイベントレコードの例を次に示します。

```
{
  "eventType": "Subscription",
  "mail": {
    "timestamp": "2022-01-12T01:00:14.340Z",
    "source": "sender@example.com",
    "sourceArn": "arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId": "123456789012",
    "messageId": "EXAMPLEe4bccb684-777bc8de-afa7-4970-92b0-f515137b1497-000000",
    "destination": ["recipient@example.com"],
    "headersTruncated": false,
    "headers": [
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
        "value": "text/html; charset=UTF-8"
      },
```

```
      {
        "name": "Content-Transfer-Encoding",
        "value": "7bit"
      }
    ],
    "commonHeaders": {
      "from": ["sender@example.com"],
      "to": ["recipient@example.com"],
      "messageId": "EXAMPLEe4bccb684-777bc8de-afa7-4970-92b0-f515137b1497-000000",
      "subject": "Message sent from Amazon SES"
    },
    "tags": {
      "ses:operation": ["SendEmail"],
      "ses:configuration-set": ["ConfigSet"],
      "ses:source-ip": ["192.0.2.0"],
      "ses:from-domain": ["example.com"],
      "ses:caller-identity": ["ses_user"],
      "myCustomTag1": ["myCustomValue1"],
      "myCustomTag2": ["myCustomValue2"]
    }
  },
  "subscription": {
    "contactList": "ContactListName",
    "timestamp": "2022-01-12T01:00:17.910Z",
    "source": "UnsubscribeHeader",
    "newTopicPreferences": {
      "unsubscribeAll": true,
      "topicSubscriptionStatus": [
        {
          "topicName": "ExampleTopicName",
          "subscriptionStatus": "OptOut"
        }
      ]
    },
    "oldTopicPreferences": {
      "unsubscribeAll": false,
      "topicSubscriptionStatus": [
        {
          "topicName": "ExampleTopicName",
          "subscriptionStatus": "OptOut"
        }
      ]
    }
  }
```

```
}
```

Amazon SNS からの Amazon SES イベントデータの解釈

Amazon SES は、E メール送信イベントをJSON レコードとして Amazon Simple Notification Service (Amazon SNS) に発行します。次に、Amazon SNS は、イベント送信先に関連付けられている Amazon SNS トピックをサブスクライブしているエンドポイントに、通知を送信します。Amazon SNS でのトピックおよびサブスクリプションの設定に関する詳細は、Amazon Simple Notification Service デベロッパーガイドの「スタート方法」を参照してください。

レコードコンテンツの説明とレコードの例については、以下のセクションを参照してください。

- イベントレコードのコンテンツ
- イベントレコードの例

Amazon が Amazon SES に発行するイベントデータの内容 SNS

Amazon SES は、E メール送信イベントレコードを JSON形式で Amazon Simple Notification Service に発行します。

これらのすべての通知タイプのレコード例については、「Amazon SES が Amazon SNS に発行するイベントデータの例」を参照してください。

このセクションのトピック:

- 最上位JSONオブジェクト
- Mail オブジェクト
- Bounce オブジェクト
- 苦情のオブジェクト
- 配信オブジェクト
- Send オブジェクト
- Reject オブジェクト
- Open オブジェクト
- Click オブジェクト
- レンダリング失敗オブジェクト
- DeliveryDelay オブジェクト

- サブスクリプションオブジェクト

最上位JSONオブジェクト

E メール送信イベントレコードの最上位JSONオブジェクトには、次のフィールドが含まれます。他にどのオブジェクトが存在するのかは、イベントタイプによって決まります。

フィールド名	説明
eventType	イベントのタイプを説明する文字列。可能な値: Bounce、Complaint 、Delivery、Send、Reject、Open、Click、Rend Failure、DeliveryDelay 、または Subscription 。 もしイベント発行のセットアップをしなかった場合は、このフィールドは notificationType と名付けられます。
mail	イベントを生成した E メールに関する情報を含むJSONオブジェクト。
bounce	このフィールドは、eventType が Bounce の場合のみ表示されます。このフィールドには、バウンスに関する情報が含まれています。
complaint	このフィールドは、eventType が Complaint の場合のみ表示されます。このフィールドには、苦情に関する情報が含まれています。
delivery	このフィールドは、eventType が Delivery の場合のみ表示されます。このフィールドには、配信に関する情報が含まれています。
send	このフィールドは、eventType が Send の場合のみ表示されます。

フィールド名	説明
reject	このフィールドは、eventType が Reject の場合のみ表示されます。このフィールドには、拒否に関する情報が含まれています。
open	このフィールドは、eventType が Open の場合のみ表示されます。このフィールドには、オープンイベントに関する情報が含まれています。
click	このフィールドは、eventType が Click の場合のみ表示されます。このフィールドには、クリックイベントに関する情報が含まれています。
failure	このフィールドは、eventType が Rendering Failure の場合のみ表示されます。このフィールドには、レンダリング失敗イベントに関する情報が含まれています。
deliveryDelay	このフィールドは、eventType が DeliveryDelay の場合のみ表示されます。E メールの遅延配信に関する情報が含まれています。
subscription	このフィールドは、eventType が Subscription の場合のみ表示されます。このフィールドには、サブスクリプション設定に関する情報が含まれています。

Mail オブジェクト

各メール送信イベントレコードには、mail オブジェクトの元のメールに関する情報が含まれています。JSON オブジェクトに関する情報を含む mail オブジェクトには、次のフィールドがあります。

フィールド名	説明
timestamp	メッセージが送信された日時を ISO8601 形式 (YYYY-MM-DDThh:mm:ss.sZ) で表します。
messageId	Amazon がメッセージにSES割り当てた一意の ID。Amazon は、メッセージを送信したときにこの値をSES返しました。 **Note** このメッセージ ID は Amazon によって割り当てられましたSES。元の E メールのメッセージ ID は、mail オブジェクトの headers および commonHeaders フィールドにあります。
source	メッセージが送信された E メールアドレス (エンベロープMAILFROMアドレス) 。
sourceArn	E メールの送信に使用された ID の Amazon リソースネーム (ARN) 。送信承認の場合、sourceArn は、ID 所有者が代理送信者が E メールの送信に使用することを承認した ID ARNの です。送信承認の詳細については、「E メールの認証方法」を参照してください。
sendingAccountId	E メールの送信に使用された AWS アカウントのアカウント ID。送信承認の場合、sendingAccountId は代理送信者のアカウント ID です。
destination	元のメールの受取人の E メールアドレスのリスト。
headersTruncated	通知でヘッダーが切り捨てられたかどうかを示す文字列。切り捨ては、ヘッダーが 10 KB

フィールド名	説明
	を超える場合に発生します。指定できる値は true および false です。
headers	E メールの元のヘッダーの一覧。リスト内の各ヘッダーには、name フィールドと value フィールドがあります。 **Note** headers フィールド内のメッセージ ID は、Amazon に渡した元のメッセージからのものですSES。Amazon が SESその後メッセージに割り当てたメッセージ ID は、 mail オブジェクトの messageId フィールドにあります。
commonHeaders	E メールの元の一般的に使用されるヘッダーのマッピング。 **Note** commonHeaders フィールド内のメッセージ ID は、Amazon がSESその後mailオブジェクトの messageId フィールドのメッセージに割り当てたメッセージ ID です。
tags	E メールに関連付けられたタグのリストです。

Bounce オブジェクト

Bounce イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
bounceType	Amazon によって決定されるバウンスのタイプSES。
bounceSubType	Amazon によって決定されるバウンスのサブタイプSES。
bouncedRecipients	バウンスとなった元のメールの受取人についての情報を含むリスト。
timestamp	がバウンス通知ISPを送信した日時ISO86 (YYYY-MM-DDThh:mm:ss.sZ) 。
feedbackId	バウンスの一意の ID。
reportingMTA	からの Reporting-MTA フィールドの値DSN。これは、 で説明されている配信、リレー、またはゲートウェイオペレーションを実行しようとした Message Transfer Authority (MTA) の値ですDSN。 **Note** このフィールドは、配信ステータス通知 (DSN) がバウンスにアタッチされた場合にのみ表示されます。

バウンスとなった受取人

バウンスイベントには、1 人の受信者に関するものと複数の受信者に関するものがあります。bouncedRecipients フィールドにはオブジェクトのリストが含まれ (バウンスを作成した E メールアドレスの受信者ごとに 1 つのオブジェクト)、以下のフィールドが含まれています。

フィールド名	説明
emailAddress	受取人の E メールアドレス。DSN が使用可能な場合、これは の Final-Recipient フィールドの値ですDSN。

オプションで、 DSNがバウンスにアタッチされている場合、次のフィールドが存在する場合もあります。

フィールド名	説明
action	からの Actionフィールドの値DSN。これは、この受信者にメッセージを配信しようとMTAした結果、レポートによって実行されたアクションを示します。
status	からの Statusフィールドの値DSN。これは、メッセージの配信状態を示す、受取人ごとに個別の、トランスポート独立型ステータスコードです。
diagnosticCode	レポート によって発行されたステータスコードMTA。これは、 の Diagnostic-Codeフィールドの値ですDSN。このフィールドは、 には存在しない場合があります DSN (したがって、 にも存在しない) JSON。

バウンスのタイプ

各バウンスイベントは、以下の表に示すいずれかのタイプになります。

イベント発行システムは、Amazon によって再試行されなくなったハードバウンスとソフトバウンスのみを公開しますSES。とマークされたバウンスを受け取ったらPermanent、対応する E メールアドレスをメーリングリストから削除する必要があります。今後送信することはできません。Transient バウンスは、メッセージのソフトバウンスが複数回発生し、Amazon SES が再配信を停

止したときに送信されます。最初は Transient バウンスとして返されても、将来的には同じアドレスに正常に再送できる場合があります。

bounceType	bounceSubType	説明
Undetermined	Undetermined	Amazon SES は特定のバウンス理由を特定できませんでした。
Permanent	General	Amazon は一般的なハードバウンスSESを受け取りました。このタイプのバウンスを受信した場合は、メーリングリストからこの受信者のメールアドレスを削除する必要があります。
Permanent	NoEmail	ターゲット E メールアドレスが存在しないため、Amazon は永続的なハードバウンスSESを受け取りました。このタイプのバウンスを受信した場合は、メーリングリストからこの受信者のメールアドレスを削除する必要があります。
Permanent	Suppressed	Amazon SES は、無効なアドレスとしてバウンスした最近の履歴があるため、このアドレスへの送信を抑制しました。グローバルサプレッションリストを上書きするには、「Amazon SESアカウントレベルのサプレッションリストの使用」を参照してください。
Permanent	OnAccountSuppressionList	Amazon SESは、アカウントレベルのサプレッションリストに含まれているため、このアドレスへの送信を抑制しました。これは、バウンス率のメトリクスに対してはカウントされません。
Transient	General	Amazon は一般的なバウンスSESを受け取りました。今後、この受信者に正常に再送できる可能性があります。

bounceType	bounceSubType	説明
Transient	MailboxFull	Amazon はメールボックスのフルバウンスSESを受け取りました。今後、この受信者に正常に再送できる可能性があります。
Transient	MessageTooLarge	Amazon は、バウンスが大きすぎるというメッセージSESを受信しました。メッセージサイズを小さくすることで、この受信者に正常に再送できる可能性があります。
Transient	CustomTimeoutExceeded	Amazon SESは、E メール送信者が指定した時間内に E メールを正常に配信できませんでした。(バウンスメッセージは、定義された 内で配信試行が失敗する可能性がある理由を指定します) TTL。
Transient	ContentRejected	Amazon はコンテンツ拒否バウンスSESを受け取りました。メッセージのコンテンツを変更することで、この受信者に正常に再送できる可能性があります。
Transient	AttachmentRejected	Amazon が添付ファイルの拒否バウンスSESを受け取りました。添付ファイルを削除または変更することで、この受信者に正常に再送できる可能性があります。

苦情のオブジェクト

Complaint イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
complainedRecipients	苦情を送信した可能性がある受取人に関する情報を含むリスト。
timestamp	が苦情通知ISPを送信した日時 (ISO86-YYYYMM-DDThh:mm:ss.sZ)。

フィールド名	説明
feedbackId	苦情の一意の ID。
complaintSubType	Amazon によって決定される苦情のサブタイプ SES。

また、フィードバックレポートが苦情に添付されている場合、以下のフィールドが示される場合があります。

フィールド名	説明
userAgent	フィードバックレポートの User-Agent フィールドの値。これは、レポートを生成したシステムの名前とバージョンを示します。
complaintFeedbackType	から受け取ったフィードバックレポートの Feedback-Type フィールドの値ISP。これには、フィードバックのタイプが含まれます。
arrivalDate	フィードバックレポートの Arrival-Date または Received-Date フィールドの値は ISO8601 形式 (YYYY-MM-DDThh:mm:ss.sZ) です。このフィールドはレポートにない場合があります (したがって、 にもないJSON) 。

苦情を申告した受取人

complainedRecipients フィールドには、苦情の送信元と思われる受信者のリストが含まれます。

> **Important**
>
> ほとんどの は、苦情を送信する受信者の E メールアドレスISPsを編集します。このため、complainedRecipients フィールドには、苦情通知を発行したドメインにアドレスが含まれていて、E メールが送信された全員のリストが含まれます。

JSON このリストの オブジェクトには、次のフィールドが含まれます。

フィールド名	説明
emailAddress	受取人の E メールアドレス。

苦情のタイプ

Internet Assigned Numbers Authority のウェブサイトによるとISP、レポート によって割り当てられた complaintFeedbackTypeフィールドに次の苦情タイプが表示される場合があります。

フィールド名	説明
abuse	迷惑メールまたはその他のメール不正使用。
auth-failure	E メールの認証障害レポート
fraud	なんらかの詐欺またはフィッシング行為を示します。
not-spam	レポートの提供者がこのメッセージをスパムではないと見なしていることを示します。このタイプは、誤ってスパムとしてタグ付けまたは分類されたメッセージを修正するために使用される場合があります。
other	その他の登録されたタイプに該当しないフィードバックを示します。
virus	元のメッセージでウイルスが見つかったことを示します。

苦情のサブタイプ

complaintSubType フィールドの値は、null または OnAccountSuppressionList のいずれかになります。値が の場合OnAccountSuppressionList、Amazon はメッセージSESを受け入れましたが、アカウントレベルのサプレッションリストに含まれていたため、送信を試みませんでした。

配信オブジェクト

Delivery イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
timestamp	Amazon が受信者のメールサーバーに E メールをSES配信した日時を ISO8601 形式 (YYYY-MM-DDThh:mm:ss.sZ) で表します。
processingTimeMillis	Amazon が送信者からのリクエストSESを受け入れてから、Amazon が受信者のメールサーバーにメッセージをSES渡したまでの時間をミリ秒単位で表します。
recipients	配信イベントの適用対象となる受信者のリスト。
smtpResponse	Amazon からの E メールを受け入れISPたリモート のSMTPレスポンスメッセージSES。このメッセージは、E メール、メールサーバー、および の受信によって異なりますISP。
reportingMTA	SES メールを送信した Amazon メールサーバーのホスト名。

Send オブジェクト

send イベントに関する情報を含むJSONオブジェクトは常に空です。

Reject オブジェクト

Reject イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
reason	メールが拒否された理由。可能な値は のみです。つまりBad content、Amazon は E メールにウイルスが含まれていることをSES検出し

フィールド名	説明
	ました。メッセージが拒否されると、Amazon はメッセージの処理SESを停止し、受信者のメールサーバーへの配信を試みません。

Open オブジェクト

Open イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
ipAddress	受信側の IP アドレス
timestamp	オープンイベントが ISO8601 形式で発生した日時 (YYYY-MM-DDThh:mm:ss.sZ) 。
userAgent	受取人がメールを開くために使用したしたデバイスまたは E メールクライアントのユーザーエージェント。

Click オブジェクト

Click イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
ipAddress	受信側の IP アドレス
timestamp	クリックイベントが発生した日時は ISO8601 形式 (YYYY-MM-DDThh:mm:ss.sZ) です。
userAgent	受信者が E メールのリンクをクリックするために使用したクライアントのユーザーエージェント。
link	受信者がクリックしたリンクURLの 。

フィールド名	説明
linkTags	ses:tags 属性を使用してリンクに追加されたタグのリスト。メール内のリンクにタグを追加する方法については、「Amazon SES E メール送信メトリクスに関するよくある質問」の「Q5. リンクに一意の識別子をタグ付けできますか?」を参照してください。

レンダリング失敗オブジェクト

Rendering Failure イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
templateName	E メールの送信に使用されるテンプレートの名前。
errorMessage	レンダリング失敗に関する詳細情報を提供するメッセージ。

DeliveryDelay オブジェクト

DeliveryDelay イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
delayType	遅延のタイプ。可能な値は以下のとおりです。 • InternalFailure – Amazon の内部SESの問題により、メッセージが遅延しました。 • 全般 — SMTP会話中に一般的な障害が発生しました。

フィールド名	説明
	• MailboxFull – 受信者のメールボックスがいっぱいで、追加のメッセージを受信できません。 • SpamDetected – 受信者のメールサーバーが、アカウントから大量の未承諾 E メールを検出しました。 • RecipientServerError – 受信者の E メールサーバーに一時的な問題が発生し、メッセージの配信が妨げられています。 • IPFailure – メッセージを送信している IP アドレスが、受信者の E メールプロバイダーによってブロックまたはスロットリングされています。 • TransientCommunicationFailure – 受信者の E メールプロバイダーとのSMTP会話中に一時的な通信障害が発生しました。 • BYOIPHostNameLookupUnavailable — Amazon SES は IP アドレスのDNSホスト名を検索できませんでした。このタイプの遅延は、Bring Your Own IP を使用する場合にのみ発生します。 • Undetermined – Amazon SESは配信遅延の理由を特定できませんでした。 • SendingDeferral – Amazon SESは、メッセージを内部的に延期することが適切であると判断しました。
`delayedRecipients`	E メールの受取人に関する情報を含むオブジェクト。
`expirationTime`	Amazon SESがメッセージの配信を停止する日時。この値は 8601 ISO 形式で表示されます。

フィールド名	説明
reportingMTA	遅延を報告した Message Transfer Agent (MTA) の IP アドレス。
timestamp	遅延が発生した日時。8601 ISO 形式で表示されます。

遅延受取人

delayedRecipients オブジェクトには、以下の値が含まれています。

フィールド名	説明
emailAddress	メッセージの配信が遅れる原因となった E メールアドレス。
status	配信遅延に関連付けられたSMTPステータスコード。
diagnosticCode	受信側 Message Transfer Agent (MTA) によって提供される診断コード。

サブスクリプションオブジェクト

Subscription イベントに関する情報を含む JSON オブジェクトには、次のフィールドがあります。

フィールド名	説明
contactList	連絡先が含まれているリストの名前。
timestamp	がサブスクリプション通知ISPを送信したときの ISO8601 形式（ YYYY-MM-DDThh:mm:ss.sZ) の日時。

フィールド名	説明
source	メッセージが送信された E メールアドレス (エンベロープMAILFROMアドレス) 。
newTopicPreferences	データ構造 (マップ) JSON。連絡先リスト内のすべてのトピックのサブスクリプションステータスを指定し、変更後のステータスを示します (連絡先がサブスクライブまたはサブスクライブ解除) 。
oldTopicPreferences	データ構造 (マップ) JSON。連絡先リスト内のすべてのトピックのサブスクリプションステータスを指定し、変更前のステータス (連絡先がサブスクライブまたはサブスクライブ解除) を示します。

新旧トピックの設定

newTopicPreferences および oldTopicPreferences オブジェクトには、以下の値が含まれています。

フィールド名	説明
unsubscribeAll	連絡先リストのすべてのトピックから、連絡先がサブスクライブ解除されているかどうかを指定します。
topicSubscriptionStatus	指定されたイベントタイプの から通知を受信するように現在サブスクライブされているかどうかを示す topicName フィールドで、トピックSESのサブスクリプションステータスを指定します。指定できる値は、 subscriptionStatus フィールドの OptIn (サブスクライブ済み) または OptOut (サブスクライブ解除済み) です。

フィールド名	説明
topicDefaultSubscriptionStatus	イベント送信先に追加された新しいトピックをデフォルトでサブスクライブするか、サブスクライブ解除するかを決定する topicName フィールドで、トピックのデフォルトのサブスクリプションステータスを指定します。指定できる値は、 subscriptionStatus フィールドの OptIn (デフォルトでサブスクライブ) または OptOut (デフォルトでサブスクライブ解除) です。

Amazon SES が Amazon SNS に発行するイベントデータの例

このセクションでは、Amazon SES から Amazon SNS に発行するメール送信イベントレコードの例をイベントタイプ別に示します。

このセクションのトピック:

- バウンスレコード
- 苦情レコード
- 配信レコード
- Send レコード
- Reject レコード
- Open レコード
- Click レコード
- レンダリング失敗レコード
- DeliveryDelay レコード
- サブスクリプションレコード

> **Note**
>
> 次の例では、tag フィールドが使用されていて、すべてのイベントタイプについて、SES がタグの発行をサポートする設定セットを通じて、イベント発行を使用しています。ID で

フィードバック通知を直接使用する場合、SES はタグを発行しません。設定セットの作成または設定セットの変更をする場合、タグの追加について参照してください。

バウンスレコード

Amazon SES が Amazon SNS に発行する Bounce イベントレコードの例を以下に示しています。

```
{
  "eventType":"Bounce",
  "bounce":{
    "bounceType":"Permanent",
    "bounceSubType":"General",
    "bouncedRecipients":[
      {
        "emailAddress":"recipient@example.com",
        "action":"failed",
        "status":"5.1.1",
        "diagnosticCode":"smtp; 550 5.1.1 user unknown"
      }
    ],
    "timestamp":"2017-08-05T00:41:02.669Z",
    "feedbackId":"01000157c44f053b-61b59c11-9236-11e6-8f96-7be8aexample-000000",
    "reportingMTA":"dsn; mta.example.com"
  },
  "mail":{
    "timestamp":"2017-08-05T00:40:02.012Z",
    "source":"Sender Name <sender@example.com>",
    "sourceArn":"arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId":"123456789012",
    "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination":[
      "recipient@example.com"
    ],
    "headersTruncated":false,
    "headers":[
      {
        "name":"From",
        "value":"Sender Name <sender@example.com>"
      },
      {
        "name":"To",
        "value":"recipient@example.com"
```

```
      },
      {
        "name":"Subject",
        "value":"Message sent from Amazon SES"
      },
      {
        "name":"MIME-Version",
        "value":"1.0"
      },
      {
        "name":"Content-Type",
        "value":"multipart/alternative; boundary=\"----
=_Part_7307378_1629847660.1516840721503\""
      }
    ],
    "commonHeaders":{
      "from":[
        "Sender Name <sender@example.com>"
      ],
      "to":[
        "recipient@example.com"
      ],
      "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject":"Message sent from Amazon SES"
    },
    "tags":{
      "ses:configuration-set":[
        "ConfigSet"
      ],
      "ses:source-ip":[
        "192.0.2.0"
      ],
      "ses:from-domain":[
        "example.com"
      ],
      "ses:caller-identity":[
        "ses_user"
      ]
    }
  }
}
```

苦情レコード

Amazon SES が Amazon SNS に発行する Complaint イベントレコードの例を以下に示しています。

```
{
  "eventType":"Complaint",
  "complaint": {
    "complainedRecipients":[
      {
        "emailAddress":"recipient@example.com"
      }
    ],
    "timestamp":"2017-08-05T00:41:02.669Z",
    "feedbackId":"01000157c44f053b-61b59c11-9236-11e6-8f96-7be8aexample-000000",
    "userAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/60.0.3112.90 Safari/537.36",
    "complaintFeedbackType":"abuse",
    "arrivalDate":"2017-08-05T00:41:02.669Z"
  },
  "mail":{
    "timestamp":"2017-08-05T00:40:01.123Z",
    "source":"Sender Name <sender@example.com>",
    "sourceArn":"arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId":"123456789012",
    "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination":[
      "recipient@example.com"
    ],
    "headersTruncated":false,
    "headers":[
      {
        "name":"From",
        "value":"Sender Name <sender@example.com>"
      },
      {
        "name":"To",
        "value":"recipient@example.com"
      },
      {
        "name":"Subject",
        "value":"Message sent from Amazon SES"
      },
      {
```

```
        "name":"MIME-Version","value":"1.0"
      },
      {
        "name":"Content-Type",
        "value":"multipart/alternative; boundary=\"----
=_Part_7298998_679725522.1516840859643\""
      }
    ],
    "commonHeaders":{
      "from":[
        "Sender Name <sender@example.com>"
      ],
      "to":[
        "recipient@example.com"
      ],
      "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject":"Message sent from Amazon SES"
    },
    "tags":{
      "ses:configuration-set":[
        "ConfigSet"
      ],
      "ses:source-ip":[
        "192.0.2.0"
      ],
      "ses:from-domain":[
        "example.com"
      ],
      "ses:caller-identity":[
        "ses_user"
      ]
    }
  }
}
```

配信レコード

Amazon SES が Amazon SNS に発行する `Delivery` イベントレコードの例を以下に示しています。

```
{
  "eventType": "Delivery",
  "mail": {
```

```
    "timestamp": "2016-10-19T23:20:52.240Z",
    "source": "sender@example.com",
    "sourceArn": "arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId": "123456789012",
    "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination": [
      "recipient@example.com"
    ],
    "headersTruncated": false,
    "headers": [
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
        "value": "text/html; charset=UTF-8"
      },
      {
        "name": "Content-Transfer-Encoding",
        "value": "7bit"
      }
    ],
    "commonHeaders": {
      "from": [
        "sender@example.com"
      ],
      "to": [
        "recipient@example.com"
      ],
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject": "Message sent from Amazon SES"
```

```
    },
    "tags": {
      "ses:configuration-set": [
        "ConfigSet"
      ],
      "ses:source-ip": [
        "192.0.2.0"
      ],
      "ses:from-domain": [
        "example.com"
      ],
      "ses:caller-identity": [
        "ses_user"
      ],
      "ses:outgoing-ip": [
        "192.0.2.0"
      ],
      "myCustomTag1": [
        "myCustomTagValue1"
      ],
      "myCustomTag2": [
        "myCustomTagValue2"
      ]
    }
  },
  "delivery": {
    "timestamp": "2016-10-19T23:21:04.133Z",
    "processingTimeMillis": 11893,
    "recipients": [
      "recipient@example.com"
    ],
    "smtpResponse": "250 2.6.0 Message received",
    "reportingMTA": "mta.example.com"
  }
}
```

Send レコード

Amazon SES が Amazon SNS に発行する Send イベントレコードの例を以下に示しています。一部のフィールドは、常に存在するわけではありません。例えばテンプレート化された E メールの場合、件名は後でレンダリングされ、後続のイベントに含まれています。

```
{
```

```
  "eventType": "Send",
  "mail": {
    "timestamp": "2016-10-14T05:02:16.645Z",
    "source": "sender@example.com",
    "sourceArn": "arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId": "123456789012",
    "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination": [
      "recipient@example.com"
    ],
    "headersTruncated": false,
    "headers": [
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
        "value": "multipart/mixed;  boundary=\"----=_Part_0_716996660.1476421336341\""
      },
      {
        "name": "X-SES-MESSAGE-TAGS",
        "value": "myCustomTag1=myCustomTagValue1, myCustomTag2=myCustomTagValue2"
      }
    ],
    "commonHeaders": {
      "from": [
        "sender@example.com"
      ],
      "to": [
        "recipient@example.com"
      ],
```

```
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject": "Message sent from Amazon SES"
    },
    "tags": {
      "ses:configuration-set": [
        "ConfigSet"
      ],
      "ses:source-ip": [
        "192.0.2.0"
      ],
      "ses:from-domain": [
        "example.com"
      ],
      "ses:caller-identity": [
        "ses_user"
      ],
      "myCustomTag1": [
        "myCustomTagValue1"
      ],
      "myCustomTag2": [
        "myCustomTagValue2"
      ]
    }
  },
  "send": {}
}
```

Reject レコード

Amazon SES が Amazon SNS に発行する `Reject` イベントレコードの例を以下に示しています。

```
{
  "eventType": "Reject",
  "mail": {
    "timestamp": "2016-10-14T17:38:15.211Z",
    "source": "sender@example.com",
    "sourceArn": "arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId": "123456789012",
    "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination": [
      "sender@example.com"
    ],
    "headersTruncated": false,
    "headers": [
```

```
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
        "value": "multipart/mixed; boundary=\"qMm9M+Fa2AknHoGS\""
      },
      {
        "name": "X-SES-MESSAGE-TAGS",
        "value": "myCustomTag1=myCustomTagValue1, myCustomTag2=myCustomTagValue2"
      }
    ],
    "commonHeaders": {
      "from": [
        "sender@example.com"
      ],
      "to": [
        "recipient@example.com"
      ],
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject": "Message sent from Amazon SES"
    },
    "tags": {
      "ses:configuration-set": [
        "ConfigSet"
      ],
      "ses:source-ip": [
        "192.0.2.0"
      ],
      "ses:from-domain": [
        "example.com"
```

```
      ],
      "ses:caller-identity": [
        "ses_user"
      ],
      "myCustomTag1": [
        "myCustomTagValue1"
      ],
      "myCustomTag2": [
        "myCustomTagValue2"
      ]
    }
  },
  "reject": {
    "reason": "Bad content"
  }
}
```

Open レコード

Amazon SES が Amazon SNS に発行する Open イベントレコードの例を以下に示します。

```
{
  "eventType": "Open",
  "mail": {
    "commonHeaders": {
      "from": [
        "sender@example.com"
      ],
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject": "Message sent from Amazon SES",
      "to": [
        "recipient@example.com"
      ]
    },
    "destination": [
      "recipient@example.com"
    ],
    "headers": [
      {
        "name": "X-SES-CONFIGURATION-SET",
        "value": "ConfigSet"
      },
      {
        "name":"X-SES-MESSAGE-TAGS",
```

```
        "value":"myCustomTag1=myCustomValue1, myCustomTag2=myCustomValue2"
      },
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
        "value": "multipart/alternative; boundary=\"XBoundary\""
      }
    ],
    "headersTruncated": false,
    "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "sendingAccountId": "123456789012",
    "source": "sender@example.com",
    "tags": {
      "myCustomTag1":[
        "myCustomValue1"
      ],
      "myCustomTag2":[
        "myCustomValue2"
      ],
      "ses:caller-identity": [
        "IAM_user_or_role_name"
      ],
      "ses:configuration-set": [
        "ConfigSet"
      ],
      "ses:from-domain": [
        "example.com"
      ],
      "ses:source-ip": [
```

```
        "192.0.2.0"
      ]
    },
    "timestamp": "2017-08-09T21:59:49.927Z"
  },
  "open": {
    "ipAddress": "192.0.2.1",
    "timestamp": "2017-08-09T22:00:19.652Z",
    "userAgent": "Mozilla/5.0 (iPhone; CPU iPhone OS 10_3_3 like Mac OS X)
 AppleWebKit/603.3.8 (KHTML, like Gecko) Mobile/14G60"
  }
}
```

Click レコード

Amazon SES が Amazon SNS に発行する Click イベントレコードの例を以下に示しています。

```
{
  "eventType": "Click",
  "click": {
    "ipAddress": "192.0.2.1",
    "link": "http://docs.aws.amazon.com/ses/latest/DeveloperGuide/send-email-
smtp.html",
    "linkTags": {
      "samplekey0": [
        "samplevalue0"
      ],
      "samplekey1": [
        "samplevalue1"
      ]
    },
    "timestamp": "2017-08-09T23:51:25.570Z",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/60.0.3112.90 Safari/537.36"
  },
  "mail": {
    "commonHeaders": {
      "from": [
        "sender@example.com"
      ],
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "subject": "Message sent from Amazon SES",
      "to": [
        "recipient@example.com"
```

```
      ]
    },
    "destination": [
      "recipient@example.com"
    ],
    "headers": [
      {
        "name": "X-SES-CONFIGURATION-SET",
        "value": "ConfigSet"
      },
      {
        "name":"X-SES-MESSAGE-TAGS",
        "value":"myCustomTag1=myCustomValue1, myCustomTag2=myCustomValue2"
      },
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
        "value": "multipart/alternative; boundary=\"XBoundary\""
      },
      {
        "name": "Message-ID",
        "value": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000"
      }
    ],
    "headersTruncated": false,
    "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "sendingAccountId": "123456789012",
    "source": "sender@example.com",
    "tags": {
```

```
      "myCustomTag1":[
        "myCustomValue1"
      ],
      "myCustomTag2":[
        "myCustomValue2"
      ],
      "ses:caller-identity": [
        "ses_user"
      ],
      "ses:configuration-set": [
        "ConfigSet"
      ],
      "ses:from-domain": [
        "example.com"
      ],
      "ses:source-ip": [
        "192.0.2.0"
      ]
    },
    "timestamp": "2017-08-09T23:50:05.795Z"
  }
}
```

レンダリング失敗レコード

Amazon SES が Amazon SNS に発行する `Rendering Failure` イベントレコードの例を以下に示しています。

```
{
  "eventType":"Rendering Failure",
  "mail":{
    "timestamp":"2018-01-22T18:43:06.197Z",
    "source":"sender@example.com",
    "sourceArn":"arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId":"123456789012",
    "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination":[
      "recipient@example.com"
    ],
    "headersTruncated":false,
    "tags":{
      "ses:configuration-set":[
        "ConfigSet"
```

```
      ]
    }
  },
  "failure":{
    "errorMessage":"Attribute 'attributeName' is not present in the rendering data.",
    "templateName":"MyTemplate"
  }
}
```

DeliveryDelay レコード

Amazon SES が Amazon SNS に発行する DeliveryDelay イベントレコードの例を以下に示しています。

```
{
  "eventType": "DeliveryDelay",
  "mail":{
    "timestamp":"2020-06-16T00:15:40.641Z",
    "source":"sender@example.com",
    "sourceArn":"arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId":"123456789012",
    "messageId":"EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
    "destination":[
      "recipient@example.com"
    ],
    "headersTruncated":false,
    "tags":{
      "ses:configuration-set":[
        "ConfigSet"
      ]
    }
  },
  "deliveryDelay": {
    "timestamp": "2020-06-16T00:25:40.095Z",
    "delayType": "TransientCommunicationFailure",
    "expirationTime": "2020-06-16T00:25:40.914Z",
    "delayedRecipients": [{
      "emailAddress": "recipient@example.com",
      "status": "4.4.1",
      "diagnosticCode": "smtp; 421 4.4.1 Unable to connect to remote host"
    }]
  }
}
```

サブスクリプションレコード

Amazon SES が Firehose に発行する Subscription イベントレコードの例は、以下のとおりです。

```
{
  "eventType": "Subscription",
  "mail": {
    "timestamp": "2022-01-12T01:00:14.340Z",
    "source": "sender@example.com",
    "sourceArn": "arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
    "sendingAccountId": "123456789012",
    "messageId": "EXAMPLEe4bccb684-777bc8de-afa7-4970-92b0-f515137b1497-000000",
    "destination": ["recipient@example.com"],
    "headersTruncated": false,
    "headers": [
      {
        "name": "From",
        "value": "sender@example.com"
      },
      {
        "name": "To",
        "value": "recipient@example.com"
      },
      {
        "name": "Subject",
        "value": "Message sent from Amazon SES"
      },
      {
        "name": "MIME-Version",
        "value": "1.0"
      },
      {
        "name": "Content-Type",
        "value": "text/html; charset=UTF-8"
      },
      {
        "name": "Content-Transfer-Encoding",
        "value": "7bit"
      }
    ],
    "commonHeaders": {
      "from": ["sender@example.com"],
      "to": ["recipient@example.com"],
```

```
      "messageId": "EXAMPLEe4bccb684-777bc8de-afa7-4970-92b0-f515137b1497-000000",
      "subject": "Message sent from Amazon SES"
    },
    "tags": {
      "ses:operation": ["SendEmail"],
      "ses:configuration-set": ["ConfigSet"],
      "ses:source-ip": ["192.0.2.0"],
      "ses:from-domain": ["example.com"],
      "ses:caller-identity": ["ses_user"],
      "myCustomTag1": ["myCustomValue1"],
      "myCustomTag2": ["myCustomValue2"]
    }
  },
  "subscription": {
    "contactList": "ContactListName",
    "timestamp": "2022-01-12T01:00:17.910Z",
    "source": "UnsubscribeHeader",
    "newTopicPreferences": {
      "unsubscribeAll": true,
      "topicSubscriptionStatus": [
        {
          "topicName": "ExampleTopicName",
          "subscriptionStatus": "OptOut"
        }
      ]
    },
    "oldTopicPreferences": {
      "unsubscribeAll": false,
      "topicSubscriptionStatus": [
        {
          "topicName": "ExampleTopicName",
          "subscriptionStatus": "OptOut"
        }
      ]
    }
  }
}
```

Amazon SES 送信者評価のモニタリング

Amazon SES では、送信者としての評価にダメージを与える可能性のあるメトリクスや、メール配信率の低下を招く可能性のあるメトリクスなど、複数のメトリクスがアクティブに追跡されます。このプロセスで監視する 2 つの重要なメトリクスは、アカウントのバウンス率と苦情率です。アカウントのバウンス率や苦情率が高すぎる場合、アカウントを確認し、アカウントの E メール送信機能を一時停止することがあります。

アカウントの状態にとって返送率および苦情率が非常に重要であるため、Amazon SES consoleにはこれらのメトリクスを追跡するための評価メトリクスが用意されています。評価メトリクスでは、バウンスや苦情に関係なく送信者の評価にダメージを与える可能性のある要因に関する情報も表示できます。たとえば、既知のスパムトラップにメールを送信すると、このダッシュボードにメッセージが表示されます。

このセクションには、評価メトリックへのアクセス、そこに含まれる情報の解釈、および送信者の評価に影響を与える可能性のある要因を積極的に通知するシステム設定に関する情報が含まれています。

このセクションでは、以下のトピックについて説明します。

- 評価メトリクスを使用して返送率と苦情率を追跡する
- 評価メトリクスのメッセージ
- CloudWatch を使用して評価モニタリングアラームを作成する
- SNDS 専用 の メトリクス IPs
- E メール送信の自動的な一時停止

評価メトリクスを使用して返送率と苦情率を追跡する

評価メトリクスコンソールページには、Amazon SES チームが個人アカウントの状態を判断する際に確認するものと同じ情報が含まれています。

評価メトリクスを表示するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 画面左側のナビゲーションペインで、[Reputation metrics]を選択します。

   ダッシュボードには、以下の内容が表示されます。

- アカウントのステータス — 返送率と苦情率の合計ヘルス状態のサマリー。可能な値は以下のとおりです:
  - [正常] - 現在、アカウントに影響する問題はありません。
  - [Under review] - お客様のアカウントはレビュー対象です。アカウントのレビュー対象となった原因の問題が、レビュー期間の終了までに修正されなかった場合、アカウントの E メール送信が一時停止される可能性があります。
  - [Pending end of review decision] - お客様のアカウントはレビュー対象です。お客様のアカウントをレビュー対象とした問題の内容により、何らかの処置を行う前に、手動でレビューを実行する必要があります。
  - [Sending paused (送信一時停止)] - アカウントの E メール送信機能を一時停止しました。アカウントが一時停止されている間は、Amazon SES を使用して E メールを送信することはできません。この決定の見直しをリクエストできます。レビューをリクエストする方法の詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。
  - [Pending sending pause (送信一時停止保留中)] - お客様のアカウントはレビュー対象です。アカウントをレビュー対象とした問題は解決されていません。このような状況では、通常アカウントの E メール送信機能を一時停止します。ただし、アカウントの性質により、何らかの処置を行う前に、アカウントの確認が行われます。
- [Bounce Rate] - アカウントから送信され、ハードバウンスとなった E メールのパーセンテージ。「バウンス率の計算方法」を参照してください。
- [Complaint Rate] – アカウントから送信され、受信者によってスパムと報告されたメールのパーセンテージ。「苦情率の計算方法」を参照してください。

> **Note**
>
> [Bounce Rate] セクションおよび [Complaint Rate] セクションには、それぞれのメトリクスに関するステータスメッセージも含まれています。以下は、これらに対してメトリクスについて表示される可能性のあるステータスメッセージのリストです。
>
> - Healthy - メトリクスは通常のレベルです。
> - Almost healed - メトリクスにより、アカウントがレビュー対象になっています。レビュー期間が開始してから、メトリクスは最大レート未満を維持しています。メトリクスが最大レート未満のままであれば、レビュー期間が終了する前にこのメトリクスのステータスが Healthy に変化する可能性があります。

- Under review - メトリクスによってアカウントがレビュー対象になり、メトリクスはまだ最大レートを超過しています。メトリクスが最大レートを超過した問題が、レビュー期間の終了までに修正されなかった場合、アカウントの E メール送信が一時停止される可能性があります。
- Sending pause - メトリクスにより、アカウントの E メール送信機能が一時停止されます。アカウントが一時停止されている間は、Amazon SES を使用して E メールを送信することはできません。この決定の見直しをリクエストできます。レビューのリクエストを送信する方法の詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。
- [Pending sending pause (送信一時停止保留中)] - メトリクスによりアカウントがレビューされています。このレビューの原因となった問題が、解決していません。この問題により、アカウントの E メール送信機能が一時停止される可能性があります。その先のアクションが発生する前に、Amazon SES チームのメンバーによってアカウントの確認が行われます。

- [Other Notifications (その他の通知)] - 返送や苦情に関係のない評価関連の問題がアカウントで発生している場合、ここに短いメッセージが表示されます。ここに表示される可能性のある通知の詳細については、「評価メトリクスのメッセージ」を参照してください。

評価メトリクスのメッセージ

Amazon SES 評価メトリクスコンソールページには、アカウントに関する重要なメトリクスが表示されます。以下のセクションでは、このダッシュボードに表示される可能性のあるメッセージについて説明し、送信者評価に関連する問題の解決に役立つ可能性があるヒントと使用できる情報を示します。

このセクションでは、以下のタイプの通知についての情報を提供します。

- ステータスメッセージ
- バウンス率の通知
- 苦情率の通知
- アンチスパム組織の通知
- リスト型攻撃通知
- 直接フィードバック通知
- ドメインブロックリスト通知

- 内部レビュー通知
- メールボックスプロバイダー通知
- 受信者フィードバック通知
- 関連アカウント通知
- スパムトラップ通知
- 脆弱サイト通知
- 認証情報の漏洩に関する通知
- その他の通知

ステータスメッセージ

評価メトリクスコンソールページを使用する際に、Amazon SES アカウントのステータスを表すメッセージが表示されます。表示される可能性のあるアカウントステータス値の一覧を以下に示します。

- [正常] - 現在、アカウントに影響する問題はありません。
- [Under review] - お客様のアカウントはレビュー対象です。アカウントのレビュー対象となった原因の問題が、レビュー期間の終了までに修正されなかった場合、アカウントの E メール送信が一時停止される可能性があります。
- [Pending end of review decision] - お客様のアカウントはレビュー対象です。お客様のアカウントをレビュー対象とした問題の内容により、何らかの処置を行う前に、手動でレビューを実行する必要があります。
- [Sending paused (送信一時停止)] - アカウントの E メール送信機能を一時停止しました。アカウントが一時停止されている間は、Amazon SES を使用して E メールを送信することはできません。この決定の見直しをリクエストできます。レビューをリクエストする方法の詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。
- [Pending sending pause (送信一時停止保留中)] - お客様のアカウントはレビュー対象です。アカウントをレビュー対象とした問題は解決されていません。このような状況では、通常アカウントの E メール送信機能を一時停止します。ただし、アカウントの性質により、何らかの処置を行う前に、アカウントの確認が行われます。

さらに、評価メトリクスページの [Bounce Rate] セクションと [Complaint Rate] セクションには、それぞれのメトリクスのステータス要約が表示されます。表示される可能性のあるメトリクスのステータス値の一覧を以下に示します。

- Healthy - メトリクスは通常のレベルです。
- Almost healed - メトリクスにより、アカウントがレビュー対象になっています。レビュー期間が開始してから、メトリクスは最大レート未満を維持しています。メトリクスが最大レート未満のままであれば、レビュー期間が終了する前にこのメトリクスのステータスが Healthy に変化する可能性があります。
- Under review - メトリクスによってアカウントがレビュー対象になり、メトリクスはまだ最大レートを超過しています。メトリクスが最大レートを超過した問題が、レビュー期間の終了までに修正されなかった場合、アカウントの E メール送信が一時停止される可能性があります。
- Sending pause - メトリクスにより、アカウントの E メール送信機能が一時停止されます。アカウントが一時停止されている間は、Amazon SES を使用して E メールを送信することはできません。この決定の見直しをリクエストできます。レビューのリクエストを送信する方法の詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。
- [Pending sending pause (送信一時停止保留中)] - メトリクスによりアカウントがレビューされています。このレビューの原因となった問題が、解決していません。この問題により、アカウントの E メール送信機能が一時停止される可能性があります。その先のアクションが発生する前に、Amazon SES チームのメンバーによってアカウントの確認が行われます。

バウンス率の通知

このセクションでは、Amazon SES の評価メトリクスページに表示されるバウンス率通知についての追加情報を提供します。

この通知を受け取った理由

この通知は、アカウントのバウンス率が高すぎたために受け取りました。バウンス率は、Amazon SES アカウントによって生成されたハードバウンスの数に基づいています。E メールプロバイダーでは、バウンス率の高さは、送信者が受取人リストを適切に管理しておらず、未承諾 E メールを送信している可能性を示すサインとみなします。

ハードバウンスは、存在しないアドレスに E メールが送信されると発生します。Amazon SES はこの計算においてソフトバウンス (受信者のアドレスで一時的にメッセージを受信できない場合に発生するもの) を考慮しません。確認済みのアドレスおよびドメインに送信するバウンスメールや、Amazon SES インボックスシミュレーターに送信するメールも、この計算に反映されません。

バウンス率は、E メールの代表ボリューム に基づいて計算されます。代表ボリュームは、通常の送信行為を表す E メールの量です。大量のメールの送信者と少量のメールの送信者を公平に扱うため

に、代表ボリュームはアカウントごとに異なっており、アカウントの送信パターンの変化にともなって代表ボリュームも変わります。

最良の結果を得るには、バウンス率を 5% 未満に保ちます。これより高いバウンス率は、E メールの配信に影響する可能性があります。バウンス率が 5% 以上になると、アカウントは自動的にレビュー対象になります。バウンス率が 10% を超える場合は、高いバウンス率の原因となった問題が解決するまで、お客様のアカウントの以後の E メール送信を一時停止することがあります。

問題を解決する方法

バウンスおよび苦情をキャプチャおよび管理するためのプロセスを正しく配備してください (まだの場合)。Amazon SES アカウントはすべて、これらのプロセスを正しく配備する必要があります。詳細については、「E メールプログラムの成功のメトリクス」を参照してください。

次に、どのメールアドレスでバウンスが発生しているか特定し、バウンスの発生を抑えて排除するためのプランを作成および実施します。アカウントの E メール送信機能がすでに一時停止している場合は、AWS Management Console にサインインし、AWS サポート に移動します。代理でオープンしたケースに返信してください。

お客様のアカウントがレビュー対象になっている場合

レビュー期間の終了時にアカウントのバウンス率が 10% を超えている場合、問題が解決されるまで、お客様のアカウントの E メール送信機能を一時停止することがあります。

問題解決のための変更を実施した場合は、 AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。ケースへのレスポンスで、実装した変更について説明します。この変更によってバウンス率が低下すると見なされた場合は、変更の実施後に受信するバウンスのみが含まれるように計算が調整されます。

アカウントの E メール送信機能が一時停止された場合

この決定の再検討をリクエストできます。詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

問題解決のための変更を実施する場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メールには、この問題の解決のために実行したアクションの詳細と、この問題を再度発生させないための計画の詳細を記入してください。リクエストを受け取った後、ご提供いただいた情報を確認し、必要に応じてアカウントのステータスを変更します。

苦情率の通知

このセクションでは、Amazon SES の評価メトリクスページに表示される苦情率通知についての追加情報を提供します。

この通知を受け取った理由

この通知は、アカウントの苦情率が高すぎたために受け取りました。苦情率は、Amazon SES アカウントによって生成された苦情の数に基づいています。E メールプロバイダーでは、苦情率の高さは、送信者が受取人リストを適切に管理しておらず、未承諾 E メールを送信している可能性を示すサインとみなします。

苦情は、送信した E メールを受信者がスパムと認識した場合に発生します。これは通常、受信者が E メールクライアントの [スパムを報告] ボタンを使用する場合に発生します。Amazon SES メールボックスシミュレーターに送信する E メールによって生成された苦情は、この計算では考慮されません。

苦情率は、E メールの代表ボリューム に基づいて計算されます。代表ボリュームは、通常の送信行為を表す E メールの量です。大量のメールの送信者と少量のメールの送信者を公平に扱うために、代表ボリュームはアカウントごとに異なっており、アカウントの送信パターンの変化にともなって代表ボリュームも変わります。

最良の結果を得るには、苦情率を 0.1% 未満に維持してください。これより高い苦情率は、E メールの配信に影響する可能性があります。苦情率が 0.1% 以上になると、アカウントは自動的にレビュー対象になります。苦情率が 0.5% を超える場合は、高い苦情率の原因となった問題が解決するまで、お客様のアカウントの以後の E メール送信を一時停止することがあります。

問題を解決する方法

バウンスおよび苦情をキャプチャおよび管理するためのプロセスを正しく配備してください (まだの場合)。Amazon SES アカウントはすべて、これらのプロセスを正しく配備する必要があります。詳細については、「E メールプログラムの成功のメトリクス」を参照してください。

次に、送信したどのメッセージが苦情を招いたか特定し、このような苦情を減少させるためのプランを実施します。アカウントの E メール送信機能がすでに一時停止している場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。

苦情があったアドレスへの送信は直ちに停止する必要があると同時に、受信側で苦情が発生する要因を特定することも重要です。これらの要因を特定した後、対処できるようにメール送信の動作を調整します。

お客様のアカウントがレビュー対象になっている場合

レビュー期間の終了時にアカウントの苦情率が 0.5% を超えている場合、問題が解決されるまで、お客様のアカウントの E メール送信機能を一時停止することがあります。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。ケースへのレスポンスで、実装した変更について説明します。この変更によって苦情率が低下すると見なされた場合は、変更の実施後に受信した苦情のみが考慮されるように計算が調整されます。

アカウントの E メール送信機能が一時停止された場合

この決定の再検討をリクエストできます。詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メールには、この問題の解決のために実行したアクションの詳細と、この問題を再度発生させないための計画の詳細を記入してください。リクエストを受け取った後、ご提供いただいた情報を確認し、必要に応じてアカウントのステータスを変更します。

アンチスパム組織の通知

このセクションでは、Amazon SES の評価メトリクスページに表示されるアンチスパム組織通知についての追加情報を提供します。

この通知を受け取った理由

信頼できるアンチスパム組織により、お客様の Amazon SES アカウントから送信されているコンテンツの一部について、未承諾または問題ありというシステムによるフラグ付けがされたと報告されています。

アンチスパム組織がコンテンツに問題があると判断した原因となった特定のメッセージに関する情報を提供することはできません。また、そのレポートを発行した組織の名前を開示することもできません。一般的にアンチスパム組織では、受信者のフィードバック、メッセージのエンゲージメントに関するメトリクス、無効なアドレスへの配信の試行数、スパムフィルタによってフラグ付けされたコンテンツ、スパムトラップのヒット数などの要因の組み合わせが考慮されます。これはすべてを網羅したリストではありません。これ以外の要因によって、お客様のコンテンツへのフラグ付けが行われることもあります。

問題を解決する方法

この問題を解決するには、E メール送信プログラムのどのような点が原因で、アンチスパム組織によって問題のあるメールとしてのフラグ付けが行われているのか調べる必要があります。そのうえで、問題に対処できるようにメール送信プログラムを変更する必要があります。

お客様のアカウントがレビュー対象になっている場合

レビュー期間の終了時に、アンチスパム組織が引き続きお客様のアカウントから送信された E メールを問題のあるものとして特定した場合は、問題が解決されるまで、お客様のアカウントの E メール送信機能を一時停止する可能性があります。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージには、加えた変更の詳細を入力します。この情報の到着後、レビュー期間が延長されます。これは、お客様による変更の実施後にアンチスパム組織から送信される通知のみを分析するためです。延長されたレビュー期間の終了時に、お客様のアカウントはアンチスパム組織のリストに含まれなくなり、お客様のアカウントのレビュー期間は解除されます。

アカウントの E メール送信機能が一時停止された場合

この決定の再検討をリクエストできます。詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メールには、この問題の解決のために実行したアクションの詳細と、この問題を再度発生させないための計画の詳細を記入してください。リクエストを受け取った後、ご提供いただいた情報を確認し、必要に応じてアカウントのステータスを変更します。

リスト型攻撃通知

このセクションでは、Amazon SES の評価メトリクスページに表示されるリスト型攻撃通知についての追加情報を提供します。

この通知を受け取った理由

スパム対策組織は、メール送信プロセスが「リスト型攻撃」に対して脆弱であることを特定しました。リスト型攻撃は、攻撃者が非常に多数の E メールアドレスをウェブベースのフォームに登録する不正行為の一形態です。リスト型攻撃は、影響を受ける E メールサービスのユーザーへのサービ

ス提供が中断される可能性があります。また、E メールプロバイダーによって E メールがブロックされることもあります。

スパム対策組織は、独自の方法を使用して、リスト型攻撃に脆弱なサイトを特定します。このため、スパム対策組織が E メール送信プロセスを問題と見なした理由については、追加の詳細を提供することはできません。また、問題を特定した組織の名前を共有することはできません。

問題を解決する方法

ウェブベースのサインアップフォームをすべて調べ、この種の不正行為に対して脆弱でないことを確認する必要があります。自動スクリプトによってサブスクリプションリクエストが送信されるのを防ぐため、すべてのフォームに CAPTCHA を組み込む必要があります。さらに、新しいユーザーが製品やサービスにサインアップしたら、実際にサインアップしようとしているかを確認する E メールを送信します。お客様がコミュニケーションに明示的にオプトインしない場合は、そのお客様には追加の E メールを送信しないでください。

最後に、E メールリストでアクセス許可手順を実行する必要があります。アクセス許可手順では、E メールを引き続き受け取りたいかどうかをすべてのお客様に確認します。E メールを引き続き受け取ることを希望しているお客様にのみ、E メールを送信してください。

お客様のアカウントがレビュー対象になっている場合

レビュー期間の終了時に、アンチスパム組織が引き続きお客様のアカウントから送信された E メールを問題のあるものとして特定した場合は、問題が解決されるまで、お客様のアカウントの E メール送信機能を一時停止する可能性があります。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージには、加えた変更の詳細を入力します。この情報の到着後、レビュー期間が延長されます。これは、お客様による変更の実施後にアンチスパム組織から送信される通知のみを分析するためです。延長されたレビュー期間の終了時に、お客様のアカウントはアンチスパム組織のリストに含まれなくなり、お客様のアカウントのレビュー期間は解除されます。

アカウントの E メール送信機能が一時停止された場合

この決定の再検討をリクエストできます。詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メールには、この問題の解決のために実

行したアクションの詳細と、この問題を再度発生させないための計画の詳細を記入してください。リクエストを受け取った後、ご提供いただいた情報を確認し、必要に応じてアカウントのステータスを変更します。

直接フィードバック通知

このセクションでは、Amazon SES の評価メトリクスページに表示される直接フィードバック通知についての追加情報を提供します。

この通知を受け取った理由

お客様の Amazon SES アカウントに関連するアドレスまたはドメインから受信したメッセージについてのレポートが、相当数のユーザーから Amazon SES に直接寄せられました。このタイプのフィードバックは、メールボックスプロバイダーから直接報告される苦情には反映されておらず、評価メトリクスページに表示されるバウンスと苦情のメトリクスにも含まれていません。

これらの問題を報告したユーザーのプライバシーを保護するために、そのユーザーの E メールアドレスを開示することはできません。

受信者から Amazon SES への苦情は、受信者が受信登録していないメッセージを受信した場合や、受信する予定のタイプのメールを受信しなかった場合、受信したメールが役立つメールまたは興味を引くメールではなかった場合、受信したメッセージが受信登録したものではないと思われる場合、受信するメッセージが多すぎる場合などに生じます。これはすべてを網羅したリストではありません。お客様のケースに該当する要因は、お使いの E メール送信プログラムによって異なります。

問題を解決する方法

新しいアドレスを取得するには「リストの構築とメンテナンス」に記載されているダブルオプトイン戦略を実装し、ダブルオプトインプロセスを完了したアドレスのみにメールを送信することをお勧めします。

さらに、送信したメールに対してしばらく応答がなかった一連のアドレスは、消去する必要があります。送信したコンテンツに対して閲覧および応答を行っているユーザーを特定するには、「Amazon SES 送信アクティビティのモニタリング」に記載されているオープンとクリックの追跡を使用できます。

お客様のアカウントがレビュー対象になっている場合

レビュー期間の終了時に、Amazon SES が引き続きお客様のアカウントから送信されたメッセージに関する直接的な苦情を多数受け取る場合、問題が解決されるまで、お客様のアカウントの E メール送信機能を一時停止する可能性があります。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。問題を解決するために行った手順の詳細情報を提供し、これらの手順を実行して今後問題が再発するのを防ぐ方法を説明してください。お客様が加えた変更により問題が適切に対処されたと見なされた場合は、アカウントのレビュー期間をキャンセルします。

アカウントの E メール送信機能が一時停止された場合

この決定の再検討をリクエストできます。詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メールには、この問題の解決のために実行したアクションの詳細と、この問題を再度発生させないための計画の詳細を記入してください。リクエストを受け取った後、ご提供いただいた情報を確認し、必要に応じてアカウントのステータスを変更します。

ドメインブロックリスト通知

このセクションでは、Amazon SES の評価メトリクスページに表示されるドメインブロックリスト通知についての追加情報を提供します。

この通知を受け取った理由

信頼できるメインブロックリストに記載のあるドメインへの参照が、お客様の Amazon SES アカウントから送信されたメールに含まれています。これらのリストに記載のあるドメインは一般的に、不正または悪意のある動作と関連しています。そのドメインがメールの送信元ドメインかどうかは問題ではありません。ブロックリストに記載のあるドメインへの参照またはリンクが含まれるメッセージのほか、このようなドメインでホストされている画像が含まれたメッセージにもフラグが付けられる可能性があります。

お客様のメッセージにフラグが付く原因となったドメイン名を提供することも、どのメッセージにフラグが付いたかを特定することもできません。

問題を解決する方法

まず、Amazon SES 経由で送信する E メールで参照されているすべてのドメインのリストを作成します。次に、Spamhaus ドメインルックアップツールを使用して、E メールのどのドメインがドメインブロックリストに含まれているかを判断します。送信した E メール内で参照されている複数のドメインが、このブロックリストに含まれていることもあります。

Spamhaus ドメインブロックリストは、Amazon SES や AWS とは関係がありません。当社では、このリストのドメインの正確性を保証していません。Spamhaus Domain Blocklist および Domain Lookup Tool は、Spamhaus Project によって所有、運用、保守されています。

お客様のアカウントがレビュー対象になっている場合

レビュー期間中に送信する E メールで、悪意のある目的で使用されたドメインへの参照がないか確認されます。これらのドメインへの参照が E メールに引き続き多数含まれている場合、問題が解決されるまで、アカウントの E メール送信機能が一時停止されることがあります。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージには、加えた変更の詳細を入力します。この情報の到着後、レビュー期間が延長されます。これは、お客様による変更の実施後にお客様の E メールに含まれるブロックリスト記載ドメイン数のみを分析するためです。この延長されたレビュー期間の終了時に、ドメインブロックリスト通知の数が減ったり削除されたりした場合、およびお客様がこの問題の今後の再発を防ぐための対策を講じたと考えられる場合、アカウントのレビュー期間はキャンセルされます。

アカウントの E メール送信機能が一時停止された場合

この決定の再検討をリクエストできます。詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メールには、この問題の解決のために実行したアクションの詳細と、この問題を再度発生させないための計画の詳細を記入してください。リクエストを受け取った後、ご提供いただいた情報を確認し、必要に応じてアカウントのステータスを変更します。

内部レビュー通知

このセクションでは、Amazon SES の評価メトリクスページに表示される内部レビュー通知についての追加情報を提供します。

この通知を受け取った理由

お客様のアカウントに関する総合的なレビューにより、メールボックスプロバイダーまたは受信者によってお客様のメッセージがスパムであると見なされる原因となった特性がいくつか特定されました。

不正検出プロセスを保護するために、お客様のアカウントがこのようにフラグ付けされる原因となった特定の要因を開示することはできません。

この判断につながる可能性のある一般的な要因としては、次のようなものがあります。

- 商用のアンチスパムシステムによってフラグ付けされるメッセージ。
- 受取人がメールを明示的にリクエストしていないことを示すメッセージコンテンツ。
- メッセージの送信者とメール本文内に記載のあるブランドとの不一致。
- 送信者を明らかにしていないコンテンツ。
- 未承諾メールに関連したコンテンツを扱うメッセージの送信。
- 未承諾メールに関連する書式パターン。
- 評価の低いドメインからの送信または評価の低いドメインの参照。

これは包括的なリストではありません。この通知が発生した理由は、これらの要因の組み合わせである場合も、ここに記載されていない場合もあります。

問題を解決する方法

問題の深刻度の軽減には、以下の推奨事項が役立つ可能性があります。

- お客様からの E メールを明示的に要求した人のみがメールの受信者になっていることを確認します。
- メールの受取人のリストは、絶対に購入、賃借、借用しないでください。
- 送信するメッセージでは、送信者名やメールの目的を隠そうとしないでください。
- Amazon SES を介して送信する E メール内で参照されているすべてのドメインのリストを作成し、Spamhaus Domain Lookup ツール (https://www.spamhaus.org/lookup/) を使用して、これらのドメインが Spamhaus Domain Blocklist に記載されていないか調べます。
- メールを設計するときは、業界のベストプラクティスに従っていることを確認します。

これはすべてを網羅したリストではありませんが、お客様の E メールにフラグが付けられる原因となった一般的な要因の一部を特定するために役立つ可能性があります。

Spamhaus ドメインブロックリストは、Amazon SES や AWS とは関係がありません。当社では、このリストのドメインの正確性を保証していません。Spamhaus Domain Blocklist および Domain Lookup Tool は、Spamhaus Project によって所有、運用、保守されています。

アカウントがレビュー対象になっている場合、またはアカウントの E メール送信機能が一時停止している場合

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。問題を解決するために行った手順の詳細情報を提供し、これらの手順を実行して今後問題が再発するのを防ぐ方法を説明してください。お客様が加えた変更により問題が適切に対処されたと見なされた場合は、アカウントのレビュー期間をキャンセル、または一時停止を解除します。

お客様のアカウントからレビュー期間または送信一時停止を解除したものの、後で同じ問題が発生した場合、アカウントを再度レビュー対象としたり、お客様の E メール送信機能を一時停止する可能性があります。極端なケースや、同じ問題が繰り返し発生した場合は、お客様のアカウントの E メール送信機能を完全に停止する可能性があります。

アカウントがレビュー対象になった場合、またはアカウントの E メール送信機能が一時停止された場合の対処方法については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

メールボックスプロバイダー通知

このセクションでは、Amazon SES の評価メトリクスページに表示されるメールボックスプロバイダー通知についての追加情報を提供します。

この通知を受け取った理由

お客様の Amazon SES アカウントに関連するアドレスまたはドメインから、未承諾メールまたは悪意のあるメールが送信されたと、主要なメールボックスプロバイダーから報告されました。

このレポートを発行した組織の名前を開示することはできません。また、メールボックスプロバイダーからレポートが発行される原因となった特定の要因に関する情報も提供できません。一般的に、メールボックスプロバイダーでは、顧客フィードバック、顧客エンゲージメントに関するメトリクス、無効アドレスに対する配信の試行数、スパムフィルタによってフラグ付けされたコンテンツなどの情報に基づいて、このような判断が行われます。これはすべてを網羅したリストではありません。メールボックスプロバイダーでお客様のコンテンツにフラグが付けられる原因となった要因は、他にも存在する可能性があります。

問題を解決する方法

この問題を解決するには、E メール送信プログラムのどのような点が原因で、メールボックスプロバイダーでお客様のメールに問題があるとしてフラグが付けられたのか調べる必要があります。そのうえで、問題に対処できるようにメール送信プログラムを変更する必要があります。

お客様のアカウントがレビュー対象になっている場合

レビュー期間の終了時に、メールボックスプロバイダーが引き続きお客様のアカウントから送信された E メールを問題があるとして特定する場合は、問題が解決されるまで、お客様のアカウントの E メール送信機能を一時停止する可能性があります。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージには、加えた変更の詳細を入力します。この情報の到着後、レビュー期間が延長されます。これは、お客様による変更の実施後に受信したメールボックスプロバイダー通知の数のみを分析するためです。延長されたレビュー期間の終了時に、お客様のコンテンツに問題があるとメールボックスプロバイダーから報告されなければ、お客様のアカウントからレビューが解除される場合があります。

アカウントの E メール送信機能が一時停止された場合

この決定の再検討をリクエストできます。詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メールには、この問題の解決のために実行したアクションの詳細と、この問題を再度発生させないための計画の詳細を記入してください。リクエストを受け取った後、ご提供いただいた情報を確認し、必要に応じてアカウントのステータスを変更します。

受信者フィードバック通知

このセクションでは、Amazon SES の評価メトリクスページに表示される受信者フィードバック通知についての追加情報を提供します。

この通知を受け取った理由

主要なメールボックスプロバイダーから受けた報告により、そのプロバイダーの多くのユーザーによって、お客様の Amazon SES アカウントから送信されたメールが未承諾であるとされていることが判明しました。このタイプのフィードバックは、メールボックスプロバイダーから直接報告される苦情には反映されておらず、Amazon SES のバウンス通知および苦情通知にも含まれていません。

多数の苦情が発生すると、すべての Amazon SES ユーザーに悪影響が及ぶ可能性があります。アカウントに対して一定数の苦情が寄せられた場合は、お客様およびその他の Amazon SES ユーザーの評価を保護するために、迅速なアクションが実行されます。

お客様のメールを未承諾であるとしている特定のメールアドレスのリストは、提供できません。さらに、この問題に関する報告元であるメールボックスプロバイダー名も開示できません。

問題を解決する方法

この問題を解決するには、E メール送信プログラムのどのような点が原因で、受取人がお客様から受け取った E メールメッセージに関する苦情が生じたのか調べる必要があります。これらの要因を特定した後、対処できるようにメール送信の処理を変更します。

新しいアドレスを取得するには「リストの構築とメンテナンス」に記載されているダブルオプトイン戦略を実装することをお勧めします。また、ダブルオプトインプロセスを完了したアドレスのみにメールを送信することをお勧めします。

さらに、送信したメールに対してしばらく応答がなかった一連のアドレスは、消去する必要があります。送信したコンテンツに対して閲覧および応答を行っているユーザーを特定するには、「Amazon SES 送信アクティビティのモニタリング」に記載されているオープンとクリックの追跡を使用できます。

お客様のアカウントがレビュー対象になっている場合

レビュー期間の終了時に、メールボックスプロバイダーが直接的な苦情を引き続き多数報告する場合、問題が解決されるまで、お客様のアカウントの E メール送信機能を一時停止する可能性があります。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージには、加えた変更の詳細を入力します。この情報の到着後、レビュー期間が延長されます。これは、お客様による変更の実施後に受信したメールボックスプロバイダーからの苦情数のみを分析するためです。延長されたレビュー期間の終了時に、メールボックスプロバイダーからの苦情数が減少するかゼロになっていれば、お客様のアカウントからレビューが解除される場合があります。

アカウントの E メール送信機能が一時停止された場合

この決定の再検討をリクエストできます。詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メールには、この問題の解決のために実行したアクションの詳細と、この問題を再度発生させないための計画の詳細を記入してください。リクエストを受け取った後、ご提供いただいた情報を確認し、必要に応じてアカウントのステータスを変更します。

関連アカウント通知

このセクションでは、Amazon SES の評価メトリクスページに表示される関連アカウント通知についての追加情報を提供します。

この通知を受け取った理由

別の Amazon SES アカウントから送信されたメールに関連する重要な問題が検出されました。問題のあるアカウントはお客様の AWS アカウント と関連していると思われ、同様の問題を回避するためにアクションが実行されました。

問題を解決する方法

あるアカウントの E メール送信機能を一時停止する場合、送信一時停止の理由に関する情報が常にそのアカウントの所有者に送信されます。詳細については、関連するアカウントの所有者に送信した E メールを参照してください。

まず、この関連アカウントの問題に対処してください。問題解決のための変更を実施した後は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。問題を解決するために行った手順の詳細情報を提供し、これらの手順を実行して今後問題が再発するのを防ぐ方法を説明してください。お客様が加えた変更により問題が適切に対処されたと見なされた場合は、アカウントのレビュー期間をキャンセル、または一時停止を解除します。

スパムトラップ通知

このセクションでは、Amazon SES の評価メトリクスページに表示されるスパムトラップ通知についての追加情報を提供します。

この通知を受け取った理由

サードパーティーのアンチスパム組織から、その組織のスパムトラップアドレスで、お客様の Amazon SES アカウントに関連付けられている確認済みアドレスまたはドメインからメールを受信したという報告を受けました。

スパムトラップは、未承諾メール (スパム) だけを引き付けるために使用される休眠メールアドレスです。多数のスパムトラップレポートが発生すると、すべての Amazon SES ユーザーに悪影響が及ぶ可能性があります。アカウントから一定量のメールがスパムトラップアドレスに送信された場合は、お客様およびその他の Amazon SES ユーザーの評価を保護するために、迅速なアクションが実行されます。

問題を解決する方法

使用されたスパムトラップに関連する E メールアドレスを開示することはできません。これらのアドレスは、公開されると意味がなくなるため、所有元の組織で厳重に保護されています。

スパムトラップアドレスにメールが送信される場合は、一般的に、顧客のメールアドレス入手方法に問題があることを示します。たとえば、購入したメールアドレスリストには、スパムトラップアドレスが含まれている可能性があります。Amazon SES のサービスの利用条件で、購入または借用したリストへの送信が禁止されているのは、このためです。新しいアドレスを取得するには「リストの構築とメンテナンス」に記載されているダブルオプトイン戦略を実装することをお勧めします。また、ダブルオプトインプロセスを完了したアドレスのみにメールを送信することをお勧めします。

さらに、送信したメールに対してしばらく応答がなかった一連のアドレスは、消去する必要があります。送信したコンテンツに対して閲覧および応答を行っているユーザーを特定するには、「Amazon SES 送信アクティビティのモニタリング」に記載されているオープンとクリックの追跡を使用できます。

お客様のアカウントがレビュー対象になっている場合

レビュー期間の終了時に、お客様のアカウントから引き続きスパムトラップアドレスにメッセージが送信されている場合は、問題が解決されるまで、お客様のアカウントの E メール送信機能を一時停止する可能性があります。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージには、加えた変更の詳細を入力します。この情報の到着後、レビュー期間が延長されます。これは、お客様による変更の実施後に受信したスパムトラップレポートの数のみを分析するためです。延長されたレビュー期間の終了時に、スパムトラップレポートの数が減少するかゼロになっていれば、お客様のアカウントからレビューが解除される場合があります。

アカウントの E メール送信機能が一時停止された場合

この決定の再検討をリクエストできます。詳細については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メールには、この問題の解決のために実行したアクションの詳細と、この問題を再度発生させないための計画の詳細を記入してください。リクエストを受け取った後、ご提供いただいた情報を確認し、必要に応じてアカウントのステータスを変更します。

脆弱サイト通知

このセクションでは、Amazon SES の評価メトリクスページに表示される脆弱性があるサイト通知についての追加情報を提供します。

この通知を受け取った理由

お客様のアカウントに関する総合的なレビューにより、お客様が意図していないと思われるメッセージがお客様のアカウントから送信されたことがわかりました。これらのメッセージは、高い確率でメールボックスプロバイダーおよび受取人からスパムとしてフラグ付けされます。

このような場合は、お客様のウェブサイトの機能を悪用して第三者が迷惑メールを送信していることが考えられます。たとえば、お客様のウェブサイトに、"友だちにメールを送る"、"お問い合わせ"、"友だちを招待する" などの機能がある場合、第三者はその機能を使用して未承諾メールを送信できます。

問題を解決する方法

まず、お客様が知らない間に Amazon SES を使用して第三者が E メールを送信する可能性のある、お客様のウェブサイトまたはアプリケーションの機能を特定します。サポートセンターのケースでは、この方法で送信されたメッセージのサンプルをリクエストできます。

次に、未承諾送信を回避できるように、アプリケーションまたはウェブサイトを修正します。たとえば、CAPTCHA を追加する、メールを送信できる速度を制限する、ユーザーによるカスタムコンテンツの送信を禁止する、メール送信前にユーザーにログインを求める、アプリケーションによる複数の同時通知の生成を禁止するなどの方法があります。

アカウントがレビュー対象になっている場合、またはアカウントの E メール送信機能が一時停止している場合

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メールには、この問題の解決のために実行したアクションの詳細と、この問題を再度発生させないための計画の詳細を記入してください。リ

クエストを受け取った後、ご提供いただいた情報を確認し、必要に応じてアカウントのステータスを変更します。

お客様のアカウントからレビュー期間または送信一時停止を解除したものの、後で同じ問題が発生した場合、アカウントを再度レビュー対象としたり、お客様の E メール送信機能を一時停止する可能性があります。極端な問題や、同じ問題が繰り返し確認された場合は、お客様のアカウントの E メール送信機能を完全に停止する可能性があります。

アカウントがレビュー対象になった場合、またはアカウントの E メール送信機能が一時停止された場合の対処方法については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

認証情報の漏洩に関する通知

このセクションでは、Amazon SES の評価メトリクスページに表示される認証情報の漏洩に関する通知についての追加情報を提供します。

この通知を受け取った理由

お客様のアカウントに関する総合的なレビューにより、お客様が意図していないと思われるメッセージがお客様のアカウントから送信されたことがわかりました。これらのメッセージは、高い確率でメールボックスプロバイダーおよび受取人からスパムとしてフラグ付けされます。

一般的な原因は、IAM アクセスキーの漏洩、SMTP パスワードの漏洩、またはその他のセキュリティの脆弱性です。

問題を解決する方法

SES 利用メカニズムの包括的なセキュリティレビューを実施する必要があります。適用可能なパスワードまたは SMTP パスワードをローテーションし、無許可のユーザーまたはリソースをアカウントから削除したことを確認してください。パスワードやアクセスキーなどの機密情報をサードパーティーのウェブサイトやリポジトリに保存していないことを確認します。現在、ユーザーには IAM アクセスキーを使用せず、ルートユーザーには絶対に使用しないことが推奨されています。まだ使用している場合は、AWS IAM Identity Center でユーザーを作成するなど、一時的な認証情報を提供するメカニズムに移行する必要があります。

アカウントがレビュー対象になっている場合、またはアカウントの E メール送信機能が一時停止している場合

問題解決のための変更を実施した場合は、AWS コンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メールには、この問題の解決のために実

行したアクションの詳細と、この問題を再度発生させないための計画の詳細を記入してください。リクエストを受け取った後、ご提供いただいた情報を確認し、必要に応じてアカウントのステータスを変更します。

お客様のアカウントからレビュー期間または送信一時停止を解除したものの、後で同じ問題が発生した場合、アカウントを再度レビュー対象としたり、お客様の E メール送信機能を一時停止する可能性があります。極端な問題や、同じ問題が繰り返し確認された場合は、お客様のアカウントの E メール送信機能を完全に停止する可能性があります。

アカウントがレビュー対象になった場合、またはアカウントの E メール送信機能が一時停止された場合の対処方法については、「Amazon SES 送信レビュープロセスに関するよくある質問」を参照してください。

その他の通知

このセクションでは、Amazon SES の評価メトリクスページに表示されるその他の通知についての追加情報を提供します。

この通知を受け取った理由

自動レビューまたは人によるレビューで、このドキュメントのこれまでのセクションに記載されていない問題が見つかりました。

問題を解決する方法

特定の問題の詳細については、当社がお客様に代わってオープンしたサポートセンターのケースを参照してください。サポートセンターにアクセスするには、AWS Management Console にサインインし、サポートセンターを選択します。ケースへのレスポンスで、実装した変更について説明します。検出された問題の状況と内容に応じて、レビュー期間が終了したり、お客様のアカウントの E メール送信機能が復旧する可能性があります。

CloudWatch を使用して評価モニタリングアラームを作成する

Amazon SES は、一連の評価関連のメトリクスを自動的に Amazon CloudWatch に発行します。これらのメトリクスを使用して、バウンス率または苦情率がアカウントのメール送信機能に影響するレベルに達すると通知するアラームを作成できます。

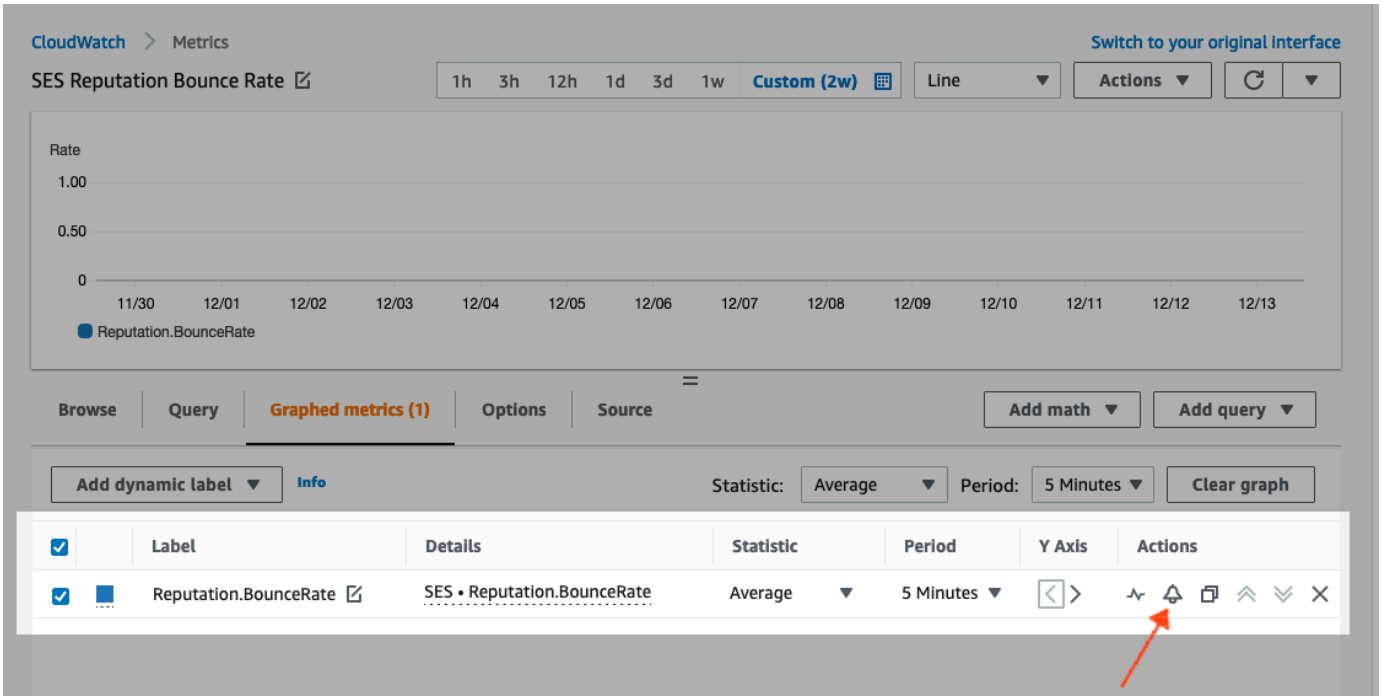
> **Note**
>
> このセクションの手順の CloudWatch の部分は、SES 送信者評価をモニタリングするための CloudWatch アラームの設定について主な手順を説明することを目的としています。CloudWatch アラームのオプション設定に関する詳細設定については説明しません。CloudWatch アラームの設定の詳細情報については、Amazon CloudWatch ユーザーガイドの「Amazon CloudWatch アラームの使用」を参照してください。

前提条件

- Amazon SNS トピックを作成し、任意のエンドポイント (メールや SMS など) を使用してそのトピックをサブスクライブします。詳細については、Amazon Simple Notification Service デベロッパーガイドの「Amazon SNS トピックを作成する」および「Amazon SNS トピックへサブスクライブする」を参照してください。
- 現在のリージョンで E メールを送信したことがない場合は、SES 名前空間が無いことがあります。メトリクスを確実に取得するには、メールボックスシミュレーター宛にテストメールを送信します。

送信評価をモニタリングするための CloudWatch アラームを作成するには

1. AWS Management Console にサインインして Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. 画面左側のナビゲーションペインで、[Reputation metrics]を選択します。
3. [評価メトリクス] ページの [アカウントレベル] タブの [バウンス率] または [苦情率] ペインで、[CloudWatch で表示する] を選択します。これにより、CloudWatch コンソールが開いて、選択したメトリクスが表示されます。
4. [Graphed metrics] (グラフ化されたメトリクス) タブでは、選択したメトリックスの行で (この例では、[Reputation.BounceRate] (評価バウンス率))、[Actions] (アクション) 列 (下の画像を参照) の [alarm bell] (アラームベル) アイコンを選択します - これにより、[Specify metric and conditions] (メトリクスと条件の指定) ページが開きます。



5. [Conditions] (条件) ペインまで下にスクロールして、[Threshold type] (しきい値タイプ) フィールドの [Static] (静的) を選択します。

   a. [Whenever ***metric*** is...] (メトリクスが...の場合) フィールドで、[Greater/Equal] (大きい/等しい) を選択します。

   b. [than...] (...より) フィールドで、CloudWatch でアラームを発生させる値を指定します。

      - アラームを作成してバウンス率をモニタリングする場合、Amazon SES ではバウンス率を 5% よりも低く維持することをお勧めします。アカウントのバウンス率が 10% を超える場合、当社はお客様のアカウントによる E メール送信機能を一時停止することがあります。このため、アカウントのバウンス率が 0.05 (5%) 以上になると通知が送信されるように CloudWatch を設定する必要があります。
      - アラームを作成して苦情率をモニタリングする場合、Amazon SES では苦情率を 0.1% よりも低く維持することをお勧めします。アカウントの苦情率が 0.5% を超える場合、当社はお客様のアカウントによる E メール送信機能を一時停止することがあります。このため、アカウントの苦情率が 0.001 (0.1%) 以上になると通知が送信されるように CloudWatch を設定する必要があります。

   c. [Additional configuration] (追加設定) を展開し、[Missing data treatment]] (欠落データの処理) フィールドで、[Treat missing data as ignore (maintain the alarm state)] (欠落データを無視として処理する (アラーム状態を維持する) を選択します。

   d. [Next] を選択します。

6. [Configure actions] (アクションの設定) ペインの [Alarm state trigger] (アラーム状態トリガー) で [In Alarm] (アラーム状態) を選択します。

   a. [Select an SNS topic] (SNS トピックの選択) フィールドで、[Select an existing SNS topic] (既存の SNS トピックの選択) を選択します。

   b. [Send notification to...] (...に通知を送信する) 検索ボックスで、前提条件であらかじめ作成してサブスクライブしておいたトピックを選択します。

   c. [Next] を選択します。

7. [Add name and description] (名前と説明の追加) ペインで、アラームの名前と説明を入力し、[Next] (次へ) を選択します。

8. [Preview and create] (プレビューと作成) ペインで、設定を確認し、問題がなければ [Create alarm] (アラームの作成) を選択します。変更したい箇所がある場合は、戻って編集するセクションの [Previous] (戻る) ボタンを選択します。

SNDS 専用 の メトリクス IPs

Amazon AWS リージョン を使用する各 で、リース専用 IP アドレスのスマートネットワークデータサービス (SNDS) データを表示できますSES。このSNDSデータは Amazon CloudWatch コンソールから入手できます。

SNDS は、IP 所有者が IP スペース内のスパムを防止できるようにする Outlook プログラムです。Amazon SESは、専用の をリースするユーザー向けに、この重要なデータを提供しますIPs。このSNDSデータは、IP のメール送信動作に関するインサイトを提供し、送信者の評価に懸念のある領域を呼び出します。

> **Note**
>
> - SNDS は、専用 IP アドレス (マネージド) と互換性がありません。
> - Outlookを参照する際、これはトラッキングするすべてのドメインをカバーします。例えば、これは Hotmail.com、Outlook.com、および Live.com をカバーできます。

専用 IP アドレスSNDSのデータを表示するには

1. で Amazon CloudWatch コンソールにサインインしますhttps://console.aws.amazon.com/cloudwatch/。

2. ナビゲーションペインで、[Metrics] (メトリクス) を展開し、[All metrics] (すべてのメトリクス) を選択します。

   (新しい CloudWatch コンソールインターフェイスに指示が与えられます)。

3. メトリクスコンテナの参照タブで、 を選択し AWS リージョン、 を選択しますSES。

4. によってIPs追跡されるすべての専用 を表示する IP メトリクスを選択しますSNDS。

   (注: 選択したリージョンにアカウントに関連付けられた専用 IP アドレスがない場合、IP メトリクスは CloudWatch コンソールに表示されません)。

5. このリストSNDSで によってIPs追跡されるすべての専用 を表示するか、個々の IP アドレスを選択してメトリクスのみを表示します。

次のメトリクスは、専用 IP アドレスごとに提供され、Outlook によって定義されます。詳細については、「Outlook の SNDS 」を参照してくださいFAQs。

> **Note**
>
> これらのメトリクスは、1 日に 1 回更新データを提供する活動期間を表します。メトリクスには、24 時間に対応するタイムスタンプもあります。

- SNDS。RCPTCommandsこれは、アクティビティ期間中に特定の IP アドレスSNDSに対して が認識するRCPTコマンドの数です。 RCPT コマンドは、E メールの送信に使用されるSMTPプロトコルの一部であり、E メールの配信先となる受信者アドレスを指定します。
- SNDS.DATACommands - アクティビティ期間中に特定の IP アドレスSNDSに対して によって認識されるDATAコマンドの数。 DATA コマンドは、メールの送信に使用されるSMTPプロトコルの一部です。特に、以前に確立された目的の受信者に実際にメッセージを送信する部分です (複数可)。
- SNDS.MessageRecipients - アクティビティ期間中に特定の IP アドレスSNDSについて によって認識されたメッセージに対する受信者の数。
- SNDS.SpamRate - 指定されたアクティビティ期間中に IP アドレスによって送信されたすべてのメッセージに適用されたスパムフィルタリングの集計結果を表示します。
  - SpamRate が 0 の場合、IP アドレスのスパムが 10% 未満であることを意味します。
  - SpamRate が 0.5 の場合、10% から 90% のスパムが IP アドレスから生成されることを意味します。

  - SpamRate が 1 の場合、90% 以上のスパムが IP アドレスから生成されることを意味します。
- SNDS.ComplaintRate - これは、アクティビティ期間中に Outlook ユーザーが IP から受信したメッセージについて苦情を申し立てた時間の割合です。
  - ComplaintRate が 1 の場合、苦情率は 100% になります。
  - たとえば、 ComplaintRate が 0.05 の場合、苦情率は 5% になります。
  - ComplaintRate が 0 の場合は、レートが 0.1% 未満であることを意味します。
- SNDS.TrapHits - 「アカウントのトラップ」に送信されたメッセージの数を表示します。トラップアカウントは、Outlook によって管理され、メールを要求しないアカウントです。したがって、トラップアカウントに送信されたメッセージは、スパムである可能性が非常に高いです。

トラブルシューティングに関する質問

Q1. データが毎日移入されないのはなぜですか ？ 次のシナリオのいずれかが適用できます。

- SNDS データは Outlook のSNDSプログラムによって異なります。
- 値を計算するために受信SNDSする必要がある E メールのしきい値は最小限です。IP上の E メールのボリュームが少ない場合、データは利用できない場合があります。

Q2. SNDS.ComplaintRate metrics が変更されるのはなぜですかSpamRate SNDS。また、レートが 1 の値に変わった場合はどうすればよいですか？

これは、送信動作で何かが Outlook SNDS プログラムからの否定的な応答をトリガーしたことを示す指標です。この場合、他のインターネットサービスプロバイダー (ISPs) とエンゲージメント番号をチェックして、それがグローバルな問題ではないことを確認します。グローバル問題の場合、リスト、コンテンツISPs、ディストリビューション、またはアクセス許可の問題を示唆する複数の に問題がある可能性があります。Outlook に固有のものである場合は、Outlook に最適に配信する方法を参照ください。

Q3. SNDS. が 0 (または 0.5) から 1 SpamRate の値に変更された場合、 は AWS サポート どのようなアクションを実行しますか？

AWS には に対する制御がないためSNDS、 に対する影響はありませんSNDS。すべての緩和リクエストは新しいサポートリクエストフォームを通じて Outlook に直接ファイルする必要があります。

E メール送信の自動的な一時停止

送信者の評価の低下を防ぐために、特定の設定セットを使用して送信されているメッセージ、または特定のAWS地域の Amazon SES アカウントから送信されているメッセージすべてについて、E メールの送信を一時的に停止できます。

Amazon CloudWatch と Lambda を使用して、自動化することで、評価メトリクス (バウンス率や苦情率など) が一定の特定のしきい値を超えた場合に、E メール送信を自動的に停止させるソリューションを作成できます。このトピックでは、このソリューションをセットアップする手順について説明します。

このセクションのトピック:

Amazon SES アカウントの E メール送信を自動的に一時停止する

このセクションの手順では、単一のAWS地域で Amazon SES アカウントの E メール送信を自動的に一時停止するように Amazon SES、Amazon SNS、Amazon CloudWatch、 AWS Lambdaを設定するためのステップを説明します。複数のリージョンから E メールを送信する場合は、このソリューションを実装するリージョンごとにこのセクションの手順を繰り返します。

このセクションのトピック:

パート 1: IAM ロールの作成

E メール送信の自動一時停止を設定するための最初のステップは、`UpdateAccountSendingEnabled` API オペレーションを実行できる IAM ロール を作成することです。

IAM ロールを作成するには

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで Roles (ロール) を選択します。
3. [Create role] を選択します。
4. [信頼されたエンティティを選択] ページで、[信頼されたエンティティタイプ] の [AWS のサービス] を選択します。
5. [Use case] (ユースケース) で、Lambda を選択し、[Next] (次へ) を選択します。
6. [Add permissions] (アクセス許可を追加) ページで、次のポリシーを選択します。

   - AWSLambdaBasicExecutionRole
   - AmazonSESFullAccess

   > **Tip**
   >
   > [Permission policies] (アクセス許可ポリシー) の下にある検索ボックスを使用して、これらのポリシーをすばやく特定します。ただし、最初のポリシーを検索して選択した後、2 番目のポリシーを検索して選択する前に、[Clear filters] (フィルターをクリア) を選択する必要があります。

   次いで、[次へ] を選択します。
7. [Name, review, and create] (名前、確認、および作成) ページの [Role details] (ロールの詳細) の下にある [Role name] (ロール名) フィールドに、意味のあるポリシーの名前を入力します。
8. 選択した 2 つのポリシーが、[Permissions policy summary] (アクセス許可ポリシーの概要) テーブルに表示されていることを確認して、[Create role] (ロールの作成) を選択します。

パート 2: Lambda 関数を作成する

IAM ロールを作成した後で、Lambda 関数を作成して、アカウントの E メール送信を一時停止することができます。

Lambda 関数を作成するには

1. AWS Lambda コンソール (https://console.aws.amazon.com/lambda/) を開きます。

2. リージョンセレクターを使用して、この Lambda 関数をデプロイするリージョンを選択します。

> **Note**
>
> この関数では、このステップで選択した AWS リージョン内の E メール送信を一時停止します。複数のリージョンから E メールを送信する場合は、E メール送信を自動的に一時停止するリージョンごとにこのセクションの手順を繰り返します。

3. [Create function] を選択します。
4. [Create function] で [Author from scratch] を選択します。
5. [Basic information] (基本的な情報) ページで、以下のステップを完了します。

   - [Function name] (関数名) に Lambda 関数の名前を入力します。
   - [ランタイム] で、[Node.js 18x] (または、現在選択リストで提供されているバージョン) を選択します。
   - [Architecture] (アーキテクチャ) で、事前に選択されたデフォルト、x86_64 を保持します。
   - アクセス許可で、[Change default execution role] (デフォルト実行ロールの変更) を拡張し、[Use an existing role] (既存のロールを使用) を選択します。
   - [Existing role] (既存のロール) リストボックスで、the section called “パート 1: IAM ロールの作成” で作成した IAM ロールを選択します。

   次に、[Create function] を選択します。
6. コードエディタの [Code source] (コードソース) に、以下のコードを貼り付けます。

```
'use strict';


const { SES } = require("@aws-sdk/client-ses")

// Create a new SES object.

var ses = new SES({});

// Specify the parameters for this operation. In this case, there is only one
// parameter to pass: the Enabled parameter, with a value of false
// (Enabled = false disables email sending, Enabled = true enables it).
var params = {
```

```
    Enabled: false
};

exports.handler = (event, context, callback) => {
    // Pause sending for your entire SES account
    ses.updateAccountSendingEnabled(params, function(err, data) {
        if(err) {
            console.log(err.message);
        } else {
            console.log(data);
        }
    });
};
```

その後、[デプロイ] を選択します。

7. [テスト] を選択します。[Configure test event] (テストイベントの設定) ウィンドウが表示されたら、[Event name] (イベント名) フィールドに名前を入力し、[Save] (保存) を選択します。
8. [Test] (テスト) ドロップボックスを拡張し、作成したイベントの名前を選択してから、[Test] (テスト) を選択します。
9. そのすぐ下に、[Execution results] (実行結果) タブが表示されます。右に `Status: Succeeded` が表示されていることを確認します。関数の実行に失敗した場合は、次のことを実行します。

   - the section called “パート 1: IAM ロールの作成” で作成した IAM ロールに正しいポリシーが含まれていることを確認します。
   - Lambda 関数のコードにエラーがないことを確認します。Lambda コードエディタでは、構文エラーやその他の潜在的な問題が自動的にハイライトされます。

パート 3: アカウントの E メール送信の再有効化

the section called “パート 2: Lambda 関数を作成する” で Lambda 関数のテストをする副作用は、Amazon SES アカウントの E メール送信が一時停止していることです。ほとんどの場合は、CloudWatch アラームがトリガーされるまで、アカウントの送信を一時停止しません。

このセクションの手順では、Amazon SES アカウントの E メール送信を再度有効にします。これらの手順を完了するには、AWS Command Line Interface をインストールして設定する必要があります。詳細については、AWS Command Line Interface ユーザーガイドを参照してください。

E メール送信を再有効化するには

1. コマンドラインで次のコマンドを入力して、アカウントの E メールの送信を再度有効にします。*sending_region* を、E メール送信を再び有効にするリージョンの名前に置き換えます。

```
aws ses update-account-sending-enabled --enabled --region sending_region
```

2. コマンドラインで次のコマンドを入力して、アカウントの E メールの送信ステータスを確認します。

```
aws ses get-account-sending-enabled --region sending_region
```

次の出力が表示された場合は、アカウントの E メール送信が正常に再有効化されています。

```
{
    "Enabled": true
}
```

パート 4: Amazon SNS トピックおよびサブスクリプションを作成する

アラームが発生したときに CloudWatch が Lambda 関数を実行するには、まず Amazon SNS トピックを作成し、Lambda 関数をサブスクライブする必要があります。

Amazon SNS トピックを作成し、Lambda 関数をそのトピックにサブスクライブするには

1. Amazon SNS コンソール (https://console.aws.amazon.com/sns/v3/home) を開きます。
2. 「Amazon Simple Notification Service デベロッパーガイド」のステップに従ってトピックを作成します。
   - [Type] (タイプ) は、(FIFO ではなく) [Standard] (標準) にする必要があります。
3. Amazon Simple Notification Service デベロッパーガイドのステップに従ってトピックにサブスクライブします。
   a. [プロトコル] で AWS Lambda を選択します。
   b. [Endpoint] に、the section called “パート 2: Lambda 関数を作成する” で作成した Lambda 関数を選択します。

パート 5: CloudWatch アラームの作成

このセクションでは、メトリクスが特定のしきい値に到達するとトリガーされるアラームを CloudWatch に作成する手順を紹介します。アラームが発生すると、the section called “パート 4: Amazon SNS トピックおよびサブスクリプションを作成する” で作成した Amazon SNS トピックに通知が送信され、the section called “パート 2: Lambda 関数を作成する” で作成した Lambda 関数が実行されます。

CloudWatch アラームの作成

1. CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. リージョンセレクターを使用して、E メール送信を自動的に一時停止するリージョンを選択します。
3. ナビゲーションペインで、[Alarms] を選択します。
4. [Create Alarm (アラーム作成)] を選択します。
5. [Create Alarm] ウィンドウの、[SES Metrics] の下で、[Account Metrics] を選択します。
6. [Metric Name] で、次のいずれかのオプションを選択します。

   - Reputation.BounceRate – アカウントの全体のハードバウンス率が定義したしきい値を超えた場合に、アカウントの E メール送信を一時停止する場合は、このメトリクスを選択します。
   - Reputation.ComplaintRate – アカウントの全体の苦情率が定義したしきい値を超えた場合に、アカウントの E メール送信を一時停止する場合は、このメトリクスを選択します。

   [Next] を選択します。
7. 以下の手順を実行します。

   - [Alarm Threshold] の [Name] に、アラームの名前を入力します。
   - [Whenever: Reputation.BounceRate] または [Whenever: Reputation.ComplaintRate] で、アラームをトリガーするしきい値を指定します。

   > **Note**
   >
   > バウンス率が 5% を超えるか、苦情率が 0.1% を超えると、アカウントは自動的に確認中になります。CloudWatch アラームを発生させるバウンス率または苦情率を指定する場合は、これらの割合より低い値を使用して、お客様のアカウントが確認中とならないようにすることをお勧めします。

- [アクション] の [アラームが次の時:] で、[状態: 警告] を選択します。[Send notification to] に the section called “パート 4: Amazon SNS トピックおよびサブスクリプションを作成する” で作成した Amazon SNS トピックを選択します。

[Create Alarm] を選択します。

パート 6: ソリューションをテストする

ALARM 状態に入ったときに Lambda 関数を実行するようにアラームをテストできるようになりました。SetAlarmState API オペレーションを使用して、アラームの状態を一時的に変更することができます。

このセクションの手順はオプションですが、ソリューション全体が正しく設定されていることを確認するために、これらの手順を完了することをお勧めします。

1. コマンドラインで次のコマンドを入力して、アカウントの E メールの送信ステータスを確認します。*region* は、使用しているリージョンの名前に置き換えます。

```
aws ses get-account-sending-enabled --region region
```

送信がアカウントに対して有効になっている場合、次の出力が表示されます。

```
{
    "Enabled": true
}
```

2. コマンドラインで次のコマンドを入力して、アラーム状態を一時的に ALARM に変更します。aws cloudwatch set-alarm-state --alarm-name ***MyAlarm*** --state-value ALARM --state-reason "Testing execution of Lambda function" --region ***region***

前述のコマンドで、*MyAlarm* を、「the section called “パート 5: CloudWatch アラームの作成”」で作成したアラームの名前に置き換え、*region* を、E メール送信を自動的に一時停止するリージョンに置き換えます。

> **Note**
>
> このコマンドを実行すると、アラームの状態は OK から ALARM に切り替わり、数秒で OK に戻ります。これらのステータスの変更は、CloudWatch コンソールのアラームの [History] タブで DescribeAlarmHistory オペレーションを使用して表示できます。

3. コマンドラインで次のコマンドを入力して、アカウントの E メールの送信ステータスを確認します。

```
aws ses get-account-sending-enabled --region region
```

Lambda 関数が正常に実行された場合、次の出力が表示されます。

```
{
    "Enabled": false
}
```

4. the section called “パート 3: アカウントの E メール送信の再有効化” の手順を実行して、アカウントの E メール送信を再度有効にします。

設定セットの E メール送信を自動的に一時停止する

特定の設定セットを使用して Amazon SESに送信される E メールに固有の評価メトリクスをエクスポートするように Amazon を設定できます CloudWatch。その後、これらのメトリクスを使用して、これらの設定セットに固有の CloudWatch アラームを作成できます。これらのアラームが特定のしきい値を超えると、Amazon SESアカウントの E メール送信機能全体に影響を与えることなく、指定された設定セットを使用する E メールの送信を自動的に一時停止できます。

> **Note**
>
> このセクションで説明するソリューションは、1 つの AWS リージョン内の特定の設定セットの E メール送信を一時停止します。複数のリージョンから E メールを送信する場合は、このソリューションを実装するリージョンごとにこのセクションの手順を繰り返します。

このセクションのトピック:

- パート 1: 設定セットの評価メトリクスレポートを有効にする

- パート 2: IAM ロールの作成
- パート 3: Lambda 関数を作成する
- パート 4: 設定セットの E メール送信の再有効化
- パート 5: Amazon SNSトピックを作成する
- パート 6: CloudWatch アラームを作成する
- パート 7: ソリューションをテストする

パート 1: 設定セットの評価メトリクスレポートを有効にする

設定セットの E メール送信を自動的に一時停止SESするように Amazon を設定する前に、まず設定セットの評価メトリクスのエクスポートを有効にする必要があります。

設定セットのバウンスと苦情メトリクスのエクスポートを有効にするには、the section called “レピュテーションメトリクスの表示とエクスポートします” の手順を完了します。

パート 2: IAM ロールの作成

E メール送信の自動一時停止を設定する最初のステップは、`UpdateConfigurationSetSendingEnabled`APIオペレーションを実行できる IAMロールを作成することです。

IAM ロールを作成するには

1. 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 [Roles (ロール)] を選択します。
3. [Create role] を選択します。
4. [Select type of trusted entity] (信頼されたエンティティの種類を選択) で、[AWS service] (AWS のサービス) を選択します。
5. [このロールを使用するサービスを選択] の下で、[Lambda] を選択します。[Next: Permissions (次へ: アクセス許可)] を選択します。
6. [Attach permissions policies] ページで、次のいずれかのポリシーを選択します。

    - AWS Lambda BasicExecutionRole
    - mazonSESFullアクセス (を呼び出すアクセス許可を含む、ニーズに合わせてカスタマイズされたカスタムロールを使用することをお勧めします) `UpdateConfigurationSetSendingEnabled`。

> **Tip**
>
> ポリシーのリストの上部にある検索ボックスを使用すると、これらのポリシーをすばやく特定できます。

[Next: Review] を選択します。

7. [Review] ページで、[Name] にロールの名前を入力します。[ロールの作成] を選択します。

パート 3: Lambda 関数を作成する

IAM ロールを作成したら、設定セットの E メール送信を一時停止する Lambda 関数を作成できます。

Lambda 関数を作成するには

1. で AWS Lambda コンソールを開きますhttps://console.aws.amazon.com/lambda/。
2. リージョンセレクターを使用して、この Lambda 関数をデプロイするリージョンを選択します。

> **Note**
>
> この関数では、このステップで選択した AWS リージョン内の設定セットの E メール送信を一時停止します。複数のリージョンから E メールを送信する場合は、E メール送信を自動的に一時停止するリージョンごとにこのセクションの手順を繰り返します。

3. [Create function] を選択します。
4. [Create function] で [Author from scratch] を選択します。
5. [Author from scratch] で、次のステップを実行します。

   - [名前] に Lambda 関数の名前を入力します。
   - ランタイムには、Node.js 14x (または、現在選択リストで提供されているバージョン) を選択します。
   - [ロール] で、[Choose an existing role] を選択します。

- [既存のロール] には、the section called “パート 2: IAM ロールの作成” で作成した IAM ロールを選択します。

[Create function] を選択します。

6. コードエディタの [Function code] に、以下のコードを貼り付けます。

```
'use strict';

import {
    SES
}
from 'aws-sdk';

const ses = new SES();
const configSet = 'CONFIG_SET_NAME_HERE';

const params = {
    ConfigurationSetName: configSet,
    Enabled: false
};

export const handler = async (event) => {
    try {
        const data = await
 ses.updateConfigurationSetSendingEnabled(params).promise();

        console.log('Configuration Set Update:', data);

        return {
            statusCode: 200,
            body: JSON.stringify({
                message: 'Successfully paused email sending for configuration
 set.',
                data
            }),
        };
    }
    catch (err) {
        console.error('Error:', err.message);
        return {
            statusCode: 500,
            body: JSON.stringify({
```

```
                message: 'Failed to pause email sending for configuration set.',
                error: err.message
            }),
        };
    }
};
```

上記のコード*ConfigSet*の を、設定セットの名前に置き換えます。[Save] を選択します。

7. [Test] を選択します。[Configure test event] ウィンドウが表示されたら、[Event name] フィールドに名前を入力してから、[Create] を選択します。
8. ページ上部の通知バーに `Execution result: succeeded` と表示されていることを確認します。関数の実行に失敗した場合は、次のことを実行します。

   - the section called “パート 2: IAM ロールの作成” で作成した IAM ロールに正しいポリシーが含まれていることを確認します。
   - Lambda 関数のコードにエラーがないことを確認します。Lambda コードエディタでは、構文エラーやその他の潜在的な問題が自動的にハイライトされます。

パート 4: 設定セットの E メール送信の再有効化

the section called “パート 3: Lambda 関数を作成する” で Lambda 関数のアップロードとテストをする副作用は、設定セットの E メール送信が一時停止していることです。ほとんどの場合、CloudWatch アラームがトリガーされるまで、設定セットの送信を一時停止しないでください。

このセクションの手順では、E メール送信設定を再度有効にします。これらの手順を完了するには、AWS Command Line Interfaceをインストールして設定する必要があります。詳細については、AWS Command Line Interface ユーザーガイドを参照してください。

E メール送信を再有効化するには

1. コマンドラインで次のコマンドを入力して、設定セットの E メールの送信を再度有効にします。

   ```
   aws ses update-configuration-set-sending-enabled \
   --configuration-set-name ConfigSet \
   --enabled
   ```

   前述のコマンドで、 を E メール送信を一時停止する設定セットの名前*ConfigSet*に置き換えます。

2. コマンドラインで次のコマンドを入力して、E メールの送信が有効であることを確認します。

```
aws ses describe-configuration-set \
--configuration-set-name ConfigSet \
--configuration-set-attribute-names reputationOptions
```

このコマンドは、次の例と同様の出力を生成します。

```
{
    "ConfigurationSet": {
        "Name": "ConfigSet"
    },
    "ReputationOptions": {
        "ReputationMetricsEnabled": true,
        "SendingEnabled": true
    }
}
```

SendingEnabled の値が true の場合、設定セットの E メール送信は正常に再有効化されています。

パート 5: Amazon SNSトピックを作成する

アラームがトリガーされたときに が Lambda 関数を実行する CloudWatch には、まず Amazon SNS トピックを作成し、Lambda 関数をサブスクライブする必要があります。

Amazon SNSトピックを作成するには

1. https://console.aws.amazon.com/sns/v3/home で Amazon SNSコンソールを開きます。
2. リージョンセレクターを使用して、E メール送信を自動的に一時停止するリージョンを選択します。
3. ナビゲーションペインで、[トピック] を選択します。
4. [Create new topic] を選択します。
5. [Create new topic] ウィンドウで、[Topic name] にトピックの名前を入力します。必要に応じて、[Display name] フィールドにわかりやすい名前を入力することもできます。

   [トピックの作成] を選択します。

6. トピックのリストで、前のステップで作成したトピックの横にあるチェックボックスをオンにします。[Actions] メニューで、[Subscribe to topic] を選択します。
7. [Create subscription] ウィンドウで、以下の選択を行います。
   - [プロトコル] で [AWS Lambda] を選択します。
   - [Endpoint] に、the section called “パート 3: Lambda 関数を作成する” で作成した Lambda 関数を選択します。
   - [Version or alias] に [default] を選択します。
8. [Create subscription] を選択します。

パート 6: CloudWatch アラームを作成する

このセクションでは、メトリクスが特定のしきい値に達したときにトリガー CloudWatch されるアラームを で作成する手順について説明します。アラームがトリガーされると、 で作成した Amazon SNSトピックに通知が配信されthe section called “パート 5: Amazon SNSトピックを作成する”、 で作成した Lambda 関数が実行されますthe section called “パート 3: Lambda 関数を作成する”。

CloudWatch アラームを作成するには

1. で CloudWatch コンソールを開きますhttps://console.aws.amazon.com/cloudwatch/。
2. リージョンセレクターを使用して、E メール送信を自動的に一時停止するリージョンを選択します。
3. 左側のナビゲーションペインで、[Alarms] を選択します。
4. [Create Alarm] を選択します。
5. アラームの作成ウィンドウのSESメトリクスで、設定セットメトリクスを選択します。
6. [ses:configuration-set] 列で、アラームを作成する設定セットを検索します。[Metric Name] で、次のいずれかのオプションを選択します。
   - 評価。BounceRate– 設定セットの全体的なハードバウンス率が定義したしきい値を超えたときに、設定セットの E メール送信を一時停止する場合は、このメトリクスを選択します。
   - 評価。ComplaintRate– 設定セットの全体的な苦情率が定義したしきい値を超えたときに、設定セットの E メール送信を一時停止する場合は、このメトリクスを選択します。

   [Next (次へ)] を選択します。
7. 以下の手順を実行します。

- [Alarm Threshold] の [Name] に、アラームの名前を入力します。
- Whenever: Reputation.BounceRate または Whenever: Reputation.ComplaintRate で、アラームをトリガーするしきい値を指定します。

> **Note**
>
> Amazon SESアカウントの全体的なバウンス率が 10% を超える場合、または Amazon SESアカウントの全体的な苦情率が 0.5% を超える場合、Amazon SESアカウントは自動的にレビュー対象になります。 CloudWatch アラームをトリガーするバウンス率または苦情率を指定する場合は、アカウントがレビュー対象にならないように、これらの率よりはるかに低い値を使用することをお勧めします。

- アクション で、このアラームが発生するたびに、状態が ALARMを選択します。通知の送信先として、「」で作成した Amazon SNSトピックを選択しますthe section called “パート 5: Amazon SNSトピックを作成する”。

[Create Alarm] を選択します。

パート 7: ソリューションをテストする

ALARM 状態に入ったときに Lambda 関数を実行するようにアラームをテストできるようになりました。の CloudWatch API `SetAlarmState`オペレーションを使用して、アラームの状態を一時的に変更できます。

このセクションの手順はオプションですが、ソリューション全体が正しく設定されていることを確認するために、これらの手順を完了することをお勧めします。

ソリューションをテストするには

1. コマンドラインで次のコマンドを入力して、設定セットの E メールの送信ステータスを確認します。

```
aws ses describe-configuration-set --configuration-set-name ConfigSet
```

送信が設定セットに対して有効になっている場合、次の出力が表示されます。

```
{
```

```
    "ConfigurationSet": {
        "Name": "ConfigSet"
    },
    "ReputationOptions": {
        "ReputationMetricsEnabled": true,
        "SendingEnabled": true
    }
}
```

SendingEnabled の値が true の場合、現在設定セットの E メール送信は有効になっています。

2. コマンドラインで次のコマンドを入力して、アラーム状態を一時的に ALARM に変更します。

```
aws cloudwatch set-alarm-state \
--alarm-name MyAlarm \
--state-value ALARM \
--state-reason "Testing execution of Lambda function"
```

前述のコマンドMyAlarmの を、「」で作成したアラームの名前に置き換えますthe section called “パート 6: CloudWatch アラームを作成する”。

> **Note**
>
> このコマンドを実行すると、アラームの状態は OK から ALARM に切り替わり、数秒で OK に戻ります。これらのステータスの変更は、 CloudWatch コンソールのアラームの履歴タブで、または DescribeAlarmHistoryオペレーションを使用して表示できます。

3. コマンドラインで次のコマンドを入力して、設定セットの E メールの送信ステータスを確認します。

```
aws ses describe-configuration-set \
--configuration-set-name ConfigSet
```

Lambda 関数が正常に実行された場合、次の例のような出力が表示されます。

```
{
    "ConfigurationSet": {
        "Name": "ConfigSet"
    },
```

```
    "ReputationOptions": {
        "ReputationMetricsEnabled": true,
        "SendingEnabled": false
    }
}
```

SendingEnabled の値が false の場合、設定セットの E メール送信は無効になり、Lambda 関数が正常に実行されたことを示します。

4. the section called “パート 4: 設定セットの E メール送信の再有効化” の手順を実行して、設定セットの E メール送信を再度有効にします。

Amazon EventBridge を使用した SES イベントのモニタリング

EventBridge は、イベントを使用してアプリケーションコンポーネント同士を接続するサーバーレスサービスです。これにより、スケーラブルなイベント駆動型アプリケーションを簡単に構築できます。イベント駆動型アーキテクチャとは、イベントの発信と応答によって連携する、ゆるやかに結合されたソフトウェアシステムを構築するスタイルです。イベントは JSON 形式のメッセージで、通常はリソースや環境の変更、またはその他の管理イベントを表します。

特定の SES 機能は、EventBridge のデフォルトイベントバスに、イベント送信先の作成時に定義したイベントを生成して送信します。イベントバスは、イベントを受信するルーターであり、ゼロ個以上の送信先やターゲットに配信します。イベントバスに関連付けたルールによって、受信したイベントが評価されます。各ルールは、イベントがルールのパターンに一致するかどうかをチェックします。イベントが一致すると、EventBridge はイベントを指定されたターゲットに送信します。

SES は、機能のステータスが変化したり、更新した際、EventBridge にイベントを送信します。EventBridge ルールを使用すると、イベントを定義したターゲットにルートできます。これらのイベントはベストエフォート方式で配信されるため、順不同で配信される場合があります。

トピック

SES イベント

以下のイベントは SES の機能が生成し、EventBridge のデフォルトイベントバスに送信されます。各イベントタイプの詳細データなどの詳細については、「???」を参照してください。

Virtual Deliverability Manager アドバイザーのイベント

イベントタイプ	説明
アドバイザー推奨事項ステータスのオープン	Virtual Deliverability Manager アドバイザーで新しい推奨事項が開かれるたびに生成されるイベント。

イベントタイプ	説明
アドバイザー推奨事項ステータスの解決	Virtual Deliverability Manager アドバイザーで推奨事項が解決されるたびに生成されるイベント。

SES E メール送信イベント

イベントタイプ	説明
E メールがバウンスしました	受信者のメールサーバーが E メールを完全に拒否したハードバウンスです。(ソフトバウンスは、SES が一定期間にわたって再試行してもメールを配信できなかった場合に限ります。)
E メールがクリックされました	受取人はメール内の単一または複数のリンクをクリックしました。
E メールの苦情を受信しました	E メールは受信者のメールサーバーに正常に配信されましたが、受信者がスパムとしてマークしました。
E メール配信済み	SES は、受取人のメールサーバーに E メールを正常に配信しました。
E メール配信が遅延しました	一時的な問題が発生したため、E メールを受信者のメールサーバーに配信できませんでした。配信の遅延は、受信者の受信トレイがいっぱいになった場合や、受信側の電子メールサーバーで一時的な問題が発生した場合などに発生します。
E メールが開封されました	受信者はメッセージを受信し、E メールクライアントで開封しました。
E メールが拒否されました	SES は E メールを受け取りましたが、この E メールにウイルスが含まれていると判断したため、受信者のメールサーバーへの E メールの配信を試みませんでした。
E メールのレンダリングに失敗しました	テンプレートのレンダリングの問題により、この E メールは送信されませんでした。このイベントタイプは、テンプレートデータが見つからない場合や、テンプレートのパラメータとデータが一致しない場合に発生します。(このイベントタイプは、SendTemplatedEmail または SendBulkT

イベントタイプ	説明
	emplatedEmail API オペレーションを使用して E メールを送信する場合にのみ発生します。)
E メールが送信されました	送信リクエストが正常に完了し、SES はこのメッセージを受信者のメールサーバーに配信しようと試行します。(アカウントレベルまたはグローバル抑制が使用されている場合でも、SES により送信済みとしてカウントされますが、配信は抑制されます)。
E メールがサブスクライブしました	メールは正常に配信されましたが、受信者が E メールヘッダーの List-Unsubscribe またはフッターの Unsubscribe リンクをクリックして、サブスクリプションの設定を更新しました。

SES イベントスキーマリファレンス

AWS サービスのすべてのイベントには、イベントのソースである AWS サービス、イベントが生成された時刻、イベントが発生したアカウントと地域など、イベントに関するメタデータを含む共通のフィールドセットがあります。これらの一般的なフィールドの定義については、「EventBridge ユーザーガイド」の「イベント構造リファレンス」を参照してください。

さらに、各イベントには、その特定のイベントに固有のデータを含む detail フィールドがあります。以下のリファレンスでは、さまざまな SES イベントの詳細フィールドを定義しています。

EventBridge を使用して SES イベントの選択と管理を行う場合、以下の点に留意するのが有用です。

- SES からのすべてのイベントの source フィールドは、aws.ses に設定されています。
- detail-type フィールドはイベントタイプを指定します。「the section called “SES イベント”」のイベントタイプ表を参照してください。
- detail フィールドには、その特定のイベントに固有のデータが含まれます。

  Virtual Deliverability Manager など、一部のイベントタイプでは、詳細フィールドは、静的値の有限なセットから入力される比較的シンプルなデータ文字列です。逆に、E メール送信イベントの詳細フィールドは、E メールの送信日時のタイムスタンプ、受信者アドレス、その他のさまざまな E

メール属性など、静的値と動的値の両方を組み合わせた多くの詳細サブフィールドで構成されているため、より複雑になります。

トピック

Virtual Deliverability Manager アドバイザーのステータススキーマ

以下のスキーマリファレンスでは、Virtual Deliverability Manager アドバイザーのステータスイベント固有のフィールドを定義しています。

すべてのイベントスキーマ (`version`、`id`、`account` など) に表示される一般的なフィールドの定義は、「EventBridge ユーザーガイド」の「イベント構造リファレンス」に記載されています。`source` および `detail-type` フィールドには SES イベントの SES 固有の値が含まれているため、以下のリファレンスに含まれています。

`source`

イベントを発生させたサービスを識別します。SES イベントの場合、この値は `aws.ses` です。

`detail-type`

イベントのタイプを示します。

このフィールドの値は、「the section called “SES イベント”」の 「Virtual Deliverability Manager アドバイザーイベント」表に記載されています。

`detail`

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

このフィールドで想定される値は、以下のとおりです。

- `DKIM verification is not enabled.`
- `DKIM verification has failed.`
- `DKIM signing key length is below 2048 bits.`
- `DMARC configuration was not found.`
- `DMARC configuration could not be parsed.`

- DKIM record was not found.
- DKIM record is not aligned.
- MAIL FROM record is not aligned.
- SPF record was not found.
- SPF record for Amazon SES was not found.
- SPF all qualifier is missing.
- An SPF configuration issue was found.
- BIMI record not found or configured without default selector.
- BIMI has malformed TXT record.

Example 例: Virtual Deliverability Manager アドバイザーのステータスイベント

以下は、イベントタイプが Advisor Recommendation Status Open の場合の Virtual Deliverability Manager アドバイザーのステータスイベントの例です。この例の詳細なイベント値は SPF record was not found. です。

```
{
 "version": "0",
  "id": "abcd9999-ef33-0123-90ab-abcdef666666",
  "detail-type": "Advisor Recommendation Status Open",
  "source": "aws.ses",
  "account": "012345678901",
  "time": "2023-11-15T17:00:59Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ses:us-east-1:012345678901:identity/vdm.events-publishing.cajun.syster-
games.example.com"
  ],
  "detail": { "version": "1.0.0", "data": "SPF record was not found." }
}
```

SES E メール送信ステータスのスキーマ

以下のスキーマリファレンスでは、SES E　メール送信ステータスイベント固有のフィールドを定義しています。

すべてのイベントスキーマ (version、id、account など) に表示される一般的なフィールドの定義は、「EventBridge ユーザーガイド」の「イベント構造リファレンス」に記載されていま

す。source および detail-type フィールドには SES イベントの SES 固有の値が含まれているため、以下のリファレンスに含まれています。

`source`

イベントを発生させたサービスを識別します。SES イベントの場合、この値は aws.ses です。

`detail-type`

イベントのタイプを示します。

このフィールドの値は、「the section called “SES イベント”」の 「SES E メール送信イベント」表に記載されています。

`detail`

イベントに関する情報を含む JSON オブジェクト。このフィールドの内容は、イベントを生成するサービスによって決まります。

このフィールドに指定できる値は、特定の時点で送信される一意の E メールごとに生成される静的値と動的値で構成されるため、ここにすべては一覧表示できません。ここでは、このフィールドに含めることができるタイプのデータを認識しておく目的で、例を示します。すべての E メール送信イベントタイプの詳細データ例は、EventBridge Sandbox を使用して取得できます。「EventBridge でサンプルイベントを指定する」を参照してください。

SES E メール送信イベント Email Rendering Failed に対して生成された詳細データの例:

```
...,
  "detail": {
    "eventType": "Rendering Failure",
    "mail": {
      "timestamp": "2018-01-22T18:43:06.197Z",
      "source": "sender@example.com",
      "sourceArn": "arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
      "sendingAccountId": "123456789012",
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "destination": ["recipient@example.com"],
      "headersTruncated": false,
      "tags": {
        "ses:configuration-set": ["ConfigSet"]
      }
    },
    "failure": {
```

```
      "errorMessage": "Attribute 'attributeName' is not present in the rendering
  data.",
      "templateName": "MyTemplate"
    }
  }
```

Example 例: E メール送信ステータスのイベント

イベントタイプ Email Rendering Failed の E メール送信ステータスイベント全体の例は、以下のとおりです。この例の詳細イベント値は、特定の E メールの E メール送信イベントに基づく静的値と動的値の組み合わせです。

```
{
  "version": "0",
  "id": "12a18625-3328-fafd-2809-a5e16004f112",
  "detail-type": "Email Rendering Failed",
  "source": "aws.ses",
  "account": "123456789012",
  "time": "2023-07-17T16:48:05Z",
  "region": "us-east-1",
  "resources": ["arn:aws:ses:us-east-1:123456789012:identity/example.com"],
  "detail": {
    "eventType": "Rendering Failure",
    "mail": {
      "timestamp": "2018-01-22T18:43:06.197Z",
      "source": "sender@example.com",
      "sourceArn": "arn:aws:ses:us-east-1:123456789012:identity/sender@example.com",
      "sendingAccountId": "123456789012",
      "messageId": "EXAMPLE7c191be45-e9aedb9a-02f9-4d12-a87d-dd0099a07f8a-000000",
      "destination": ["recipient@example.com"],
      "headersTruncated": false,
      "tags": {
        "ses:configuration-set": ["ConfigSet"]
      }
    },
    "failure": {
      "errorMessage": "Attribute 'attributeName' is not present in the rendering
 data.",
      "templateName": "MyTemplate"
    }
  }
}
```

SES イベントでの EventBridge の使用

デフォルトで SES は、EventBridge のデフォルトイベントバスにイベントを送信します。デフォルトイベントバスにルールを作成して、EventBridge が 1 つ以上の指定されたターゲットに送信する特定のイベントを特定できます。各ルールには、EventBridge がイベントバスに到着したイベントの照合 (照合) に使用するイベントパターンが含まれています。イベントが指定されたルールのイベントパターンに一致すると、EventBridge はルールで指定されたターゲットにイベントを送信します。

EventBridge でのイベントパターンの定義は、通常、新しいルールの作成や既存のルールの編集という、より大きなプロセスの一環として行います。EventBridge ルールを作成する方法については、「EventBridge ユーザーガイド」の「イベントに反応する Amazon EventBridge ルールの作成」を参照してください。

EventBridge のサンドボックス機能を使用すると、まずルールを作成したり編集したりする必要なく、イベントパターンを迅速に定義し、サンプルイベントを使用してパターンが目的のイベントと一致することを確認できます。サンドボックスの詳細な使用方法については、EventBridge ユーザーガイドの「EventBridge サンドボックスを使用したイベントパターンのテスト」を参照してください。

EventBridge サンドボックスで SES サンプルイベントを指定する

SES イベントのサンプルイベントを選択して、作成したイベントパターンのテストに使用できます。

EventBridge サンドボックスで SES サンプルイベントを指定するには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [デベロッパーリソース]、[サンドボックス] の順に選択し、[サンドボックス] ページで [イベントパターン] タブを選択します。
3. [Event source] (イベントソース) で、[AWS events or EventBridge partner events] (イベントまたは EventBridge パートナーイベント) を選択します。
4. [サンプルイベント] セクションの [サンプルイベントタイプ] で、[AWS イベント] を選択します。
5. [サンプルイベント] で、[SES] までスクロールし、目的の SES イベントを選択します。

   EventBridge は、該当するイベントタイプのサンプルイベントとそのすべての詳細データを表示します。

このイベントを使用して、[イベントパターン] セクションで作成したイベントパターンをテストしたり、パターンテスト用の独自のサンプルイベントを作成するための基盤として使用したりできます。

SES イベントのイベントパターンの作成とテスト

前のセクションで説明したとおり、サンプルイベントを選択したら、イベントパターンを作成し、サンプルイベントを使用して、必要となるイベントと一致していることを確認できます。

EventBridge サンドボックス内の SES イベントと一致するイベントパターンを作成してテストするには

1. Amazon EventBridge コンソール (https://console.aws.amazon.com/events/) を開きます。
2. ナビゲーションペインで [デベロッパーリソース]、[サンドボックス] の順に選択し、[サンドボックス] ページで [イベントパターン] タブを選択します。
3. [イベントソース] では、[AWSイベント] または [EventBridge パートナーイベント] を選択し、前のセクションで説明したとおり、テストするサンプルイベントを選択します。
4. [作成方法] まで下にスクロールして、[パターンフォームを使用する] を選択します。
5. [イベントパターン] セクションの [イベントソース] で [AWS サービス] を選択します。
6. [AWS のサービス] で [SES] を選択します。
7. [イベントタイプ] で、一致させたい SES イベントタイプを選択します。

   EventBridge は、選択した SES イベントと一致する最小のイベントパターンを `source` と `detail-type` フィールドで構成して表示します。

   この 2 つの例では、最初のイベントパターンはすべての `Advisor Recommendation Status Resolved` イベントと一致し、2 番目のイベントパターンはすべての `Email Bounced` イベントと一致します。

```
{
  "source": ["aws.ses"],
  "detail-type": ["Advisor Recommendation Status Resolved"]
}
```

```
{
  "source": ["aws.ses"],
```

```
  "detail-type": ["Email Bounced"]
}
```

8. イベントパターンを変更するには、[パターンを編集] をクリックして、JSON エディタで変更を行います。

   1 つ以上の詳細データフィールドの値と照合することもできます。これには、1 つのフィールド値に対して複数の可能な値を指定することも含まれます。

   次の例では、同じ詳細値を持つすべての Virtual Deliverability Manager アドバイザーイベントを検索するために、`DKIM record was not found` と指定された `data` フィールド値を持つ、生成された最小イベントパターンに詳細フィールドが追加されました。

```
{
  "source": ["aws.ses"],
  "detail-type": ["Advisor Recommendation Status Resolved"],
  "detail": {
    "data": ["DKIM record was not found."]
  }
}
```

   この例では、2024-08-05 に noreply@example.com から送信されたすべての E メールがバウンスしたことで生成されたイベントを報告するために、詳細サブフィールドが追加されました。(ここでは、コンテンツフィルタリングの一環として、プレフィックスフィルタリングが使用されています)。

```
{
  "source": ["aws.ses"],
  "detail-type": ["Email Bounced"],
  "detail": {
    "mail": {
      "timestamp": [{
        "prefix": "2024-08-05"
      }],
      "source": ["noreply@example.com"]
    }
  }
}
```

   「EventBridge ユーザーガイド」の「イベントパターン」を読んでおくことが重要です。このセクションでは、JSON エディタで入力するイベントパターンの値は配列と見なされるため、角括

弧 [. . .] で囲む必要があることが説明されています。この点と、高度なイベントパターンを構築する方法に関する詳細も提供されています。

9. イベントパターンが、上記の [サンプルイベント] ペインで指定したサンプルイベントと一致するかをテストするには、[パターンのテスト] をクリックします。一致すると、JSON エディタの下部に緑色のバナーが表示され、「サンプルイベントがイベントパターンと一致しました」と表示されます。
10. [パターンのテスト] をクリックした後のエラーをトラブルシューティングするには:
    - JSON 関連のエラーが発生した場合、メッセージには「イベントパターンが不正です」などの理由が表示されます。理由:「data」は、行: 5、列: 14 のオブジェクトまたは配列である必要があります。これを修正するには、5 行目の値を角括弧 [. . .] で囲みます。
    - サンプルイベントの値とイベントパターンが一致しない場合は、「サンプルイベントがイベントパターンと一致しませんでした」というメッセージが表示されます。これは、テストする単数または複数の値が、サンプルイベントジェネレータによって生成されたサンプル値とは異なることを意味します。これを修正するには、残りのステップを続行します。
11. イベントパターンのテストを正常に完了するためにサンプルイベントのサンプル値を変更するには、[サンプルイベント] ペインで、JSON エディタの下にある [コピー] をクリックします。
12. エディタの上部にある [サンプルイベントタイプ] の [独自のサンプルイベントを入力] の横にあるラジオボタンをオンにします
13. サンプルイベントを JSON エディタに貼り付けます。イベントパターンで使用しているフィールドについては、その同じフィールドの値をイベントパターンで指定した値に合わせて置き換えます。
14. イベントパターンペインまで下にスクロールして、[パターンのテスト] をもう一度クリックします。すべての値が適切に入力されて一致すると、JSON エディタの下部に緑色のバナーが表示され、「サンプルイベントがイベントパターンと一致しました」と表示されます。

追加の EventBridge リソース

EventBridge を使用してイベントを処理および管理する方法の詳細については、Amazon EventBridge ユーザーガイドの以下のトピックを参照してください。

- イベントバスの仕組みの詳細については、「Amazon EventBridge イベントバス」を参照してください。
- イベント構造については、イベントのページを参照してください。

- イベントをルールと照合するときに使用する EventBridge のイベントパターンの構築については、イベントパターンのページを参照してください。
- EventBridge が処理するイベントを指定するルールを作成する方法については、ルールのページを参照してください。
- EventBridge がマッチしたイベントを送信するサービスやその他の宛先を指定する方法については、ターゲットのページを参照してください。

AWS SDK を使用した Amazon SES のコード例

次のコード例は、AWS ソフトウェア開発キット (SDK) による Amazon SES の使用方法を示しています。

AWS SDK デベロッパーガイドとコード例の完全なリストについては、「SES での Amazon の使用 AWS SDK」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- SES を使用した Amazon のコード例 AWS SDKs
  - SES を使用した Amazon の基本的な例 AWS SDKs
    - SES を使用した Amazon のアクション AWS SDKs
      - CreateReceiptFilter で を使用する AWS SDK
      - CreateReceiptRule で を使用する AWS SDK
      - CreateReceiptRuleSet で を使用する AWS SDK
      - CreateTemplate で を使用する AWS SDK
      - または DeleteIdentityAWS SDKで を使用する CLI
      - DeleteReceiptFilter で を使用する AWS SDK
      - DeleteReceiptRule で を使用する AWS SDK
      - DeleteReceiptRuleSet で を使用する AWS SDK
      - DeleteTemplate で を使用する AWS SDK
      - DescribeReceiptRuleSet で を使用する AWS SDK
      - または GetIdentityVerificationAttributesで を使用する AWS SDK CLI
      - または GetSendQuotaで を使用する AWS SDK CLI
      - GetSendStatistics で を使用する CLI
      - GetTemplate で を使用する AWS SDK
      - または ListIdentitiesAWS SDKで を使用する CLI
      - ListReceiptFilters で を使用する AWS SDK
      - ListTemplates で を使用する AWS SDK
      - SendBulkTemplatedEmail で を使用する AWS SDK
      - または SendEmailで を使用する AWS SDK CLI

SES を使用した Amazon のコード例 AWS SDKs

次のコード例は、 AWS ソフトウェア開発キット () SESで Amazon を使用する方法を示しています SDK。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

コードの例

    - または GetIdentityVerificationAttributesで を使用する AWS SDK CLI
    - または GetSendQuotaで を使用する AWS SDK CLI
    - GetSendStatistics で を使用する CLI
    - GetTemplate で を使用する AWS SDK
    - または ListIdentitiesAWS SDKで を使用する CLI
    - ListReceiptFilters で を使用する AWS SDK
    - ListTemplates で を使用する AWS SDK
    - SendBulkTemplatedEmail で を使用する AWS SDK
    - または SendEmailで を使用する AWS SDK CLI
    - または SendRawEmailAWS SDKで を使用する CLI
    - SendTemplatedEmail で を使用する AWS SDK
    - UpdateTemplate で を使用する AWS SDK
    - または VerifyDomainIdentityで を使用する AWS SDK CLI
    - または VerifyEmailIdentityAWS SDKで を使用する CLI
- SES を使用した Amazon のシナリオ AWS SDKs
    - Amazon Transcribe ストリーミングアプリケーションを構築する
    - を使用して Amazon E SESメールとドメイン ID をあるリージョンから別の AWS リージョンにコピーする AWS SDK
    - DynamoDB データを追跡するウェブアプリケーションを作成する
    - Amazon Redshift アイテムトラッカーの作成
    - Aurora Serverless 作業項目トラッカーの作成
    - を使用して Amazon Rekognition でイメージPPE内で検出する AWS SDK
    - を使用して Amazon Rekognition でイメージ内のオブジェクトを検出する AWS SDK
    - を使用して Amazon Rekognition でビデオ内の人物とオブジェクトを検出する AWS SDK
    - Amazon SESSMTPエンドポイントに接続するための認証情報を生成する
    - Step Functions を使用して Lambda 関数を呼び出す
    - SES を使用して E メール ID を検証し、Amazon でメッセージを送信する AWS SDK

SES を使用した Amazon の基本的な例 AWS SDKs

次のコード例は、 で AWS Amazon Simple Email Service の基本を使用する方法を示しています SDKs。

例

- SES を使用した Amazon のアクション AWS SDKs
  - CreateReceiptFilter で を使用する AWS SDK
  - CreateReceiptRule で を使用する AWS SDK
  - CreateReceiptRuleSet で を使用する AWS SDK
  - CreateTemplate で を使用する AWS SDK
  - または DeleteIdentityAWS SDKで を使用する CLI
  - DeleteReceiptFilter で を使用する AWS SDK
  - DeleteReceiptRule で を使用する AWS SDK
  - DeleteReceiptRuleSet で を使用する AWS SDK
  - DeleteTemplate で を使用する AWS SDK
  - DescribeReceiptRuleSet で を使用する AWS SDK
  - または GetIdentityVerificationAttributesで を使用する AWS SDK CLI
  - または GetSendQuotaで を使用する AWS SDK CLI
  - GetSendStatistics で を使用する CLI
  - GetTemplate で を使用する AWS SDK
  - または ListIdentitiesAWS SDKで を使用する CLI
  - ListReceiptFilters で を使用する AWS SDK
  - ListTemplates で を使用する AWS SDK
  - SendBulkTemplatedEmail で を使用する AWS SDK
  - または SendEmailで を使用する AWS SDK CLI
  - または SendRawEmailAWS SDKで を使用する CLI
  - SendTemplatedEmail で を使用する AWS SDK
  - UpdateTemplate で を使用する AWS SDK
  - または VerifyDomainIdentityで を使用する AWS SDK CLI
  - または VerifyEmailIdentityAWS SDKで を使用する CLI

SES を使用した Amazon のアクション AWS SDKs

次のコード例は、 を使用して個々の Amazon SESアクションを実行する方法を示しています AWS SDKs。各例には へのリンクが含まれており GitHub、コードの設定と実行の手順を確認できます。

これらの抜粋は Amazon を呼び出しSESAPI、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。アクションは SES を使用した Amazon のシナリオ AWS SDKs のコンテキスト内で確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細なリストについては、Amazon Simple Email Service APIリファレンスを参照してください。

例

- CreateReceiptFilter で を使用する AWS SDK
- CreateReceiptRule で を使用する AWS SDK
- CreateReceiptRuleSet で を使用する AWS SDK
- CreateTemplate で を使用する AWS SDK
- または DeleteIdentityAWS SDKで を使用する CLI
- DeleteReceiptFilter で を使用する AWS SDK
- DeleteReceiptRule で を使用する AWS SDK
- DeleteReceiptRuleSet で を使用する AWS SDK
- DeleteTemplate で を使用する AWS SDK
- DescribeReceiptRuleSet で を使用する AWS SDK
- または GetIdentityVerificationAttributesで を使用する AWS SDK CLI
- または GetSendQuotaで を使用する AWS SDK CLI
- GetSendStatistics で を使用する CLI
- GetTemplate で を使用する AWS SDK
- または ListIdentitiesAWS SDKで を使用する CLI
- ListReceiptFilters で を使用する AWS SDK
- ListTemplates で を使用する AWS SDK
- SendBulkTemplatedEmail で を使用する AWS SDK
- または SendEmailで を使用する AWS SDK CLI
- または SendRawEmailAWS SDKで を使用する CLI
- SendTemplatedEmail で を使用する AWS SDK

- UpdateTemplate で を使用する AWS SDK
- または VerifyDomainIdentityで を使用する AWS SDK CLI
- または VerifyEmailIdentityAWS SDKで を使用する CLI

`CreateReceiptFilter` で を使用する AWS SDK

以下のコード例は、`CreateReceiptFilter` の使用方法を示しています。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt filter..
/*!
  \param receiptFilterName: The name for the receipt filter.
  \param cidr: IP address or IP address range in Classless Inter-Domain Routing
 (CIDR) notation.
  \param policy: Block or allow enum of type ReceiptFilterPolicy.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptFilter(const Aws::String &receiptFilterName,
                                      const Aws::String &cidr,
                                      Aws::SES::Model::ReceiptFilterPolicy
 policy,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::CreateReceiptFilterRequest createReceiptFilterRequest;
    Aws::SES::Model::ReceiptFilter receiptFilter;
    Aws::SES::Model::ReceiptIpFilter receiptIpFilter;
    receiptIpFilter.SetCidr(cidr);
    receiptIpFilter.SetPolicy(policy);
    receiptFilter.SetName(receiptFilterName);
    receiptFilter.SetIpFilter(receiptIpFilter);
```

```
    createReceiptFilterRequest.SetFilter(receiptFilter);
    Aws::SES::Model::CreateReceiptFilterOutcome createReceiptFilterOutcome =
 sesClient.CreateReceiptFilter(
            createReceiptFilterRequest);
    if (createReceiptFilterOutcome.IsSuccess()) {
        std::cout << "Successfully created receipt filter." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt filter: " <<
                  createReceiptFilterOutcome.GetError().GetMessage() <<
 std::endl;
    }

    return createReceiptFilterOutcome.IsSuccess();
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import {
  CreateReceiptFilterCommand,
  ReceiptFilterPolicy,
} from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const createCreateReceiptFilterCommand = ({ policy, ipOrRange, name }) => {
  return new CreateReceiptFilterCommand({
    Filter: {
      IpFilter: {
```

```
        Cidr: ipOrRange, // string, either a single IP address (10.0.0.1) or an
 IP address range in CIDR notation (10.0.0.1/24)).
        Policy: policy, // enum ReceiptFilterPolicy, email traffic from the
 filtered addressesOptions.
      },
      /*
        The name of the IP address filter. Only ASCII letters, numbers,
 underscores, or dashes.
        Must be less than 64 characters and start and end with a letter or
 number.
       */
      Name: name,
    },
  });
};

const FILTER_NAME = getUniqueName("ReceiptFilter");

const run = async () => {
  const createReceiptFilterCommand = createCreateReceiptFilterCommand({
    policy: ReceiptFilterPolicy.Allow,
    ipOrRange: "10.0.0.1",
    name: FILTER_NAME,
  });

  try {
    return await sesClient.send(createReceiptFilterCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_receipt_filter(self, filter_name, ip_address_or_range, allow):
        """
        Creates a filter that allows or blocks incoming mail from an IP address
 or
        range.

        :param filter_name: The name to give the filter.
        :param ip_address_or_range: The IP address or range to block or allow.
        :param allow: When True, incoming mail is allowed from the specified IP
                      address or range; otherwise, it is blocked.
        """
        try:
            policy = "Allow" if allow else "Block"
            self.ses_client.create_receipt_filter(
                Filter={
                    "Name": filter_name,
                    "IpFilter": {"Cidr": ip_address_or_range, "Policy": policy},
                }
            )
            logger.info(
```

```
                "Created receipt filter %s to %s IP of %s.",
                filter_name,
                policy,
                ip_address_or_range,
            )
        except ClientError:
            logger.exception("Couldn't create receipt filter %s.", filter_name)
            raise
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES でのAmazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`CreateReceiptRule` で を使用する AWS SDK

以下のコード例は、`CreateReceiptRule` の使用方法を示しています。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
  \param receiptRuleName: The name for the receipt rule.
  \param s3BucketName: The name of the S3 bucket for incoming mail.
  \param s3ObjectKeyPrefix: The prefix for the objects in the S3 bucket.
  \param ruleSetName: The name of the rule set where the receipt rule is added.
  \param recipients: Aws::Vector of recipients.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
```

```
 */
bool AwsDoc::SES::createReceiptRule(const Aws::String &receiptRuleName,
                                    const Aws::String &s3BucketName,
                                    const Aws::String &s3ObjectKeyPrefix,
                                    const Aws::String &ruleSetName,
                                    const Aws::Vector<Aws::String> &recipients,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleRequest createReceiptRuleRequest;

    Aws::SES::Model::S3Action s3Action;
    s3Action.SetBucketName(s3BucketName);
    s3Action.SetObjectKeyPrefix(s3ObjectKeyPrefix);

    Aws::SES::Model::ReceiptAction receiptAction;
    receiptAction.SetS3Action(s3Action);

    Aws::SES::Model::ReceiptRule receiptRule;
    receiptRule.SetName(receiptRuleName);
    receiptRule.WithRecipients(recipients);

    Aws::Vector<Aws::SES::Model::ReceiptAction> receiptActionList;
    receiptActionList.emplace_back(receiptAction);
    receiptRule.SetActions(receiptActionList);

    createReceiptRuleRequest.SetRuleSetName(ruleSetName);
    createReceiptRuleRequest.SetRule(receiptRule);

    auto outcome = sesClient.CreateReceiptRule(createReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt rule. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { CreateReceiptRuleCommand, TlsPolicy } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");
const RULE_NAME = getUniqueName("RuleName");
const S3_BUCKET_NAME = getUniqueName("S3BucketName");

const createS3ReceiptRuleCommand = ({
  bucketName,
  emailAddresses,
  name,
  ruleSet,
}) => {
  return new CreateReceiptRuleCommand({
    Rule: {
      Actions: [
        {
          S3Action: {
            BucketName: bucketName,
            ObjectKeyPrefix: "email",
          },
        },
      ],
      Recipients: emailAddresses,
      Enabled: true,
```

```
      Name: name,
      ScanEnabled: false,
      TlsPolicy: TlsPolicy.Optional,
    },
    RuleSetName: ruleSet, // Required
  });
};

const run = async () => {
  const s3ReceiptRuleCommand = createS3ReceiptRuleCommand({
    bucketName: S3_BUCKET_NAME,
    emailAddresses: ["email@example.com"],
    name: RULE_NAME,
    ruleSet: RULE_SET_NAME,
  });

  try {
    return await sesClient.send(s3ReceiptRuleCommand);
  } catch (err) {
    console.log("Failed to create S3 receipt rule.", err);
    throw err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Amazon が受信 E メールのコピーを配置SESできる Amazon S3 バケットを作成し、受信 E メールを特定の受信者リストのバケットにコピーするルールを作成します。

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_bucket_for_copy(self, bucket_name):
        """
        Creates a bucket that can receive copies of emails from Amazon SES. This
        includes adding a policy to the bucket that grants Amazon SES permission
        to put objects in the bucket.

        :param bucket_name: The name of the bucket to create.
        :return: The newly created bucket.
        """
        allow_ses_put_policy = {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "AllowSESPut",
                    "Effect": "Allow",
                    "Principal": {"Service": "ses.amazonaws.com"},
                    "Action": "s3:PutObject",
                    "Resource": f"arn:aws:s3:::{bucket_name}/*",
                }
            ],
        }
        bucket = None
        try:
            bucket = self.s3_resource.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration={
                    "LocationConstraint":
 self.s3_resource.meta.client.meta.region_name
                },
            )
            bucket.wait_until_exists()
```

```
            bucket.Policy().put(Policy=json.dumps(allow_ses_put_policy))
            logger.info("Created bucket %s to receive copies of emails.",
bucket_name)
        except ClientError:
            logger.exception("Couldn't create bucket to receive copies of
emails.")
            if bucket is not None:
                bucket.delete()
            raise
        else:
            return bucket


    def create_s3_copy_rule(
        self, rule_set_name, rule_name, recipients, bucket_name, prefix
    ):
        """
        Creates a rule so that all emails received by the specified recipients
are
        copied to an Amazon S3 bucket.

        :param rule_set_name: The name of a previously created rule set to
contain
                              this rule.
        :param rule_name: The name to give the rule.
        :param recipients: When an email is received by one of these recipients,
it
                           is copied to the Amazon S3 bucket.
        :param bucket_name: The name of the bucket to receive email copies. This
                            bucket must allow Amazon SES to put objects into it.
        :param prefix: An object key prefix to give the emails copied to the
bucket.
        """
        try:
            self.ses_client.create_receipt_rule(
                RuleSetName=rule_set_name,
                Rule={
                    "Name": rule_name,
                    "Enabled": True,
                    "Recipients": recipients,
                    "Actions": [
                        {
                            "S3Action": {
                                "BucketName": bucket_name,
```

```
                            "ObjectKeyPrefix": prefix,
                        }
                    }
                ],
            },
        )
        logger.info(
            "Created rule %s to copy mail received by %s to bucket %s.",
            rule_name,
            recipients,
            bucket_name,
        )
    except ClientError:
        logger.exception("Couldn't create rule %s.", rule_name)
        raise
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES でのAmazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

CreateReceiptRuleSet で を使用する AWS SDK

以下のコード例は、CreateReceiptRuleSet の使用方法を示しています。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
```

```
  \param ruleSetName: The name of the rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createReceiptRuleSet(const Aws::String &ruleSetName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleSetRequest createReceiptRuleSetRequest;

    createReceiptRuleSetRequest.SetRuleSetName(ruleSetName);

    Aws::SES::Model::CreateReceiptRuleSetOutcome outcome =
 sesClient.CreateReceiptRuleSet(
            createReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule set." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt rule set. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { CreateReceiptRuleSetCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");

const createCreateReceiptRuleSetCommand = (ruleSetName) => {
  return new CreateReceiptRuleSetCommand({ RuleSetName: ruleSetName });
};

const run = async () => {
  const createReceiptRuleSetCommand =
    createCreateReceiptRuleSetCommand(RULE_SET_NAME);

  try {
    return await sesClient.send(createReceiptRuleSetCommand);
  } catch (err) {
    console.log("Failed to create receipt rule set", err);
    return err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
```

```
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def create_receipt_rule_set(self, rule_set_name):
        """
        Creates an empty rule set. Rule sets contain individual rules and can be
        used to organize rules.

        :param rule_set_name: The name to give the rule set.
        """
        try:
            self.ses_client.create_receipt_rule_set(RuleSetName=rule_set_name)
            logger.info("Created receipt rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't create receipt rule set %s.",
rule_set_name)
            raise
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES でのAmazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

CreateTemplate で を使用する AWS SDK

以下のコード例は、CreateTemplate の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- E メール ID を検証してメッセージを送信する

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Create an email template.
    /// </summary>
    /// <param name="name">Name of the template.</param>
    /// <param name="subject">Email subject.</param>
    /// <param name="text">Email body text.</param>
    /// <param name="html">Email HTML body text.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string name, string subject,
string text,
        string html)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.CreateTemplateAsync(
                new CreateTemplateRequest
                {
                    Template = new Template
                    {
                        TemplateName = name,
                        SubjectPart = subject,
                        TextPart = text,
                        HtmlPart = html
                    }
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
```

```
            Console.WriteLine("CreateEmailTemplateAsync failed with exception: "
 + ex.Message);
        }

        return success;
    }
```

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

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Create an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::createTemplate(const Aws::String &templateName,
                                 const Aws::String &htmlPart,
                                 const Aws::String &subjectPart,
                                 const Aws::String &textPart,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateTemplateRequest createTemplateRequest;
    Aws::SES::Model::Template aTemplate;
```

```
    aTemplate.SetTemplateName(templateName);
    aTemplate.SetHtmlPart(htmlPart);
    aTemplate.SetSubjectPart(subjectPart);
    aTemplate.SetTextPart(textPart);

    createTemplateRequest.SetTemplate(aTemplate);

    Aws::SES::Model::CreateTemplateOutcome outcome = sesClient.CreateTemplate(
            createTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created template." << templateName << "."
                  << std::endl;
    }
    else {
        std::cerr << "Error creating template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { CreateTemplateCommand } from "@aws-sdk/client-ses";
```

```
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const TEMPLATE_NAME = getUniqueName("TestTemplateName");

const createCreateTemplateCommand = () => {
  return new CreateTemplateCommand({
    /**
     * The template feature in Amazon SES is based on the Handlebars template
 system.
     */
    Template: {
      /**
       * The name of an existing template in Amazon SES.
       */
      TemplateName: TEMPLATE_NAME,
      HtmlPart: `
        <h1>Hello, {{contact.firstName}}!</h1>
        <p>
        Did you know Amazon has a mascot named Peccy?
        </p>
      `,
      SubjectPart: "Amazon Tip",
    },
  });
};

const run = async () => {
  const createTemplateCommand = createCreateTemplateCommand();

  try {
    return await sesClient.send(createTemplateCommand);
  } catch (err) {
    console.log("Failed to create template.", err);
    return err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
 html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def create_template(self, name, subject, text, html):
        """
        Creates an email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
```

```
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.create_template(Template=template)
            logger.info("Created template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't create template %s.", name)
            raise
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES でのAmazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **DeleteIdentity**AWS SDKで を使用する CLI

以下のコード例は、DeleteIdentity の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- E メール ID を検証してメッセージを送信する

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Delete an email identity.
    /// </summary>
    /// <param name="identityEmail">The identity email to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteIdentityAsync(string identityEmail)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.DeleteIdentityAsync(
                new DeleteIdentityRequest
                {
                    Identity = identityEmail
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("DeleteIdentityAsync failed with exception: " +
ex.Message);
        }

        return success;
    }
```

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

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Delete the specified identity (an email address or a domain).
/*!
  \param identity: The identity to delete.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteIdentity(const Aws::String &identity,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteIdentityRequest deleteIdentityRequest;

    deleteIdentityRequest.SetIdentity(identity);

    Aws::SES::Model::DeleteIdentityOutcome outcome = sesClient.DeleteIdentity(
            deleteIdentityRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted identity." << std::endl;
    }
    else {
        std::cerr << "Error deleting identity. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

CLI

AWS CLI

ID を削除するには

次の例では、 delete-identity コマンドを使用して、Amazon で検証された ID のリストから ID を削除しますSES。

```
aws ses delete-identity --identity user@example.com
```

検証済み ID の詳細については、Amazon SES Simple Email Service デベロッパーガイドの「Amazon での E メールアドレスとドメインの検証」を参照してください。

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { DeleteIdentityCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const IDENTITY_EMAIL = "fake@example.com";

const createDeleteIdentityCommand = (identityName) => {
  return new DeleteIdentityCommand({
    Identity: identityName,
  });
```

```
};

const run = async () => {
  const deleteIdentityCommand = createDeleteIdentityCommand(IDENTITY_EMAIL);

  try {
    return await sesClient.send(deleteIdentityCommand);
  } catch (err) {
    console.log("Failed to delete identity.", err);
    return err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def delete_identity(self, identity):
        """
        Deletes an identity.
```

```
        :param identity: The identity to remove.
        """
        try:
            self.ses_client.delete_identity(Identity=identity)
            logger.info("Deleted identity %s.", identity)
        except ClientError:
            logger.exception("Couldn't delete identity %s.", identity)
            raise
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

DeleteReceiptFilter で を使用する AWS SDK

以下のコード例は、DeleteReceiptFilter の使用方法を示しています。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt filter.
/*!
  \param receiptFilterName: The name for the receipt filter.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptFilter(const Aws::String &receiptFilterName,
                                      const Aws::Client::ClientConfiguration
 &clientConfiguration) {
```

```
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptFilterRequest deleteReceiptFilterRequest;

    deleteReceiptFilterRequest.SetFilterName(receiptFilterName);

    Aws::SES::Model::DeleteReceiptFilterOutcome outcome =
 sesClient.DeleteReceiptFilter(
            deleteReceiptFilterRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt filter." << std::endl;
    }
    else {
        std::cerr << "Error deleting receipt filter. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { DeleteReceiptFilterCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

const RECEIPT_FILTER_NAME = getUniqueName("ReceiptFilterName");
```

```
const createDeleteReceiptFilterCommand = (filterName) => {
  return new DeleteReceiptFilterCommand({ FilterName: filterName });
};

const run = async () => {
  const deleteReceiptFilterCommand =
    createDeleteReceiptFilterCommand(RECEIPT_FILTER_NAME);

  try {
    return await sesClient.send(deleteReceiptFilterCommand);
  } catch (err) {
    console.log("Error deleting receipt filter.", err);
    return err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource
```

```
    def delete_receipt_filter(self, filter_name):
        """
        Deletes a receipt filter.

        :param filter_name: The name of the filter to delete.
        """
        try:
            self.ses_client.delete_receipt_filter(FilterName=filter_name)
            logger.info("Deleted receipt filter %s.", filter_name)
        except ClientError:
            logger.exception("Couldn't delete receipt filter %s.", filter_name)
            raise
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

DeleteReceiptRule で を使用する AWS SDK

以下のコード例は、DeleteReceiptRule の使用方法を示しています。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
  \param receiptRuleName: The name for the receipt rule.
```

```
  \param receiptRuleSetName: The name for the receipt rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptRule(const Aws::String &receiptRuleName,
                                    const Aws::String &receiptRuleSetName,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleRequest deleteReceiptRuleRequest;

    deleteReceiptRuleRequest.SetRuleName(receiptRuleName);
    deleteReceiptRuleRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleOutcome outcome =
 sesClient.DeleteReceiptRule(
            deleteReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule." << std::endl;
    }
    else {
        std::cout << "Error deleting receipt rule. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { DeleteReceiptRuleCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const RULE_NAME = getUniqueName("RuleName");
const RULE_SET_NAME = getUniqueName("RuleSetName");

const createDeleteReceiptRuleCommand = () => {
  return new DeleteReceiptRuleCommand({
    RuleName: RULE_NAME,
    RuleSetName: RULE_SET_NAME,
  });
};

const run = async () => {
  const deleteReceiptRuleCommand = createDeleteReceiptRuleCommand();
  try {
    return await sesClient.send(deleteReceiptRuleCommand);
  } catch (err) {
    console.log("Failed to delete receipt rule.", err);
    return err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_rule(self, rule_set_name, rule_name):
        """
        Deletes a rule.

        :param rule_set_name: The rule set that contains the rule to delete.
        :param rule_name: The rule to delete.
        """
        try:
            self.ses_client.delete_receipt_rule(
                RuleSetName=rule_set_name, RuleName=rule_name
            )
            logger.info("Removed rule %s from rule set %s.", rule_name,
 rule_set_name)
        except ClientError:
            logger.exception(
                "Couldn't remove rule %s from rule set %s.", rule_name,
 rule_set_name
            )
            raise
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES でのAmazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DeleteReceiptRuleSet` で を使用する AWS SDK

以下のコード例は、`DeleteReceiptRuleSet` の使用方法を示しています。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
  \param receiptRuleSetName: The name for the receipt rule set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteReceiptRuleSet(const Aws::String &receiptRuleSetName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleSetRequest deleteReceiptRuleSetRequest;

    deleteReceiptRuleSetRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleSetOutcome outcome =
 sesClient.DeleteReceiptRuleSet(
            deleteReceiptRuleSetRequest);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule set." << std::endl;
    }

    else {
        std::cerr << "Error deleting receipt rule set. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { DeleteReceiptRuleSetCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const RULE_SET_NAME = getUniqueName("RuleSetName");

const createDeleteReceiptRuleSetCommand = () => {
  return new DeleteReceiptRuleSetCommand({ RuleSetName: RULE_SET_NAME });
};

const run = async () => {
  const deleteReceiptRuleSetCommand = createDeleteReceiptRuleSetCommand();
```

```
  try {
    return await sesClient.send(deleteReceiptRuleSetCommand);
  } catch (err) {
    console.log("Failed to delete receipt rule set.", err);
    return err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def delete_receipt_rule_set(self, rule_set_name):
        """
        Deletes a rule set. When a rule set is deleted, all of the rules it
 contains
        are also deleted.

        :param rule_set_name: The name of the rule set to delete.
```

```
        """
        try:
            self.ses_client.delete_receipt_rule_set(RuleSetName=rule_set_name)
            logger.info("Deleted rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't delete rule set %s.", rule_set_name)
            raise
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES でのAmazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DeleteTemplate` で を使用する AWS SDK

以下のコード例は、`DeleteTemplate` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- E メール ID を検証してメッセージを送信する

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Delete an email template.
```

```
    /// </summary>
    /// <param name="templateName">Name of the template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
    {
        var success = false;
        try
        {
            var response = await _amazonSimpleEmailService.DeleteTemplateAsync(
                new DeleteTemplateRequest
                {
                    TemplateName = templateName
                });
            success = response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Exception ex)
        {
            Console.WriteLine("DeleteEmailTemplateAsync failed with exception: "
+ ex.Message);
        }

        return success;
    }
```

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

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Delete an Amazon Simple Email Service (Amazon SES) template.
/*!
```

```
  \param templateName: The name for the template.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::deleteTemplate(const Aws::String &templateName,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteTemplateRequest deleteTemplateRequest;

    deleteTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::DeleteTemplateOutcome outcome = sesClient.DeleteTemplate(
            deleteTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted template." << std::endl;
    }
    else {
        std::cerr << "Error deleting template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { DeleteTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");

const createDeleteTemplateCommand = (templateName) =>
  new DeleteTemplateCommand({ TemplateName: templateName });

const run = async () => {
  const deleteTemplateCommand = createDeleteTemplateCommand(TEMPLATE_NAME);

  try {
    return await sesClient.send(deleteTemplateCommand);
  } catch (err) {
    console.log("Failed to delete template.", err);
    return err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
```

```
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def delete_template(self):
        """
        Deletes an email template.
        """
        try:

self.ses_client.delete_template(TemplateName=self.template["TemplateName"])
            logger.info("Deleted template %s.", self.template["TemplateName"])
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template["TemplateName"]
            )
            raise
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES でのAmazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`DescribeReceiptRuleSet` で を使用する AWS SDK

次のコード例は、`DescribeReceiptRuleSet` を使用する方法を示しています。

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def describe_receipt_rule_set(self, rule_set_name):
        """
        Gets data about a rule set.

        :param rule_set_name: The name of the rule set to retrieve.
        :return: Data about the rule set.
        """
        try:
            response = self.ses_client.describe_receipt_rule_set(
                RuleSetName=rule_set_name
            )
            logger.info("Got data for rule set %s.", rule_set_name)
        except ClientError:
            logger.exception("Couldn't get data for rule set %s.", rule_set_name)
            raise
        else:
            return response
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES でのAmazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **GetIdentityVerificationAttributes**で を使用する AWS SDK CLI

以下のコード例は、GetIdentityVerificationAttributes の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- E メール ID を検証してメッセージを送信する

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
/// <summary>
/// Get identity verification status for an email.
/// </summary>
/// <returns>The verification status of the email.</returns>
public async Task<VerificationStatus> GetIdentityStatusAsync(string email)
{
    var result = VerificationStatus.TemporaryFailure;
    try
    {
        var response =
```

```
                await
_amazonSimpleEmailService.GetIdentityVerificationAttributesAsync(
                    new GetIdentityVerificationAttributesRequest
                    {
                        Identities = new List<string> { email }
                    });

            if (response.VerificationAttributes.ContainsKey(email))
                result =
response.VerificationAttributes[email].VerificationStatus;
        }
        catch (Exception ex)
        {
            Console.WriteLine("GetIdentityStatusAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

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

CLI

AWS CLI

ID のリストの Amazon SES検証ステータスを取得するには

次の例では、 `get-identity-verification-attributes` コマンドを使用して、ID のリストの Amazon SES検証ステータスを取得します。

```
aws ses get-identity-verification-attributes --
identities "user1@example.com" "user2@example.com"
```

出力:

```
{
   "VerificationAttributes": {
       "user1@example.com": {
```

```
            "VerificationStatus": "Success"
        },
        "user2@example.com": {
            "VerificationStatus": "Pending"
        }
    }
}
```

検証のために、送信したことがない ID を使用してこのコマンドを呼び出した場合、その ID は出力に表示されません。

検証済み ID の詳細については、Amazon SES Simple Email Service デベロッパーガイドの「Amazon での E メールアドレスとドメインの検証」を参照してください。

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.
```

```
        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity]
            )
            status = response["VerificationAttributes"].get(
                identity, {"VerificationStatus": "NotFound"}
            )["VerificationStatus"]
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status
```

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

Ruby

SDK Ruby 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
```

```
ids = client.list_identities({
                                identity_type: 'EmailAddress'
                              })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **GetSendQuota**で を使用する AWS SDK CLI

以下のコード例は、GetSendQuota の使用方法を示しています。

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get information on the current account's send quota.
    /// </summary>
```

```
    /// <returns>The send quota response data.</returns>
    public async Task<GetSendQuotaResponse> GetSendQuotaAsync()
    {
        var result = new GetSendQuotaResponse();
        try
        {
            var response = await _amazonSimpleEmailService.GetSendQuotaAsync(
                new GetSendQuotaRequest());
            result = response;
        }
        catch (Exception ex)
        {
            Console.WriteLine("GetSendQuotaAsync failed with exception: " +
 ex.Message);
        }

        return result;
    }
```

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

CLI

AWS CLI

Amazon SESの送信制限を取得するには

次の例では、 `get-send-quota` コマンドを使用して Amazon SESの送信制限を返します。

```
aws ses get-send-quota
```

出力:

```
{
   "Max24HourSend": 200.0,
   "SentLast24Hours": 1.0,
   "MaxSendRate": 1.0
}
```

Max24HourSend は送信クォータで、24 時間に送信できる E メールの最大数です。送信クォータには、期間の推移が反映されます。E メールを送信しようとするたびに、Amazon は過去 24 時間に送信された E メールの数SESを確認します。送信済みのメールの合計数がクォータ未満であれば、送信リクエストは受理され、E メールが送信されます。

SentLast24Hours、過去 24 時間以内に送信した E メールの数です。

MaxSendRate は、1 秒あたりに送信できる E メールの最大数です。

送信制限は、メッセージ数ではなく、受取人数に基づいていることに注意してください。例えば、受取人数が 10 人である E メールは、送信クォータに対しては 10 通とカウントされます。

詳細については、「Amazon Simple Email Service デベロッパーガイド」の「Amazon SES送信制限の管理」を参照してください。

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

PowerShell

のツール PowerShell

例 1: このコマンドは、ユーザーの現在の送信制限を返します。

```
Get-SESSendQuota
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`GetSendStatistics` で を使用する CLI

以下のコード例は、`GetSendStatistics` の使用方法を示しています。

CLI

AWS CLI

Amazon SES送信統計を取得するには

次の例では、 `get-send-statistics` コマンドを使用して Amazon SES送信統計を返します。

```
aws ses get-send-statistics
```

出力:

```
{
   "SendDataPoints": [
       {
           "Complaints": 0,
           "Timestamp": "2013-06-12T19:32:00Z",
           "DeliveryAttempts": 2,
           "Bounces": 0,
           "Rejects": 0
       },
       {
           "Complaints": 0,
           "Timestamp": "2013-06-12T00:47:00Z",
           "DeliveryAttempts": 1,
           "Bounces": 0,
           "Rejects": 0
       }
   ]
}
```

結果は、送信アクティビティの最新の 2 週間を示すデータポイントのリストです。このリスト内の各データポイントには、15 分間隔の統計が含まれます。

この例では、過去 2 週間にユーザーが送信した E メールが 15 分間隔で 2 件のみだったため、データポイントは 2 つしかありません。

詳細については、「Amazon Simple Email Service デベロッパーガイド」の「Amazon SES使用状況統計のモニタリング」を参照してください。

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

PowerShell

のツール PowerShell

例 1: このコマンドは、ユーザーの統計の送信を返します。結果は、送信アクティビティの最新の 2 週間を示すデータポイントのリストです。このリスト内の各データポイントには、15 分間隔の統計が含まれます。

```
Get-SESSendStatistic
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

GetTemplate で を使用する AWS SDK

以下のコード例は、GetTemplate の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- E メール ID を検証してメッセージを送信する

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Get a template's attributes.
/*!
  \param templateName: The name for the template.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::getTemplate(const Aws::String &templateName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::GetTemplateRequest getTemplateRequest;

    getTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::GetTemplateOutcome outcome = sesClient.GetTemplate(
            getTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully got template." << std::endl;
    }

    else {
        std::cerr << "Error getting template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { GetTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");

const createGetTemplateCommand = (templateName) =>
  new GetTemplateCommand({ TemplateName: templateName });

const run = async () => {
  const getTemplateCommand = createGetTemplateCommand(TEMPLATE_NAME);

  try {
    return await sesClient.send(getTemplateCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""
```

```
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def get_template(self, name):
        """
        Gets a previously created email template.

        :param name: The name of the template to retrieve.
        :return: The retrieved email template.
        """
        try:
            response = self.ses_client.get_template(TemplateName=name)
            self.template = response["Template"]
            logger.info("Got template %s.", name)
            self._extract_tags(
                self.template["SubjectPart"],
                self.template["TextPart"],
                self.template["HtmlPart"],
            )
        except ClientError:
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **ListIdentities**AWS SDKで を使用する CLI

以下のコード例は、ListIdentities の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- リージョン間で E メールとドメイン ID をコピーする
- E メール ID を検証してメッセージを送信する

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// Get the identities of a specified type for the current account.
    /// </summary>
    /// <param name="identityType">IdentityType to list.</param>
    /// <returns>The list of identities.</returns>
    public async Task<List<string>> ListIdentitiesAsync(IdentityType
identityType)
    {
        var result = new List<string>();
        try
```

```
        {
            var response = await _amazonSimpleEmailService.ListIdentitiesAsync(
                new ListIdentitiesRequest
                {
                    IdentityType = identityType
                });
            result = response.Identities;
        }
        catch (Exception ex)
        {
            Console.WriteLine("ListIdentitiesAsync failed with exception: " +
 ex.Message);
        }

        return result;
    }
```

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

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! List the identities associated with this account.
/*!
  \param identityType: The identity type enum. "NOT_SET" is a valid option.
  \param identities; A vector to receive the retrieved identities.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::listIdentities(Aws::SES::Model::IdentityType identityType,
                                 Aws::Vector<Aws::String> &identities,
```

```
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::ListIdentitiesRequest listIdentitiesRequest;

    if (identityType != Aws::SES::Model::IdentityType::NOT_SET) {
        listIdentitiesRequest.SetIdentityType(identityType);
    }

    Aws::String nextToken; // Used for paginated results.
    do {
        if (!nextToken.empty()) {
            listIdentitiesRequest.SetNextToken(nextToken);
        }
        Aws::SES::Model::ListIdentitiesOutcome outcome =
 sesClient.ListIdentities(
                listIdentitiesRequest);

        if (outcome.IsSuccess()) {
            const auto &retrievedIdentities =
 outcome.GetResult().GetIdentities();
            if (!retrievedIdentities.empty()) {
                identities.insert(identities.cend(),
 retrievedIdentities.cbegin(),
                                  retrievedIdentities.cend());
            }
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error listing identities. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    return true;
}
```

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

CLI

AWS CLI

特定の AWS アカウントのすべての ID (E メールアドレスとドメイン) を一覧表示するには

次の例では、 `list-identities` コマンドを使用して、Amazon での検証のために送信されたすべての ID を一覧表示しますSES。

```
aws ses list-identities
```

出力:

```
{
    "Identities": [
      "user@example.com",
      "example.com"
    ]
}
```

返されるリストには、検証ステータス (検証済み、検証保留中、失敗など) に関係なく、すべての ID が含まれます。

この例では、identity-type パラメータを指定しなかったため、E メールアドレスおよびドメインが返されます。

検証の詳細については、Amazon Simple Email Service デベロッパーガイドのSES「Amazon での E メールアドレスとドメインの検証」を参照してください。

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

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        listSESIdentities(client);
    }

    public static void listSESIdentities(SesClient client) {
        try {
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
            List<String> identities = identitiesResponse.identities();
            for (String identity : identities) {
```

```
                System.out.println("The identity is " + identity);
            }

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

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { ListIdentitiesCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListIdentitiesCommand = () =>
  new ListIdentitiesCommand({ IdentityType: "EmailAddress", MaxItems: 10 });

const run = async () => {
  const listIdentitiesCommand = createListIdentitiesCommand();

  try {
    return await sesClient.send(listIdentitiesCommand);
  } catch (err) {
    console.log("Failed to list identities.", err);
    return err;
  }
};
```

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

PowerShell

のツール PowerShell

例 1: このコマンドは、検証ステータスに関係なく、特定の AWS アカウントのすべての ID (E メールアドレスとドメイン) を含むリストを返します。

```
Get-SESIdentity
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.
```

```
        :param identity_type: The type of identity to retrieve, such as
 EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items
            )
            identities = response["Identities"]
            logger.info("Got %s identities for the current account.",
 len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
            raise
        else:
            return identities
```

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

Ruby

SDK Ruby 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')
```

```
# Get up to 1000 identities
ids = client.list_identities({
                               identity_type: 'EmailAddress'
                             })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`ListReceiptFilters` で を使用する AWS SDK

以下のコード例は、`ListReceiptFilters` の使用方法を示しています。

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! List the receipt filters associated with this account.
/*!
  \param filters; A vector of "ReceiptFilter" to receive the retrieved filters.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SES::listReceiptFilters(Aws::Vector<Aws::SES::Model::ReceiptFilter>
 &filters,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::ListReceiptFiltersRequest listReceiptFiltersRequest;

    Aws::SES::Model::ListReceiptFiltersOutcome outcome =
 sesClient.ListReceiptFilters(
            listReceiptFiltersRequest);
    if (outcome.IsSuccess()) {
        auto &retrievedFilters = outcome.GetResult().GetFilters();
        if (!retrievedFilters.empty()) {
            filters.insert(filters.cend(), retrievedFilters.cbegin(),
                           retrievedFilters.cend());
        }
    }
    else {
        std::cerr << "Error retrieving IP address filters: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { ListReceiptFiltersCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListReceiptFiltersCommand = () => new ListReceiptFiltersCommand({});

const run = async () => {
  const listReceiptFiltersCommand = createListReceiptFiltersCommand();

  return await sesClient.send(listReceiptFiltersCommand);
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""

    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource


    def list_receipt_filters(self):
        """
        Gets the list of receipt filters for the current account.
```

```
        :return: The list of receipt filters.
        """
        try:
            response = self.ses_client.list_receipt_filters()
            filters = response["Filters"]
            logger.info("Got %s receipt filters.", len(filters))
        except ClientError:
            logger.exception("Couldn't get receipt filters.")
            raise
        else:
            return filters
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`ListTemplates` で を使用する AWS SDK

以下のコード例は、`ListTemplates` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- E メール ID を検証してメッセージを送信する

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    /// <summary>
    /// List email templates for the current account.
    /// </summary>
    /// <returns>A list of template metadata.</returns>
    public async Task<List<TemplateMetadata>> ListEmailTemplatesAsync()
    {
        var result = new List<TemplateMetadata>();
        try
        {
            var response = await _amazonSimpleEmailService.ListTemplatesAsync(
                new ListTemplatesRequest());
            result = response.TemplatesMetadata;
        }
        catch (Exception ex)
        {
            Console.WriteLine("ListEmailTemplatesAsync failed with exception: " +
ex.Message);
        }

        return result;
    }
```

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

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesRequest;
```

```
import software.amazon.awssdk.services.sesv2.model.ListEmailTemplatesResponse;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;

public class ListTemplates {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        listAllTemplates(sesv2Client);
    }

    public static void listAllTemplates(SesV2Client sesv2Client) {
        try {
            ListEmailTemplatesRequest templatesRequest =
 ListEmailTemplatesRequest.builder()
                    .pageSize(1)
                    .build();

            ListEmailTemplatesResponse response =
 sesv2Client.listEmailTemplates(templatesRequest);
            response.templatesMetadata()
                    .forEach(template -> System.out.println("Template name: " +
 template.templateName()));

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { ListTemplatesCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListTemplatesCommand = (maxItems) =>
  new ListTemplatesCommand({ MaxItems: maxItems });

const run = async () => {
  const listTemplatesCommand = createListTemplatesCommand(10);

  try {
    return await sesClient.send(listTemplatesCommand);
  } catch (err) {
    console.log("Failed to list templates.", err);
    return err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
 html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
            response = self.ses_client.list_templates()
            templates = response["TemplatesMetadata"]
            logger.info("Got %s templates.", len(templates))
        except ClientError:
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES でのAmazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

SendBulkTemplatedEmail で を使用する AWS SDK

次の例は、SendBulkTemplatedEmail を使用する方法を説明しています。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { SendBulkTemplatedEmailCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * Replace this with the name of an existing template.
 */
const TEMPLATE_NAME = getUniqueName("ReminderTemplate");

/**
 * Replace these with existing verified emails.
 */
const VERIFIED_EMAIL_1 = postfix(getUniqueName("Bilbo"), "@example.com");
const VERIFIED_EMAIL_2 = postfix(getUniqueName("Frodo"), "@example.com");

const USERS = [
  { firstName: "Bilbo", emailAddress: VERIFIED_EMAIL_1 },
```

```
  { firstName: "Frodo", emailAddress: VERIFIED_EMAIL_2 },
];

/**
 *
 * @param { { emailAddress: string, firstName: string }[] } users
 * @param { string } templateName the name of an existing template in SES
 * @returns { SendBulkTemplatedEmailCommand }
 */
const createBulkReminderEmailCommand = (users, templateName) => {
  return new SendBulkTemplatedEmailCommand({
    /**
     * Each 'Destination' uses a corresponding set of replacement data. We can
 map each user
     * to a 'Destination' and provide user specific replacement data to create
 personalized emails.
     *
     * Here's an example of how a template would be replaced with user data:
     * Template: <h1>Hello {{name}},</h1><p>Don't forget about the party gifts!</
p>
     * Destination 1: <h1>Hello Bilbo,</h1><p>Don't forget about the party gifts!
</p>
     * Destination 2: <h1>Hello Frodo,</h1><p>Don't forget about the party gifts!
</p>
     */
    Destinations: users.map((user) => ({
      Destination: { ToAddresses: [user.emailAddress] },
      ReplacementTemplateData: JSON.stringify({ name: user.firstName }),
    })),
    DefaultTemplateData: JSON.stringify({ name: "Shireling" }),
    Source: VERIFIED_EMAIL_1,
    Template: templateName,
  });
};

const run = async () => {
  const sendBulkTemplateEmailCommand = createBulkReminderEmailCommand(
    USERS,
    TEMPLATE_NAME,
  );
  try {
    return await sesClient.send(sendBulkTemplateEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
```

```
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **SendEmail**で を使用する AWS SDK CLI

以下のコード例は、SendEmail の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- E メール ID を検証してメッセージを送信する

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
/// <summary>
///  Send an email by using Amazon SES.
/// </summary>
/// <param name="toAddresses">List of recipients.</param>
```

```
    /// <param name="ccAddresses">List of cc recipients.</param>
    /// <param name="bccAddresses">List of bcc recipients.</param>
    /// <param name="bodyHtml">Body of the email in HTML.</param>
    /// <param name="bodyText">Body of the email in plain text.</param>
    /// <param name="subject">Subject line of the email.</param>
    /// <param name="senderAddress">From address.</param>
    /// <returns>The messageId of the email.</returns>
    public async Task<string> SendEmailAsync(List<string> toAddresses,
        List<string> ccAddresses, List<string> bccAddresses,
        string bodyHtml, string bodyText, string subject, string senderAddress)
    {
        var messageId = "";
        try
        {
            var response = await _amazonSimpleEmailService.SendEmailAsync(
                new SendEmailRequest
                {
                    Destination = new Destination
                    {
                        BccAddresses = bccAddresses,
                        CcAddresses = ccAddresses,
                        ToAddresses = toAddresses
                    },
                    Message = new Message
                    {
                        Body = new Body
                        {
                            Html = new Content
                            {
                                Charset = "UTF-8",
                                Data = bodyHtml
                            },
                            Text = new Content
                            {
                                Charset = "UTF-8",
                                Data = bodyText
                            }
                        },
                        Subject = new Content
                        {
                            Charset = "UTF-8",
                            Data = subject
                        }
                    },
```

```
                    Source = senderAddress
                });
            messageId = response.MessageId;
        }
        catch (Exception ex)
        {
            Console.WriteLine("SendEmailAsync failed with exception: " +
 ex.Message);
        }

        return messageId;
    }
```

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

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Send an email to a list of recipients.
/*!
  \param recipients; Vector of recipient email addresses.
  \param subject: Email subject.
  \param htmlBody: Email body as HTML. At least one body data is required.
  \param textBody: Email body as plain text. At least one body data is required.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::sendEmail(const Aws::Vector<Aws::String> &recipients,
```

```
                            const Aws::String &subject,
                            const Aws::String &htmlBody,
                            const Aws::String &textBody,
                            const Aws::String &senderEmailAddress,
                            const Aws::Vector<Aws::String> &ccAddresses,
                            const Aws::String &replyToAddress,
                            const Aws::Client::ClientConfiguration
&clientConfiguration) {
   Aws::SES::SESClient sesClient(clientConfiguration);

   Aws::SES::Model::Destination destination;
   if (!ccAddresses.empty()) {
       destination.WithCcAddresses(ccAddresses);
   }
   if (!recipients.empty()) {
       destination.WithToAddresses(recipients);
   }

   Aws::SES::Model::Body message_body;
   if (!htmlBody.empty()) {
       message_body.SetHtml(

Aws::SES::Model::Content().WithCharset("UTF-8").WithData(htmlBody));
   }

   if (!textBody.empty()) {
       message_body.SetText(

Aws::SES::Model::Content().WithCharset("UTF-8").WithData(textBody));
   }

   Aws::SES::Model::Message message;
   message.SetBody(message_body);
   message.SetSubject(
           Aws::SES::Model::Content().WithCharset("UTF-8").WithData(subject));

   Aws::SES::Model::SendEmailRequest sendEmailRequest;
   sendEmailRequest.SetDestination(destination);
   sendEmailRequest.SetMessage(message);
   if (!senderEmailAddress.empty()) {
       sendEmailRequest.SetSource(senderEmailAddress);
   }
   if (!replyToAddress.empty()) {
       sendEmailRequest.AddReplyToAddresses(replyToAddress);
```

```
    }

    auto outcome = sesClient.SendEmail(sendEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending message. " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

CLI

AWS CLI

Amazon を使用してフォーマットされた E メールを送信するには SES

以下の例では、send-email コマンドを使用してフォーマットされた E メールを送信しています。

```
aws ses send-email --from sender@example.com --destination file://
destination.json --message file://message.json
```

出力:

```
{
   "MessageId": "EXAMPLEf3a5efcd1-51adec81-d2a4-4e3f-9fe2-5d85c1b23783-000000"
}
```

送信先とメッセージは、現在のディレクトリの .json ファイルに保存されるJSONデータ構造です。これらのファイルは以下のとおりです。

destination.json:

```
{
  "ToAddresses":  ["recipient1@example.com", "recipient2@example.com"],
  "CcAddresses":  ["recipient3@example.com"],
  "BccAddresses": []
}
```

message.json:

```
{
   "Subject": {
       "Data": "Test email sent using the AWS CLI",
       "Charset": "UTF-8"
   },
   "Body": {
       "Text": {
           "Data": "This is the message body in text format.",
           "Charset": "UTF-8"
       },
       "Html": {
           "Data": "This message body contains HTML formatting. It can, for
 example, contain links like this one: <a class=\"ulink\" href=\"http://
docs.aws.amazon.com/ses/latest/DeveloperGuide\" target=\"_blank\">Amazon SES
 Developer Guide</a>.",
           "Charset": "UTF-8"
       }
   }
}
```

送信者と受信者の E メールアドレスを、使用したい E メールアドレスに置き換えます。送信者の E メールアドレスは Amazon で検証する必要があることに注意してくださいSES。Amazon への本稼働アクセスが許可されるまでSES、受信者が Amazon SESメールボックスシミュレーターでない限り、各受信者の E メールアドレスも検証する必要があります。検証の詳細については、Amazon Simple Email Service デベロッパーガイドの SES「Amazon での E メールアドレスとドメインの検証」を参照してください。

出力のメッセージ ID は、send-email の呼び出しが成功したことを示しています。

E メールが届かない場合は、迷惑メールフォルダを確認してください。

フォーマットされた E メールの送信の詳細については、Amazon Simple Email Service デベロッパーガイドのSESAPI「Amazon を使用したフォーマットされた E メールの送信」を参照してください。

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

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

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

                Usage:
```

```
                    <sender> <recipient> <subject>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.
\s
                    subject - The  subject line.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</
h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        try {
            send(client, sender, recipient, subject, bodyHTML);
            client.close();
            System.out.println("Done");

        } catch (MessagingException e) {
            e.getStackTrace();
        }
    }

    public static void send(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyHTML) throws MessagingException {
```

```
        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();

        Content sub = Content.builder()
                .data(subject)
                .build();

        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .message(msg)
                .source(sender)
                .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES "
 + "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);

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

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
```

```
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject> <fileLocation>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.
\s
                    subject - The  subject line.\s
                    fileLocation - The location of a Microsoft Excel file to use
 as an attachment (C:/AWS/customers.xls).\s
                """;
```

```
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        String fileLocation = args[3];

        // The email body for recipients with non-HTML email clients.
        String bodyText = "Hello,\r\n" + "Please see the attached file for a list
 "
                + "of customers to contact.";

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</
h1>"
                + "<p>Please see the attached file for a " + "list of customers
 to contact.</p>" + "</body>"
                + "</html>";

        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        try {
            sendemailAttachment(client, sender, recipient, subject, bodyText,
 bodyHTML, fileLocation);
            client.close();
            System.out.println("Done");

        } catch (IOException | MessagingException e) {
            e.getStackTrace();
        }
    }

    public static void sendemailAttachment(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyText,
            String bodyHTML,
```

```
            String fileLocation) throws AddressException, MessagingException,
IOException {

        java.io.File theFile = new java.io.File(fileLocation);
        byte[] fileContent = Files.readAllBytes(theFile.toPath());

        Session session = Session.getDefaultInstance(new Properties());

        // Create a new MimeMessage object.
        MimeMessage message = new MimeMessage(session);

        // Add subject, from and to lines.
        message.setSubject(subject, "UTF-8");
        message.setFrom(new InternetAddress(sender));
        message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipient));

        // Create a multipart/alternative child container.
        MimeMultipart msgBody = new MimeMultipart("alternative");

        // Create a wrapper for the HTML and text parts.
        MimeBodyPart wrap = new MimeBodyPart();

        // Define the text part.
        MimeBodyPart textPart = new MimeBodyPart();
        textPart.setContent(bodyText, "text/plain; charset=UTF-8");

        // Define the HTML part.
        MimeBodyPart htmlPart = new MimeBodyPart();
        htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

        // Add the text and HTML parts to the child container.
        msgBody.addBodyPart(textPart);
        msgBody.addBodyPart(htmlPart);

        // Add the child container to the wrapper object.
        wrap.setContent(msgBody);

        // Create a multipart/mixed parent container.
        MimeMultipart msg = new MimeMultipart("mixed");

        // Add the parent container to the message.
        message.setContent(msg);
        msg.addBodyPart(wrap);
```

```
        // Define the attachment.
        MimeBodyPart att = new MimeBodyPart();
        DataSource fds = new ByteArrayDataSource(fileContent,
                "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
        att.setDataHandler(new DataHandler(fds));

        String reportName = "WorkReport.xls";
        att.setFileName(reportName);

        // Add the attachment to the message.
        msg.addBodyPart(att);

        try {
            System.out.println("Attempting to send an email through Amazon SES "
 + "using the AWS SDK for Java...");

            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            message.writeTo(outputStream);

            ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

            byte[] arr = new byte[buf.remaining()];
            buf.get(arr);

            SdkBytes data = SdkBytes.fromByteArray(arr);
            RawMessage rawMessage = RawMessage.builder()
                    .data(data)
                    .build();

            SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
                    .rawMessage(rawMessage)
                    .build();

            client.sendRawEmail(rawEmailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Email sent using SesClient with attachment");
    }
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { SendEmailCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createSendEmailCommand = (toAddress, fromAddress) => {
  return new SendEmailCommand({
    Destination: {
      /* required */
      CcAddresses: [
        /* more items */
      ],
      ToAddresses: [
        toAddress,
        /* more To-email addresses */
      ],
    },
    Message: {
      /* required */
      Body: {
        /* required */
        Html: {
          Charset: "UTF-8",
          Data: "HTML_FORMAT_BODY",
        },
        Text: {
          Charset: "UTF-8",
          Data: "TEXT_FORMAT_BODY",
```

```
        },
      },
      Subject: {
        Charset: "UTF-8",
        Data: "EMAIL_SUBJECT",
      },
    },
    Source: fromAddress,
    ReplyToAddresses: [
      /* more items */
    ],
  });
};

const run = async () => {
  const sendEmailCommand = createSendEmailCommand(
    "recipient@example.com",
    "sender@example.com",
  );

  try {
    return await sesClient.send(sendEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_email(self, source, destination, subject, text, html,
 reply_tos=None):
        """
        Sends an email.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param subject: The subject of the email.
        :param text: The plain text version of the body of the email.
        :param html: The HTML version of the body of the email.
        :param reply_tos: Email accounts that will receive a reply if the
 recipient
                          replies to the message.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Message": {
```

```
                "Subject": {"Data": subject},
                "Body": {"Text": {"Data": text}, "Html": {"Data": html}},
            },
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent mail %s from %s to %s.", message_id, source,
destination.tos
            )
        except ClientError:
            logger.exception(
                "Couldn't send mail from %s to %s.", source, destination.tos
            )
            raise
        else:
            return message_id
```

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

Ruby

SDK Ruby 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
```

```
sender = 'sender@example.com'

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = 'recipient@example.com'

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
#   configsetname = "ConfigSet"

# The subject line for the email.
subject = 'Amazon SES test (AWS SDK for Ruby)'

# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (AWS SDK for Ruby)</h1>'\
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  'AWS SDK for Ruby</a>.'

# The email body for recipients with non-HTML email clients.
textbody = 'This email was sent with Amazon SES using the AWS SDK for Ruby.'

# Specify the text encoding scheme.
encoding = 'UTF-8'

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
```

```
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
    source: sender
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES でのAmazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **SendRawEmail**AWS SDKで を使用する CLI

以下のコード例は、SendRawEmail の使用方法を示しています。

CLI

AWS CLI

Amazon を使用して raw E メールを送信するには SES

次の例では、 send-raw-email コマンドを使用して、TXT添付ファイルを含む E メールを送信します。

```
aws ses send-raw-email --raw-message file://message.json
```

出力:

```
{
    "MessageId": "EXAMPLEf3f73d99b-c63fb06f-d263-41f8-a0fb-d0dc67d56c07-000000"
}
```

raw メッセージは、現在のディレクトリの という名前message.jsonのファイルに保存されたJSONデータ構造です。以下の要素が含まれます。

```
{
    "Data": "From: sender@example.com\nTo: recipient@example.com\nSubject:
 Test email sent using the AWS CLI (contains an attachment)\nMIME-Version:
 1.0\nContent-type: Multipart/Mixed; boundary=\"NextPart\"\n\n--NextPart
\nContent-Type: text/plain\n\nThis is the message body.\n\n--NextPart\nContent-
Type: text/plain;\nContent-Disposition: attachment; filename=\"attachment.txt\"\n
\nThis is the text in the attachment.\n\n--NextPart--"
}
```

ご覧のとおり、「Data」は、添付ファイル attachment.txt など、raw E メールのコンテンツ全体を MIME形式で含む 1 つの長い文字列です。

sender@example.com と recipient@example.com は、使用するアドレスに置き換えてください。送信者の E メールアドレスは Amazon で検証する必要があることに注意してくださいSES。Amazon への本稼働アクセスが許可されるまでSES、受信者が Amazon SES メールボックスシミュレーターでない限り、受信者の E メールアドレスも検証する必要があります。検証の詳細については、Amazon Simple Email Service デベロッパーガイドの SES「Amazon での E メールアドレスとドメインの検証」を参照してください。

出力のメッセージ ID は、 の send-raw-email呼び出しが成功したことを示します。

E メールが届かない場合は、迷惑メールフォルダを確認してください。

raw E メールの送信の詳細については、Amazon Simple Email Service デベロッパーガイドの SESAPI「Amazon を使用した raw E メールの送信」を参照してください。

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

nodemailer を使用して、添付ファイル付きの E メールを送信します。

```
import sesClientModule from "@aws-sdk/client-ses";
/**
 * nodemailer wraps the SES SDK and calls SendRawEmail. Use this for more
 advanced
 * functionality like adding attachments to your email.
 *
 * https://nodemailer.com/transports/ses/
 */
import nodemailer from "nodemailer";

/**
 * @param {string} from An Amazon SES verified email address.
 * @param {*} to An Amazon SES verified email address.
 */
export const sendEmailWithAttachments = (
  from = "from@example.com",
  to = "to@example.com",
) => {
  const ses = new sesClientModule.SESClient({});
  const transporter = nodemailer.createTransport({
    SES: { ses, aws: sesClientModule },
  });

  return new Promise((resolve, reject) => {
    transporter.sendMail(
      {
        from,
        to,
        subject: "Hello World",
        text: "Greetings from Amazon SES!",
        attachments: [{ content: "Hello World!", filename: "hello.txt" }],
```

```
      },
      (err, info) => {
        if (err) {
          reject(err);
        } else {
          resolve(info);
        }
      },
    );
  });
};
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

SendTemplatedEmail で を使用する AWS SDK

以下のコード例は、SendTemplatedEmail の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- E メール ID を検証してメッセージを送信する

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   /// <summary>
   /// Send an email using a template.
   /// </summary>
   /// <param name="sender">Address of the sender.</param>
   /// <param name="recipients">Addresses of the recipients.</param>
   /// <param name="templateName">Name of the email template.</param>
   /// <param name="templateDataObject">Data for the email template.</param>
   /// <returns>The messageId of the email.</returns>
   public async Task<string> SendTemplateEmailAsync(string sender, List<string>
recipients,
       string templateName, object templateDataObject)
   {
       var messageId = "";
       try
       {
           // Template data should be serialized JSON from either a class or a
dynamic object.
           var templateData = JsonSerializer.Serialize(templateDataObject);

           var response = await
_amazonSimpleEmailService.SendTemplatedEmailAsync(
               new SendTemplatedEmailRequest
               {
                   Source = sender,
                   Destination = new Destination
                   {
                       ToAddresses = recipients
                   },
                   Template = templateName,
                   TemplateData = templateData
               });
           messageId = response.MessageId;
       }
       catch (Exception ex)
       {
           Console.WriteLine("SendTemplateEmailAsync failed with exception: " +
ex.Message);
       }

       return messageId;
   }
```

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

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Send a templated email to a list of recipients.
/*!
  \param recipients; Vector of recipient email addresses.
  \param templateName: The name of the template to use.
  \param templateData: Map of key-value pairs for replacing text in template.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::sendTemplatedEmail(const Aws::Vector<Aws::String> &recipients,
                                     const Aws::String &templateName,
                                     const Aws::Map<Aws::String, Aws::String>
 &templateData,
                                     const Aws::String &senderEmailAddress,
                                     const Aws::Vector<Aws::String> &ccAddresses,
                                     const Aws::String &replyToAddress,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
    }
    if (!recipients.empty()) {
        destination.WithToAddresses(recipients);
```

```
    }

    Aws::SES::Model::SendTemplatedEmailRequest sendTemplatedEmailRequest;
    sendTemplatedEmailRequest.SetDestination(destination);
    sendTemplatedEmailRequest.SetTemplate(templateName);

    std::ostringstream templateDataStream;
    templateDataStream << "{";
    size_t dataCount = 0;
    for (auto &pair: templateData) {
        templateDataStream << "\"" << pair.first << "\":\"" << pair.second <<
 "\"";
        dataCount++;
        if (dataCount < templateData.size()) {
            templateDataStream << ",";
        }
    }
    templateDataStream << "}";

    sendTemplatedEmailRequest.SetTemplateData(templateDataStream.str());

    if (!senderEmailAddress.empty()) {
        sendTemplatedEmailRequest.SetSource(senderEmailAddress);
    }
    if (!replyToAddress.empty()) {
        sendTemplatedEmailRequest.AddReplyToAddresses(replyToAddress);
    }

    auto outcome = sesClient.SendTemplatedEmail(sendTemplatedEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent templated message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending templated message. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

Java

SDK for Java 2.x

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;
import software.amazon.awssdk.services.sesv2.model.Template;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * Also, make sure that you create a template. See the following documentation
 * topic:
 *
 * https://docs.aws.amazon.com/ses/latest/dg/send-personalized-email-api.html
 */

public class SendEmailTemplate {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <template> <sender> <recipient>\s

                Where:
                    template - The name of the email template.
                    sender - An email address that represents the sender.\s
                    recipient - An email address that represents the recipient.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String templateName = args[0];
        String sender = args[1];
        String recipient = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        send(sesv2Client, sender, recipient, templateName);
    }

    public static void send(SesV2Client client, String sender, String recipient,
String templateName) {
        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        /*
         * Specify both name and favorite animal (favoriteanimal) in your code
when
         * defining the Template object.
         * If you don't specify all the variables in the template, Amazon SES
doesn't
         * send the email.
         */
        Template myTemplate = Template.builder()
                .templateName(templateName)
                .templateData("{\n" +
                        "  \"name\": \"Jason\"\n," +
```

```
                        "  \"favoriteanimal\": \"Cat\"\n" +
                        "}")
                .build();

        EmailContent emailContent = EmailContent.builder()
                .template(myTemplate)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

        try {
            System.out.println("Attempting to send an email based on a template
 using the AWS SDK for Java (v2)...");
            client.sendEmail(emailRequest);
            System.out.println("email based on a template was sent");

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { SendTemplatedEmailCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * Replace this with the name of an existing template.
 */
const TEMPLATE_NAME = getUniqueName("ReminderTemplate");

/**
 * Replace these with existing verified emails.
 */
const VERIFIED_EMAIL = postfix(getUniqueName("Bilbo"), "@example.com");

const USER = { firstName: "Bilbo", emailAddress: VERIFIED_EMAIL };

/**
 *
 * @param { { emailAddress: string, firstName: string } } user
 * @param { string } templateName - The name of an existing template in Amazon
 SES.
 * @returns { SendTemplatedEmailCommand }
 */
const createReminderEmailCommand = (user, templateName) => {
  return new SendTemplatedEmailCommand({
    /**
     * Here's an example of how a template would be replaced with user data:
     * Template: <h1>Hello {{contact.firstName}},</h1><p>Don't forget about the
 party gifts!</p>
     * Destination: <h1>Hello Bilbo,</h1><p>Don't forget about the party gifts!</
p>
     */
    Destination: { ToAddresses: [user.emailAddress] },
    TemplateData: JSON.stringify({ contact: { firstName: user.firstName } }),
    Source: VERIFIED_EMAIL,
    Template: templateName,
  });
};

const run = async () => {
```

```
  const sendReminderEmailCommand = createReminderEmailCommand(
    USER,
    TEMPLATE_NAME,
  );
  try {
    return await sesClient.send(sendReminderEmailCommand);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MessageRejected") {
      /** @type { import('@aws-sdk/client-ses').MessageRejected} */
      const messageRejectedError = caught;
      return messageRejectedError;
    }
    throw caught;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_templated_email(
```

```
        self, source, destination, template_name, template_data, reply_tos=None
    ):
        """
        Sends an email based on a template. A template contains replaceable tags
        each enclosed in two curly braces, such as {{name}}. The template data
passed
        in this function contains key-value pairs that define the values to
insert
        in place of the template tags.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement
values
                              that are inserted in the template before it is
sent.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Template": template_name,
            "TemplateData": json.dumps(template_data),
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_templated_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent templated mail %s from %s to %s.",
                message_id,
                source,
                destination.tos,
            )
        except ClientError:
            logger.exception(
                "Couldn't send templated mail from %s to %s.", source,
destination.tos
            )
```

```
            raise
        else:
            return message_id
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

`UpdateTemplate` で を使用する AWS SDK

以下のコード例は、`UpdateTemplate` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- E メール ID を検証してメッセージを送信する

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Update an Amazon Simple Email Service (Amazon SES) template.
/*!
  \param templateName: The name of the template.
  \param htmlPart: The HTML body of the email.
  \param subjectPart: The subject line of the email.
  \param textPart: The plain text version of the email.
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::updateTemplate(const Aws::String &templateName,
                                 const Aws::String &htmlPart,
                                 const Aws::String &subjectPart,
                                 const Aws::String &textPart,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Template templateValues;

    templateValues.SetTemplateName(templateName);
    templateValues.SetSubjectPart(subjectPart);
    templateValues.SetHtmlPart(htmlPart);
    templateValues.SetTextPart(textPart);

    Aws::SES::Model::UpdateTemplateRequest updateTemplateRequest;
    updateTemplateRequest.SetTemplate(templateValues);

    Aws::SES::Model::UpdateTemplateOutcome outcome =
 sesClient.UpdateTemplate(updateTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated template." << std::endl;
    } else {
        std::cerr << "Error updating template. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { UpdateTemplateCommand } from "@aws-sdk/client-ses";
import { getUniqueName } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

const TEMPLATE_NAME = getUniqueName("TemplateName");
const HTML_PART = "<h1>Hello, World!</h1>";

const createUpdateTemplateCommand = () => {
  return new UpdateTemplateCommand({
    Template: {
      TemplateName: TEMPLATE_NAME,
      HtmlPart: HTML_PART,
      SubjectPart: "Example",
      TextPart: "Updated template text.",
    },
  });
};

const run = async () => {
  const updateTemplateCommand = createUpdateTemplateCommand();

  try {
    return await sesClient.send(updateTemplateCommand);
  } catch (err) {
    console.log("Failed to update template.", err);
    return err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
 html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
```

```
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't update template %s.", name)
            raise
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **VerifyDomainIdentity**で を使用する AWS SDK CLI

以下のコード例は、VerifyDomainIdentity の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- リージョン間で E メールとドメイン ID をコピーする
- E メール ID を検証してメッセージを送信する

CLI

AWS CLI

Amazon でドメインを検証するには SES

以下の例では、verify-domain-identity コマンドを使用してドメインを認証しています。

```
aws ses verify-domain-identity --domain example.com
```

出力:

```
{
   "VerificationToken": "eoEmxw+YaYhb3h3iVJHuXMJXqeu1q1/wwmvjuEXAMPLE"
}
```

ドメインの検証を完了するには、返された検証トークンを含むTXTレコードをドメインDNSの設定に追加する必要があります。詳細については、Amazon Simple Email Service デベロッパーガイドSESの「Amazon でのドメインの検証」を参照してください。

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import { VerifyDomainIdentityCommand } from "@aws-sdk/client-ses";
import {
  getUniqueName,
  postfix,
} from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { sesClient } from "./libs/sesClient.js";

/**
 * You must have access to the domain's DNS settings to complete the
 * domain verification process.
 */
const DOMAIN_NAME = postfix(getUniqueName("Domain"), ".example.com");
```

```
const createVerifyDomainIdentityCommand = () => {
  return new VerifyDomainIdentityCommand({ Domain: DOMAIN_NAME });
};

const run = async () => {
  const VerifyDomainIdentityCommand = createVerifyDomainIdentityCommand();

  try {
    return await sesClient.send(VerifyDomainIdentityCommand);
  } catch (err) {
    console.log("Failed to verify domain.", err);
    return err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_domain_identity(self, domain_name):
```

```
        """
        Starts verification of a domain identity. To complete verification, you
 must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response["VerificationToken"]
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES でのAmazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

または **VerifyEmailIdentity**AWS SDKで を使用する CLI

以下のコード例は、VerifyEmailIdentity の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- リージョン間で E メールとドメイン ID をコピーする
- E メール ID を検証してメッセージを送信する

.NET

AWS SDK for .NET

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
   /// <summary>
   /// Starts verification of an email identity. This request sends an email
   /// from Amazon SES to the specified email address. To complete
   /// verification, follow the instructions in the email.
   /// </summary>
   /// <param name="recipientEmailAddress">Email address to verify.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> VerifyEmailIdentityAsync(string
recipientEmailAddress)
   {
       var success = false;
       try
       {
           var response = await
_amazonSimpleEmailService.VerifyEmailIdentityAsync(
               new VerifyEmailIdentityRequest
               {
                   EmailAddress = recipientEmailAddress
               });

           success = response.HttpStatusCode == HttpStatusCode.OK;
       }
       catch (Exception ex)
       {
           Console.WriteLine("VerifyEmailIdentityAsync failed with exception: "
+ ex.Message);
       }

       return success;
   }
```

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

C++

SDK C++ 用

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! Add an email address to the list of identities associated with this account
 and
//! initiate verification.
/*!
  \param emailAddress; The email address to add.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SES::verifyEmailIdentity(const Aws::String &emailAddress,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration)
{
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::VerifyEmailIdentityRequest verifyEmailIdentityRequest;

    verifyEmailIdentityRequest.SetEmailAddress(emailAddress);

    Aws::SES::Model::VerifyEmailIdentityOutcome outcome =
 sesClient.VerifyEmailIdentity(verifyEmailIdentityRequest);

    if (outcome.IsSuccess())
    {
        std::cout << "Email verification initiated." << std::endl;
    }
```

```
    else
    {
        std::cerr << "Error initiating email verification. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

CLI

AWS CLI

Amazon で E メールアドレスを検証するには SES

以下の例では、verify-email-identity コマンドを使用して E メールアドレスを認証しています。

```
aws ses verify-email-identity --email-address user@example.com
```

Amazon を使用して E メールを送信する前にSES、E メールの送信元のアドレスまたはドメインを確認して、E メールを所有していることを証明する必要があります。本番環境へのアクセス権がまだない場合は、Amazon SESメールボックスシミュレーターから提供された E メールアドレスを除き、E メールを送信する E メールアドレスも検証する必要があります。

が呼び出されると、E verify-email-identityメールアドレスに検証 E メールが送信されます。ユーザーは、E メールのリンクをクリックして、検証プロセスを完了する必要があります。

詳細については、「Amazon Simple Email Service デベロッパーガイドSES」の「Amazon での E メールアドレスの検証」を参照してください。

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

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// Import required AWS SDK clients and commands for Node.js
import { VerifyEmailIdentityCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const EMAIL_ADDRESS = "name@example.com";

const createVerifyEmailIdentityCommand = (emailAddress) => {
  return new VerifyEmailIdentityCommand({ EmailAddress: emailAddress });
};

const run = async () => {
  const verifyEmailIdentityCommand =
    createVerifyEmailIdentityCommand(EMAIL_ADDRESS);
  try {
    return await sesClient.send(verifyEmailIdentityCommand);
  } catch (err) {
    console.log("Failed to verify email identity.", err);
    return err;
  }
};
```

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

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_email_identity(self, email_address):
        """
        Starts verification of an email identity. This function causes an email
        to be sent to the specified email address from Amazon SES. To complete
        verification, follow the instructions in the email.

        :param email_address: The email address to verify.
        """
        try:
            self.ses_client.verify_email_identity(EmailAddress=email_address)
            logger.info("Started verification of %s.", email_address)
        except ClientError:
            logger.exception("Couldn't start verification of %s.", email_address)
            raise
```

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

Ruby

SDK Ruby 用の

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
require 'aws-sdk-ses' # v2: require 'aws-sdk'

# Replace recipient@example.com with a "To" address.
recipient = 'recipient@example.com'

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
begin
  ses.verify_email_identity({
                              email_address: recipient
                            })

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

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

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

SES を使用した Amazon のシナリオ AWS SDKs

次のコード例は、 SESを使用して Amazon で一般的なシナリオを実装する方法を示しています AWS SDKs。これらのシナリオでは、Amazon 内で複数の関数を呼び出すSESか、他の関数と組み合わせて特定のタスクを実行する方法を示します AWS のサービス。各シナリオには、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

シナリオは、サービスアクションをコンテキストで理解するのに役立つ中級レベルの経験を対象としています。

例

- Amazon Transcribe ストリーミングアプリケーションを構築する
- を使用して Amazon E SESメールとドメイン ID をあるリージョンから別の AWS リージョンにコピーする AWS SDK
- DynamoDB データを追跡するウェブアプリケーションを作成する
- Amazon Redshift アイテムトラッカーの作成
- Aurora Serverless 作業項目トラッカーの作成
- を使用して Amazon Rekognition でイメージPPE内で検出する AWS SDK
- を使用して Amazon Rekognition でイメージ内のオブジェクトを検出する AWS SDK
- を使用して Amazon Rekognition でビデオ内の人物とオブジェクトを検出する AWS SDK
- Amazon SESSMTPエンドポイントに接続するための認証情報を生成する
- Step Functions を使用して Lambda 関数を呼び出す
- SES を使用して E メール ID を検証し、Amazon でメッセージを送信する AWS SDK

Amazon Transcribe ストリーミングアプリケーションを構築する

次のコード例は、ライブ音声をリアルタイムで記録、転写、翻訳し、結果を E メールで送信するアプリケーションを構築する方法を示しています。

JavaScript

SDK for JavaScript (v3)

Amazon Transcribe を使用して、ライブ音声をリアルタイムで記録、書き起こし、翻訳し、Amazon Simple Email Service (Amazon) を使用して結果を E メールで送信するアプリを構築する方法を示しますSES。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

を使用して Amazon E SESメールとドメイン ID をあるリージョンから別の AWS リージョンにコピーする AWS SDK

次のコード例は、Amazon E SESメールとドメイン ID をあるリージョンから別の AWS リージョンにコピーする方法を示しています。ドメイン ID が Route 53 で管理されている場合、検証レコードはコピー先リージョンのドメインにコピーされます。

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
import argparse
```

```
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def get_identities(ses_client):
    """
    Gets the identities for the current Region. The Region is specified in the
    Boto3 Amazon SES client object.

    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of email identities and the list of domain identities.
    """
    email_identities = []
    domain_identities = []
    try:
        identity_paginator = ses_client.get_paginator("list_identities")
        identity_iterator = identity_paginator.paginate(
            PaginationConfig={"PageSize": 20}
        )
        for identity_page in identity_iterator:
            for identity in identity_page["Identities"]:
                if "@" in identity:
                    email_identities.append(identity)
                else:
                    domain_identities.append(identity)
        logger.info(
            "Found %s email and %s domain identities.",
            len(email_identities),
            len(domain_identities),
        )
    except ClientError:
        logger.exception("Couldn't get identities.")
        raise
    else:
        return email_identities, domain_identities


def verify_emails(email_list, ses_client):
    """
```

```
    Starts verification of a list of email addresses. Verification causes an
 email
    to be sent to each address. To complete verification, the recipient must
 follow
    the instructions in the email.

    :param email_list: The list of email addresses to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of emails that were successfully submitted for
 verification.
    """
    verified_emails = []
    for email in email_list:
        try:
            ses_client.verify_email_identity(EmailAddress=email)
            verified_emails.append(email)
            logger.info("Started verification of %s.", email)
        except ClientError:
            logger.warning("Couldn't start verification of %s.", email)
    return verified_emails


def verify_domains(domain_list, ses_client):
    """
    Starts verification for a list of domain identities. This returns a token for
    each domain, which must be registered as a TXT record with the DNS provider
 for
    the domain.

    :param domain_list: The list of domains to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The generated domain tokens to use to completed verification.
    """
    domain_tokens = {}
    for domain in domain_list:
        try:
            response = ses_client.verify_domain_identity(Domain=domain)
            token = response["VerificationToken"]
            domain_tokens[domain] = token
            logger.info("Got verification token %s for domain %s.", token,
 domain)
        except ClientError:
            logger.warning("Couldn't get verification token for domain %s.",
 domain)
```

```
    return domain_tokens


def get_hosted_zones(route53_client):
    """
    Gets the Amazon Route 53 hosted zones for the current account.

    :param route53_client: A Boto3 Route 53 client.
    :return: The list of hosted zones.
    """
    zones = []
    try:
        zone_paginator = route53_client.get_paginator("list_hosted_zones")
        zone_iterator = zone_paginator.paginate(PaginationConfig={"PageSize":
 20})
        zones = [
            zone for zone_page in zone_iterator for zone in
 zone_page["HostedZones"]
        ]
        logger.info("Found %s hosted zones.", len(zones))
    except ClientError:
        logger.warning("Couldn't get hosted zones.")
    return zones


def find_domain_zone_matches(domains, zones):
    """
    Finds matches between Amazon SES verified domains and Route 53 hosted zones.
    Subdomain matches are taken when found, otherwise root domain matches are
 taken.

    :param domains: The list of domains to match.
    :param zones: The list of hosted zones to match.
    :return: The set of matched domain-zone pairs. When a match is not found, the
             domain is included in the set with a zone value of None.
    """
    domain_zones = {}
    for domain in domains:
        domain_zones[domain] = None
        # Start at the most specific sub-domain and walk up to the root domain
 until a
        # zone match is found.
        domain_split = domain.split(".")
        for index in range(0, len(domain_split) - 1):
```

```
            sub_domain = ".".join(domain_split[index:])
            for zone in zones:
                # Normalize the zone name from Route 53 by removing the trailing
 '.'.
                zone_name = zone["Name"][:-1]
                if sub_domain == zone_name:
                    domain_zones[domain] = zone
                    break
            if domain_zones[domain] is not None:
                break
    return domain_zones


def add_route53_verification_record(domain, token, zone, route53_client):
    """
    Adds a domain verification TXT record to the specified Route 53 hosted zone.
    When a TXT record already exists in the hosted zone for the specified domain,
    the existing values are preserved and the new token is added to the list.

    :param domain: The domain to add.
    :param token: The verification token for the domain.
    :param zone: The hosted zone where the domain verification record is added.
    :param route53_client: A Boto3 Route 53 client.
    """
    domain_token_record_set_name = f"_amazonses.{domain}"
    record_set_paginator =
 route53_client.get_paginator("list_resource_record_sets")
    record_set_iterator = record_set_paginator.paginate(
        HostedZoneId=zone["Id"], PaginationConfig={"PageSize": 20}
    )
    records = []
    for record_set_page in record_set_iterator:
        try:
            txt_record_set = next(
                record_set
                for record_set in record_set_page["ResourceRecordSets"]
                if record_set["Name"][:-1] == domain_token_record_set_name
                and record_set["Type"] == "TXT"
            )
            records = txt_record_set["ResourceRecords"]
            logger.info(
                "Existing TXT record found in set %s for zone %s.",
                domain_token_record_set_name,
                zone["Name"],
```

```
            )
            break
        except StopIteration:
            pass
    records.append({"Value": json.dumps(token)})
    changes = [
        {
            "Action": "UPSERT",
            "ResourceRecordSet": {
                "Name": domain_token_record_set_name,
                "Type": "TXT",
                "TTL": 1800,
                "ResourceRecords": records,
            },
        }
    ]
    try:
        route53_client.change_resource_record_sets(
            HostedZoneId=zone["Id"], ChangeBatch={"Changes": changes}
        )
        logger.info(
            "Created or updated the TXT record in set %s for zone %s.",
            domain_token_record_set_name,
            zone["Name"],
        )
    except ClientError as err:
        logger.warning(
            "Got error %s. Couldn't create or update the TXT record for zone
 %s.",
            err.response["Error"]["Code"],
            zone["Name"],
        )


def generate_dkim_tokens(domain, ses_client):
    """
    Generates DKIM tokens for a domain. These must be added as CNAME records to
 the
    DNS provider for the domain.

    :param domain: The domain to generate tokens for.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of generated DKIM tokens.
    """
```

```
    dkim_tokens = []
    try:
        dkim_tokens = ses_client.verify_domain_dkim(Domain=domain)["DkimTokens"]
        logger.info("Generated %s DKIM tokens for domain %s.", len(dkim_tokens),
 domain)
    except ClientError:
        logger.warning("Couldn't generate DKIM tokens for domain %s.", domain)
    return dkim_tokens


def add_dkim_domain_tokens(hosted_zone, domain, tokens, route53_client):
    """
    Adds DKIM domain token CNAME records to a Route 53 hosted zone.

    :param hosted_zone: The hosted zone where the records are added.
    :param domain: The domain to add.
    :param tokens: The DKIM tokens for the domain to add.
    :param route53_client: A Boto3 Route 53 client.
    """
    try:
        changes = [
            {
                "Action": "UPSERT",
                "ResourceRecordSet": {
                    "Name": f"{token}._domainkey.{domain}",
                    "Type": "CNAME",
                    "TTL": 1800,
                    "ResourceRecords": [{"Value":
 f"{token}.dkim.amazonses.com"}],
                },
            }
            for token in tokens
        ]
        route53_client.change_resource_record_sets(
            HostedZoneId=hosted_zone["Id"], ChangeBatch={"Changes": changes}
        )
        logger.info(
            "Added %s DKIM CNAME records to %s in zone %s.",
            len(tokens),
            domain,
            hosted_zone["Name"],
        )
    except ClientError:
        logger.warning(
```

```
            "Couldn't add DKIM CNAME records for %s to zone %s.",
            domain,
            hosted_zone["Name"],
        )


def configure_sns_topics(identity, topics, ses_client):
    """
    Configures Amazon Simple Notification Service (Amazon SNS) notifications for
    an identity. The Amazon SNS topics must already exist.

    :param identity: The identity to configure.
    :param topics: The list of topics to configure. The choices are Bounce,
 Delivery,
                   or Complaint.
    :param ses_client: A Boto3 Amazon SES client.
    """
    for topic in topics:
        topic_arn = input(
            f"Enter the Amazon Resource Name (ARN) of the {topic} topic or press
 "
            f"Enter to skip: "
        )
        if topic_arn != "":
            try:
                ses_client.set_identity_notification_topic(
                    Identity=identity, NotificationType=topic, SnsTopic=topic_arn
                )
                logger.info("Configured %s for %s notifications.", identity,
 topic)
            except ClientError:
                logger.warning(
                    "Couldn't configure %s for %s notifications.", identity,
 topic
                )


def replicate(source_client, destination_client, route53_client):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print(
        f"Replicating Amazon SES identities and other configuration from "
```

```
        f"{source_client.meta.region_name} to
 {destination_client.meta.region_name}."
    )
    print("-" * 88)

    print(f"Retrieving identities from {source_client.meta.region_name}.")
    source_emails, source_domains = get_identities(source_client)
    print("Email addresses found:")
    print(*source_emails)
    print("Domains found:")
    print(*source_domains)

    print("Starting verification for email identities.")
    dest_emails = verify_emails(source_emails, destination_client)
    print("Getting domain tokens for domain identities.")
    dest_domain_tokens = verify_domains(source_domains, destination_client)

    # Get Route 53 hosted zones and match them with Amazon SES domains.
    answer = input(
        "Is the DNS configuration for your domains managed by Amazon Route 53 (y/
n)? "
    )
    use_route53 = answer.lower() == "y"
    hosted_zones = get_hosted_zones(route53_client) if use_route53 else []
    if use_route53:
        print("Adding or updating Route 53 TXT records for your domains.")
        domain_zones = find_domain_zone_matches(dest_domain_tokens.keys(),
 hosted_zones)
        for domain in domain_zones:
            add_route53_verification_record(
                domain, dest_domain_tokens[domain], domain_zones[domain],
 route53_client
            )
    else:
        print(
            "Use these verification tokens to create TXT records through your DNS
 "
            "provider:"
        )
        pprint(dest_domain_tokens)

    answer = input("Do you want to configure DKIM signing for your identities (y/
n)? ")
    if answer.lower() == "y":
```

```
        # Build a set of unique domains from email and domain identities.
        domains = {email.split("@")[1] for email in dest_emails}
        domains.update(dest_domain_tokens)
        domain_zones = find_domain_zone_matches(domains, hosted_zones)
        for domain, zone in domain_zones.items():
            answer = input(
                f"Do you want to configure DKIM signing for {domain} (y/n)? "
            )
            if answer.lower() == "y":
                dkim_tokens = generate_dkim_tokens(domain, destination_client)
                if use_route53 and zone is not None:
                    add_dkim_domain_tokens(zone, domain, dkim_tokens,
 route53_client)
                else:
                    print(
                        "Add the following DKIM tokens as CNAME records through
 your "
                        "DNS provider:"
                    )
                    print(*dkim_tokens, sep="\n")

    answer = input(
        "Do you want to configure Amazon SNS notifications for your identities
 (y/n)? "
    )
    if answer.lower() == "y":
        for identity in dest_emails + list(dest_domain_tokens.keys()):
            answer = input(
                f"Do you want to configure Amazon SNS topics for {identity} (y/
n)? "
            )
            if answer.lower() == "y":
                configure_sns_topics(
                    identity, ["Bounce", "Delivery", "Complaint"],
 destination_client
                )

    print(f"Replication complete for {destination_client.meta.region_name}.")
    print("-" * 88)


def main():
    boto3_session = boto3.Session()
    ses_regions = boto3_session.get_available_regions("ses")
```

```
    parser = argparse.ArgumentParser(
        description="Copies email address and domain identities from one AWS
 Region to "
        "another. Optionally adds records for domain verification and DKIM "
        "signing to domains that are managed by Amazon Route 53, "
        "and sets up Amazon SNS notifications for events of interest."
    )
    parser.add_argument(
        "source_region", choices=ses_regions, help="The region to copy from."
    )
    parser.add_argument(
        "destination_region", choices=ses_regions, help="The region to copy to."
    )
    args = parser.parse_args()
    source_client = boto3.client("ses", region_name=args.source_region)
    destination_client = boto3.client("ses", region_name=args.destination_region)
    route53_client = boto3.client("route53")
    replicate(source_client, destination_client, route53_client)


if __name__ == "__main__":
    main()
```

- API 詳細については、 AWS SDK for Python (Boto3) APIリファレンスの以下のトピックを参照してください。
  - ListIdentities
  - SetIdentityNotificationTopic
  - VerifyDomainDkim
  - VerifyDomainIdentity
  - VerifyEmailIdentity

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

DynamoDB データを追跡するウェブアプリケーションを作成する

次のコード例は、Amazon DynamoDB テーブル内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信するウェブアプリケーションを作成する方法を示しています。

.NET

AWS SDK for .NET

Amazon DynamoDB を使用して、NETDynamoDB の作業データを追跡する動的ウェブアプリケーションAPIを作成する方法を示します。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- DynamoDB
- Amazon SES

Java

SDK for Java 2.x

Amazon DynamoDB を使用してAPI、DynamoDB 作業データを追跡する動的ウェブアプリケーションを作成する方法を示します。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- DynamoDB
- Amazon SES

Kotlin

SDK Kotlin 用の

Amazon DynamoDB を使用してAPI、DynamoDB 作業データを追跡する動的ウェブアプリケーションを作成する方法を示します。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- DynamoDB
- Amazon SES

Python

SDK for Python (Boto3)

を使用して AWS SDK for Python (Boto3) Amazon DynamoDB の作業項目を追跡し、Amazon Simple Email Service (Amazon) を使用してレポートを E メールで送信するRESTサービスを作成する方法を示しますSES。この例では、Flask ウェブフレームワークを使用してHTTPルーティングを処理し、React ウェブページと統合して完全に機能するウェブアプリケーションを提示します。

- と統合する Flask RESTサービスを構築します AWS のサービス。
- DynamoDB テーブルに保存されている作業項目の読み取り、書き込み、更新を行います。
- Amazon SESを使用して、作業項目の E メールレポートを送信します。

完全なソースコードと、セットアップと実行の手順については、 のAWS コード例リポジトリの詳細な例を参照してください GitHub。

この例で使用されているサービス

- DynamoDB
- Amazon SES

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

Amazon Redshift アイテムトラッカーの作成

次のコード例は、Amazon Redshift データベースを使用して、作業項目を追跡してレポートするウェブアプリケーションを作成する方法を示しています。

Java

SDK for Java 2.x

Amazon Redshift データベースに保存されている作業項目を追跡してレポートするウェブアプリケーションを作成する方法を説明します。

Amazon Redshift データをクエリRESTAPIし、React アプリケーションで使用する Spring をセットアップする方法の完全なソースコードと手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Redshift
- Amazon SES

Kotlin

SDK Kotlin 用の

Amazon Redshift データベースに保存されている作業項目を追跡してレポートするウェブアプリケーションを作成する方法を説明します。

Amazon Redshift データをRESTAPIクエリし、React アプリケーションで使用する Spring をセットアップする方法の完全なソースコードと手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Redshift
- Amazon SES

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信するウェブアプリケーションを作成する方法を示しています。

.NET

AWS SDK for .NET

を使用して AWS SDK for .NET 、Amazon Aurora データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon) を使用してレポートを E メールで送信するウェブアプリケーションを作成する方法を示しますSES。この例では、React.js で構築されたフロントエンドを使用して RESTful .NET バックエンドとやり取りします。

- React ウェブアプリケーションを AWS サービスと統合します。
- Aurora テーブルの項目を一覧表示、更新、削除します。
- Amazon を使用して、フィルタリングされた作業項目の E メールレポートを送信しますSES。
- 含まれている AWS CloudFormation スクリプトを使用してサンプルリソースをデプロイおよび管理します。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

C++

SDK C++ 用

Amazon Aurora Serverless データベースに保存されている作業項目を追跡して報告するウェブアプリケーションを作成する方法を説明します。

完全なソースコードと、Amazon Aurora Serverless データをクエリRESTAPIする C++ をセットアップする方法、および React アプリケーションで使用する方法については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Aurora

- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Java

SDK for Java 2.x

Amazon RDS データベースに保存されている作業項目を追跡してレポートするウェブアプリケーションを作成する方法を示します。

Amazon Aurora Serverless データをRESTAPIクエリし、React アプリケーションで使用する Spring をセットアップする方法の完全なソースコードと手順については、「」の詳細な例を参照してくださいGitHub。

完全なソースコードと、 を使用する例をセットアップして実行する方法については API、JDBC「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

JavaScript

SDK for JavaScript (v3)

AWS SDK for JavaScript (v3) を使用して、Amazon Aurora データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon) を使用してレポートを E メールで送信するウェブアプリケーションを作成する方法を示しますSES。この例では、React.js で構築されたフロントエンドを使用して Express Node.js バックエンドと対話します。

- React.js ウェブアプリケーションを と統合します AWS のサービス。
- Aurora テーブルの項目を一覧表示、追加、更新します。
- Amazon を使用して、フィルタリングされた作業項目の E メールレポートを送信します SES。

- 含まれている AWS CloudFormation スクリプトを使用してサンプルリソースをデプロイおよび管理します。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Kotlin

SDK Kotlin 用の

Amazon RDS データベースに保存されている作業項目を追跡してレポートするウェブアプリケーションを作成する方法を示します。

Amazon Aurora Serverless データをRESTAPIクエリし、React アプリケーションで使用する Spring を設定する方法の完全なソースコードと手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

PHP

PHP 用の SDK

を使用して AWS SDK for PHP 、Amazon RDS データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon) を使用してレポートを E メールで送信するウェブアプリケーションを作成する方法を示しますSES。この例では、React.js で構築されたフロントエンドを使用してRESTfulPHPバックエンドとやり取りします。

- React.js ウェブアプリケーションを AWS サービスに統合します。
- Amazon RDSテーブルの項目を一覧表示、追加、更新、削除します。
- Amazon を使用して、フィルタリングされた作業項目の E メールレポートを送信しますSES。
- 含まれている AWS CloudFormation スクリプトを使用してサンプルリソースをデプロイおよび管理します。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Python

SDK for Python (Boto3)

を使用して AWS SDK for Python (Boto3) 、Amazon Aurora Serverless データベース内の作業項目を追跡し、Amazon Simple Email Service (Amazon) を使用してレポートを E メールで送信するRESTサービスを作成する方法を示しますSES。この例では、Flask ウェブフレームワークを使用してHTTPルーティングを処理し、React ウェブページと統合して完全に機能するウェブアプリケーションを提示します。

- と統合する Flask RESTサービスを構築します AWS のサービス。
- Aurora Serverless データベースに保存されている作業項目の読み取り、書き込み、更新を行います。
- データベース認証情報を含む AWS Secrets Manager シークレットを作成し、それを使用してデータベースへの呼び出しを認証します。
- Amazon SESを使用して、作業項目の E メールレポートを送信します。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

を使用して Amazon Rekognition でイメージPPE内で検出する AWS SDK

次のコード例は、Amazon Rekognition を使用してイメージ内の個人用保護具 (PPE) を検出するアプリを構築する方法を示しています。

Java

SDK for Java 2.x

個人用保護具を使用してイメージを検出する AWS Lambda 関数を作成する方法を示します。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

を使用して Amazon Rekognition でイメージ内のオブジェクトを検出する AWS SDK

次のコード例は、Amazon Rekognition を使用してイメージでカテゴリ別にオブジェクトを検出するアプリケーションを構築する方法を示しています。

.NET

AWS SDK for .NET

Amazon Rekognition NET を使用して、Amazon Rekognition を使用して Amazon Simple Storage Service (Amazon S3) バケットにあるイメージ内のオブジェクトをカテゴリ別に識別するアプリAPIを作成する方法を示します。アプリは、Amazon Simple Email Service (Amazon) を使用して、結果を含む E メール通知を管理者に送信しますSES。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK for Java 2.x

Amazon Rekognition Java を使用して、Amazon Rekognition を使用して Amazon Simple Storage Service (Amazon S3) バケットにあるイメージ内のオブジェクトをカテゴリ別に識別するアプリケーションAPIを作成する方法を示します。アプリは、Amazon Simple Email Service (Amazon) を使用して、結果を含む E メール通知を管理者に送信しますSES。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK for JavaScript (v3)

で Amazon Rekognition を使用して、Amazon Rekognition を使用して Amazon Simple Storage Service (Amazon S3) バケットにあるイメージ内のオブジェクトをカテゴリ別に識別するアプリ AWS SDK for JavaScript を作成する方法を示します。アプリは、Amazon Simple Email Service (Amazon) を使用して、結果を含む E メール通知を管理者に送信しますSES。

以下ではその方法を説明しています。

- Amazon Cognito を使用して認証されていないユーザーを作成します。
- Amazon Rekognition を使用して、オブジェクトのイメージを分析します。
- Amazon の E メールアドレスを確認しますSES。
- Amazon を使用して E メール通知を送信しますSES。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK Kotlin 用の

Amazon Rekognition Kotlin を使用して、Amazon Rekognition を使用して Amazon Simple Storage Service (Amazon S3) バケットにあるイメージ内のオブジェクトをカテゴリ別に識別するアプリケーションAPIを作成する方法を示します。アプリは、Amazon Simple Email Service (Amazon) を使用して、結果を含む E メール通知を管理者に送信しますSES。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Rekognition

- Amazon S3
- Amazon SES

Python

SDK for Python (Boto3)

を使用して AWS SDK for Python (Boto3) 、次のことを可能にするウェブアプリケーションを作成する方法を示します。

- Amazon Simple Storage Service (Amazon S3) バケットに、写真をアップロードします。
- Amazon Rekognition を使用して、写真を分析およびラベル付けします。
- Amazon Simple Email Service (Amazon SES) を使用して、イメージ分析の E メールレポートを送信します。

この例には、React で構築 JavaScript された で記述されたウェブページと、Flask- で構築された Python で記述されたRESTサービスの 2 つの主要なコンポーネントが含まれています RESTful。

React ウェブページを使用すると、次のことができます。

- S3 バケットに保存されているイメージのリストを表示します。
- イメージを S3 バケットにアップロードします。
- イメージ内で検出された項目を識別するイメージとラベルを表示します。
- S3 バケット内のすべてのイメージのレポートを取得し、レポートの E メールを送信します。

ウェブページは RESTサービスを呼び出します。サービスはリクエストを AWS に送信して、以下のアクションを実行します。

- S3 バケット内のイメージのリストを取得し、フィルタリングします。
- Amazon S3 バケットに写真をアップロードします。
- Amazon Rekognition を使用して個々の写真を分析し、写真で検出された項目を識別するラベルのリストを取得します。
- S3 バケット内のすべての写真を分析し、Amazon SES を使用してレポートを E メールで送信します。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

を使用して Amazon Rekognition でビデオ内の人物とオブジェクトを検出する AWS SDK

次のコード例は、Amazon Rekognition で動画内の人やオブジェクトを検出する方法を示します。

Java

SDK for Java 2.x

Amazon Rekognition Java を使用して、Amazon Simple Storage Service (Amazon S3) バケットにあるビデオ内の顔やオブジェクトを検出するアプリケーションAPIを作成する方法を示します。アプリは、Amazon Simple Email Service (Amazon) を使用して、結果を含む E メール通知を管理者に送信しますSES。

完全なソースコードと、セットアップと実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

Amazon SESSMTPエンドポイントに接続するための認証情報を生成する

次のコード例は、Amazon SESSMTPエンドポイントに接続するための認証情報を生成する方法を示しています。

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
#!/usr/bin/env python3

import hmac
import hashlib
import base64
import argparse

SMTP_REGIONS = [
    "us-east-2",  # US East (Ohio)
    "us-east-1",  # US East (N. Virginia)
    "us-west-2",  # US West (Oregon)
    "ap-south-1",  # Asia Pacific (Mumbai)
    "ap-northeast-2",  # Asia Pacific (Seoul)
    "ap-southeast-1",  # Asia Pacific (Singapore)
    "ap-southeast-2",  # Asia Pacific (Sydney)
    "ap-northeast-1",  # Asia Pacific (Tokyo)
    "ca-central-1",  # Canada (Central)
    "eu-central-1",  # Europe (Frankfurt)
    "eu-west-1",  # Europe (Ireland)
    "eu-west-2",  # Europe (London)
    "eu-south-1",  # Europe (Milan)
    "eu-north-1",  # Europe (Stockholm)
    "sa-east-1",  # South America (Sao Paulo)
    "us-gov-west-1",  # AWS GovCloud (US)
    "us-gov-east-1",  # AWS GovCloud (US)
]
```

```
# These values are required to calculate the signature. Do not change them.
DATE = "11111111"
SERVICE = "ses"
MESSAGE = "SendRawEmail"
TERMINAL = "aws4_request"
VERSION = 0x04


def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()


def calculate_key(secret_access_key, region):
    if region not in SMTP_REGIONS:
        raise ValueError(f"The {region} Region doesn't have an SMTP endpoint.")

    signature = sign(("AWS4" + secret_access_key).encode("utf-8"), DATE)
    signature = sign(signature, region)
    signature = sign(signature, SERVICE)
    signature = sign(signature, TERMINAL)
    signature = sign(signature, MESSAGE)
    signature_and_version = bytes([VERSION]) + signature
    smtp_password = base64.b64encode(signature_and_version)
    return smtp_password.decode("utf-8")


def main():
    parser = argparse.ArgumentParser(
        description="Convert a Secret Access Key to an SMTP password."
    )
    parser.add_argument("secret", help="The Secret Access Key to convert.")
    parser.add_argument(
        "region",
        help="The AWS Region where the SMTP password will be used.",
        choices=SMTP_REGIONS,
    )
    args = parser.parse_args()
    print(calculate_key(args.secret, args.region))


if __name__ == "__main__":
    main()
```

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

Step Functions を使用して Lambda 関数を呼び出す

次のコード例は、 AWS Lambda 関数を順番に呼び出す AWS Step Functions ステートマシンを作成する方法を示しています。

Java

SDK for Java 2.x

AWS Step Functions と を使用して AWS サーバーレスワークフローを作成する方法を示します AWS SDK for Java 2.x。各ワークフローステップは、 AWS Lambda 関数を使用して実装されます。

完全なソースコードとセットアップおよび実行の手順については、「」の詳細な例を参照してくださいGitHub。

この例で使用されているサービス

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

SES を使用して E メール ID を検証し、Amazon でメッセージを送信する AWS SDK

次のコードサンプルは、以下の操作方法を示しています。

- Amazon で E メールアドレスを追加および検証しますSES。
- 標準の E メールメッセージを送信します。
- テンプレートを作成し、テンプレート化された E メールメッセージを送信します。
- Amazon SESSMTPサーバーを使用してメッセージを送信します。

Python

SDK for Python (Boto3)

> **Note**
>
> 詳細については、「」を参照してください GitHub。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Amazon で E メールアドレスを確認しSES、メッセージを送信します。

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Simple Email Service (Amazon SES) email demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    ses_client = boto3.client("ses")
    ses_identity = SesIdentity(ses_client)
    ses_mail_sender = SesMailSender(ses_client)
    ses_template = SesTemplate(ses_client)
    email = input("Enter an email address to send mail with Amazon SES: ")
    status = ses_identity.get_identity_status(email)
    verified = status == "Success"
    if not verified:
        answer = input(
            f"The address '{email}' is not verified with Amazon SES. Unless your
 "
            f"Amazon SES account is out of sandbox, you can send mail only from "
            f"and to verified accounts. Do you want to verify this account for
 use "
            f"with Amazon SES? If yes, the address will receive a verification "
            f"email (y/n): "
        )
        if answer.lower() == "y":
            ses_identity.verify_email_identity(email)
            print(f"Follow the steps in the email to {email} to complete
 verification.")
            print("Waiting for verification...")
            try:
                ses_identity.wait_until_identity_exists(email)
```

```
                print(f"Identity verified for {email}.")
                verified = True
            except WaiterError:
                print(
                    f"Verification timeout exceeded. You must complete the "
                    f"steps in the email sent to {email} to verify the address."
                )

    if verified:
        test_message_text = "Hello from the Amazon SES mail demo!"
        test_message_html = "<p>Hello!</p><p>From the <b>Amazon SES</b> mail
demo!</p>"

        print(f"Sending mail from {email} to {email}.")
        ses_mail_sender.send_email(
            email,
            SesDestination([email]),
            "Amazon SES demo",
            test_message_text,
            test_message_html,
        )
        input("Mail sent. Check your inbox and press Enter to continue.")

        template = {
            "name": "doc-example-template",
            "subject": "Example of an email template.",
            "text": "This is what {{name}} will {{action}} if {{name}} can't
display "
            "HTML.",
            "html": "<p><i>This</i> is what {{name}} will {{action}} if {{name}}
"
            "<b>can</b> display HTML.</p>",
        }
        print("Creating a template and sending a templated email.")
        ses_template.create_template(**template)
        template_data = {"name": email.split("@")[0], "action": "read"}
        if ses_template.verify_tags(template_data):
            ses_mail_sender.send_templated_email(
                email, SesDestination([email]), ses_template.name(),
template_data
            )
            input("Mail sent. Check your inbox and press Enter to continue.")

        print("Sending mail through the Amazon SES SMTP server.")
```

```
        boto3_session = boto3.Session()
        region = boto3_session.region_name
        credentials = boto3_session.get_credentials()
        port = 587
        smtp_server = f"email-smtp.{region}.amazonaws.com"
        password = calculate_key(credentials.secret_key, region)
        message = """
Subject: Hi there

This message is sent from the Amazon SES SMTP mail demo."""
        context = ssl.create_default_context()
        with smtplib.SMTP(smtp_server, port) as server:
            server.starttls(context=context)
            server.login(credentials.access_key, password)
            server.sendmail(email, email, message)
        print("Mail sent. Check your inbox!")

    if ses_template.template is not None:
        print("Deleting demo template.")
        ses_template.delete_template()
    if verified:
        answer = input(f"Do you want to remove {email} from Amazon SES (y/n)? ")
        if answer.lower() == "y":
            ses_identity.delete_identity(email)
    print("Thanks for watching!")
    print("-" * 88)
```

Amazon SES ID アクションをラップする関数を作成します。

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def verify_domain_identity(self, domain_name):
```

```
        """
        Starts verification of a domain identity. To complete verification, you
 must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response["VerificationToken"]
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token


    def verify_email_identity(self, email_address):
        """
        Starts verification of an email identity. This function causes an email
        to be sent to the specified email address from Amazon SES. To complete
        verification, follow the instructions in the email.

        :param email_address: The email address to verify.
        """
        try:
            self.ses_client.verify_email_identity(EmailAddress=email_address)
            logger.info("Started verification of %s.", email_address)
        except ClientError:
            logger.exception("Couldn't start verification of %s.", email_address)
            raise


    def wait_until_identity_exists(self, identity):
        """
        Waits until an identity exists. The waiter polls Amazon SES until the
```

```
        identity has been successfully verified or until it exceeds its maximum
time.

        :param identity: The identity to wait for.
        """
        try:
            waiter = self.ses_client.get_waiter("identity_exists")
            logger.info("Waiting until %s exists.", identity)
            waiter.wait(Identities=[identity])
        except WaiterError:
            logger.error("Waiting for identity %s failed or timed out.",
identity)
            raise


    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity]
            )
            status = response["VerificationAttributes"].get(
                identity, {"VerificationStatus": "NotFound"}
            )["VerificationStatus"]
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status


    def delete_identity(self, identity):
        """
        Deletes an identity.

        :param identity: The identity to remove.
        """
```

```
        try:
            self.ses_client.delete_identity(Identity=identity)
            logger.info("Deleted identity %s.", identity)
        except ClientError:
            logger.exception("Couldn't delete identity %s.", identity)
            raise


    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as
 EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items
            )
            identities = response["Identities"]
            logger.info("Got %s identities for the current account.",
 len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
            raise
        else:
            return identities
```

Amazon SES テンプレートアクションをラップする関数を作成します。

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
```

```
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text +
html))
        logger.info("Extracted template tags: %s", self.template_tags)


    def create_template(self, name, subject, text, html):
        """
        Creates an email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.create_template(Template=template)
            logger.info("Created template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't create template %s.", name)
            raise


    def delete_template(self):
        """
        Deletes an email template.
```

```
        """
        try:

self.ses_client.delete_template(TemplateName=self.template["TemplateName"])
            logger.info("Deleted template %s.", self.template["TemplateName"])
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template["TemplateName"]
            )
            raise


    def get_template(self, name):
        """
        Gets a previously created email template.

        :param name: The name of the template to retrieve.
        :return: The retrieved email template.
        """
        try:
            response = self.ses_client.get_template(TemplateName=name)
            self.template = response["Template"]
            logger.info("Got template %s.", name)
            self._extract_tags(
                self.template["SubjectPart"],
                self.template["TextPart"],
                self.template["HtmlPart"],
            )
        except ClientError:
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template


    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
```

```
            response = self.ses_client.list_templates()
            templates = response["TemplatesMetadata"]
            logger.info("Got %s templates.", len(templates))
        except ClientError:
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates


    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                "TemplateName": name,
                "SubjectPart": subject,
                "TextPart": text,
                "HtmlPart": html,
            }
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't update template %s.", name)
            raise
```

Amazon E SESメールアクションをラップする関数を作成します。

```
class SesDestination:
    """Contains data about an email destination."""

    def __init__(self, tos, ccs=None, bccs=None):
```

```
        """
        :param tos: The list of recipients on the 'To:' line.
        :param ccs: The list of recipients on the 'CC:' line.
        :param bccs: The list of recipients on the 'BCC:' line.
        """
        self.tos = tos
        self.ccs = ccs
        self.bccs = bccs

    def to_service_format(self):
        """
        :return: The destination data in the format expected by Amazon SES.
        """
        svc_format = {"ToAddresses": self.tos}
        if self.ccs is not None:
            svc_format["CcAddresses"] = self.ccs
        if self.bccs is not None:
            svc_format["BccAddresses"] = self.bccs
        return svc_format



class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""

    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client


    def send_email(self, source, destination, subject, text, html,
 reply_tos=None):
        """
        Sends an email.

        Note: If your account is in the Amazon SES  sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param subject: The subject of the email.
        :param text: The plain text version of the body of the email.
```

```
        :param html: The HTML version of the body of the email.
        :param reply_tos: Email accounts that will receive a reply if the
recipient
                          replies to the message.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Message": {
                "Subject": {"Data": subject},
                "Body": {"Text": {"Data": text}, "Html": {"Data": html}},
            },
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent mail %s from %s to %s.", message_id, source,
destination.tos
            )
        except ClientError:
            logger.exception(
                "Couldn't send mail from %s to %s.", source, destination.tos
            )
            raise
        else:
            return message_id


    def send_templated_email(
        self, source, destination, template_name, template_data, reply_tos=None
    ):
        """
        Sends an email based on a template. A template contains replaceable tags
        each enclosed in two curly braces, such as {{name}}. The template data
passed
        in this function contains key-value pairs that define the values to
insert
        in place of the template tags.

        Note: If your account is in the Amazon SES  sandbox, the source and
```

```
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement
values
                              that are inserted in the template before it is
sent.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            "Source": source,
            "Destination": destination.to_service_format(),
            "Template": template_name,
            "TemplateData": json.dumps(template_data),
        }
        if reply_tos is not None:
            send_args["ReplyToAddresses"] = reply_tos
        try:
            response = self.ses_client.send_templated_email(**send_args)
            message_id = response["MessageId"]
            logger.info(
                "Sent templated mail %s from %s to %s.",
                message_id,
                source,
                destination.tos,
            )
        except ClientError:
            logger.exception(
                "Couldn't send templated mail from %s to %s.", source,
destination.tos
            )
            raise
        else:
            return message_id
```

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

- DeleteIdentity
- DeleteTemplate
- GetIdentityVerificationAttributes
- GetTemplate
- ListIdentities
- ListTemplates
- SendEmail
- SendTemplatedEmail
- UpdateTemplate
- VerifyDomainIdentity
- VerifyEmailIdentity

開発者ガイドとコード例の完全なリスト AWS SDKについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、開始方法に関する情報と以前のSDKバージョンに関する詳細も含まれています。

SDK を使用した Amazon SES API v2 のコード例 AWS SDKs

次のコード例は、 AWS Software Development Kit (SDK) で Amazon SES API v2 を使用する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

コードの例

- AWS SDKs を使用した Amazon SES API v2 の基本的な例
  - SDK を使用した Amazon SES API v2 のアクション AWS SDKs

    - AWS SDK CreateContactで を使用する
    - AWS SDK CreateContactListで を使用する
    - AWS SDK CreateEmailIdentityで を使用する
    - AWS SDK CreateEmailTemplateで を使用する
    - AWS SDK DeleteContactListで を使用する
    - AWS SDK DeleteEmailIdentityで を使用する
    - AWS SDK DeleteEmailTemplateで を使用する
    - AWS SDK GetEmailIdentityで を使用する
    - AWS SDK ListContactListsで を使用する
    - AWS SDK ListContactsで を使用する
    - AWS SDK SendEmailで を使用する
- SDK を使用した Amazon SES API v2 のシナリオ AWS SDKs
    - AWS SDK を使用した完全な Amazon SES API v2 ニュースレターシナリオ

AWS SDKs を使用した Amazon SES API v2 の基本的な例

次のコード例は、 AWS SDK を使用した Amazon Simple Email Service API v2 の基本的な使用方法を説明しています。

例

- SDK を使用した Amazon SES API v2 のアクション AWS SDKs
    - AWS SDK CreateContactで を使用する
    - AWS SDK CreateContactListで を使用する
    - AWS SDK CreateEmailIdentityで を使用する
    - AWS SDK CreateEmailTemplateで を使用する
    - AWS SDK DeleteContactListで を使用する
    - AWS SDK DeleteEmailIdentityで を使用する
    - AWS SDK DeleteEmailTemplateで を使用する
    - AWS SDK GetEmailIdentityで を使用する
    - AWS SDK ListContactListsで を使用する
    - AWS SDK ListContactsで を使用する
    - AWS SDK SendEmailで を使用する

SDK を使用した Amazon SES API v2 のアクション AWS SDKs

次のコード例は、 AWS SDKs を使用して個々の Amazon SES API v2 アクションを実行する方法を示しています。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

これらは Amazon SES API v2 API を呼び出すもので、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。アクションは SDK を使用した Amazon SES API v2 のシナリオ AWS SDKs のコンテキスト内で確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細なリストについては、「Amazon Simple Email Service API v2 API リファレンス」を参照してください。

例

- AWS SDK CreateContactで を使用する
- AWS SDK CreateContactListで を使用する
- AWS SDK CreateEmailIdentityで を使用する
- AWS SDK CreateEmailTemplateで を使用する
- AWS SDK DeleteContactListで を使用する
- AWS SDK DeleteEmailIdentityで を使用する
- AWS SDK DeleteEmailTemplateで を使用する
- AWS SDK GetEmailIdentityで を使用する
- AWS SDK ListContactListsで を使用する
- AWS SDK ListContactsで を使用する
- AWS SDK SendEmailで を使用する

AWS SDK **`CreateContact`**で を使用する

以下のコード例は、`CreateContact` の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ニュースレターのシナリオ

.NET

AWS SDK for .NET

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

```
   /// <summary>
   /// Creates a contact and adds it to the specified contact list.
   /// </summary>
   /// <param name="emailAddress">The email address of the contact.</param>
   /// <param name="contactListName">The name of the contact list.</param>
   /// <returns>The response from the CreateContact operation.</returns>
   public async Task<bool> CreateContactAsync(string emailAddress, string
contactListName)
   {
       var request = new CreateContactRequest
       {
           EmailAddress = emailAddress,
           ContactListName = contactListName
       };

       try
       {
           var response = await _sesClient.CreateContactAsync(request);
           return response.HttpStatusCode == HttpStatusCode.OK;
       }
       catch (AlreadyExistsException ex)
       {
           Console.WriteLine($"Contact with email address {emailAddress} already
exists in the contact list {contactListName}.");
           Console.WriteLine(ex.Message);
           return true;
       }
       catch (NotFoundException ex)
       {
           Console.WriteLine($"The contact list {contactListName} does not
exist.");
           Console.WriteLine(ex.Message);
```

```
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact:
{ex.Message}");
        }
        return false;
    }
```

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

Java

SDK for Java 2.x

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

```
    try {
      // Create a new contact with the provided email address in the
      CreateContactRequest contactRequest = CreateContactRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .emailAddress(emailAddress)
          .build();

      sesClient.createContact(contactRequest);
      contacts.add(emailAddress);

      System.out.println("Contact created: " + emailAddress);
```

```
        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
                            .html(Content.builder().data(welcomeHtml).build())
                            .build())
                        .build())
                .build())
            .build();
        SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
      } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
 proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
      } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
        throw e;
      }
    }
```

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

Python

SDK for Python (Boto3)

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

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


            try:
                # Create a new contact
                self.ses_client.create_contact(
                    ContactListName=CONTACT_LIST_NAME, EmailAddress=email
```

```
                )
                print(f"Contact with email '{email}' created successfully.")

                # Send the welcome email
                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons
Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
                if self.sleep:
                    # 1 email per second in sandbox mode, remove in production.
                    sleep(1.1)
            except ClientError as e:
                # If the contact already exists, skip and proceed
                if e.response["Error"]["Code"] == "AlreadyExistsException":
                    print(f"Contact with email '{email}' already exists.
Skipping...")
                else:
                    raise e
```

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

Rust

SDK for Rust

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

```
async fn add_contact(client: &Client, list: &str, email: &str) -> Result<(),
 Error> {
    client
        .create_contact()
        .contact_list_name(list)
        .email_address(email)
        .send()
        .await?;

    println!("Created contact");

    Ok(())
}
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK **CreateContactList**で を使用する

以下のコード例は、CreateContactList の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ニュースレターのシナリオ

.NET

AWS SDK for .NET

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

```
    /// <summary>
    /// Creates a contact list with the specified name.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateContactListAsync(string contactListName)
    {
        var request = new CreateContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact list with name {contactListName} already
exists.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for contact lists has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
```

```
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact
list: {ex.Message}");
        }
        return false;
    }
```

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

Java

SDK for Java 2.x

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

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
```

```
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
      throw e;
    }
```

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

Python

SDK for Python (Boto3)

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

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """
```

```
    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:

 self.ses_client.create_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' created successfully.")
        except ClientError as e:
            # If the contact list already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Contact list '{CONTACT_LIST_NAME}' already exists.")
            else:
                raise e
```

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

Rust

SDK for Rust

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

```
async fn make_list(client: &Client, contact_list: &str) -> Result<(), Error> {
    client
        .create_contact_list()
        .contact_list_name(contact_list)
        .send()
        .await?;

    println!("Created contact list.");

    Ok(())
```

```
}
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK **CreateEmailIdentity**で を使用する

以下のコード例は、CreateEmailIdentity の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ニュースレターのシナリオ

.NET

AWS SDK for .NET

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

```
    /// <summary>
    /// Creates an email identity (email address or domain) and starts the
verification process.
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to create and
verify.</param>
    /// <returns>The response from the CreateEmailIdentity operation.</returns>
    public async Task<CreateEmailIdentityResponse>
CreateEmailIdentityAsync(string emailIdentity)
    {
        var request = new CreateEmailIdentityRequest
```

```
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.CreateEmailIdentityAsync(request);
            return response;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email identity {emailIdentity} already exists.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being
modified by another operation or thread.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email identities has been
exceeded.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine($"An error occurred while creating the email
identity: {ex.Message}");
            throw;
        }
    }
```

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

Java

SDK for Java 2.x

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

```
    try {
      CreateEmailIdentityRequest createEmailIdentityRequest =
CreateEmailIdentityRequest.builder()
          .emailIdentity(verifiedEmail)
          .build();
      sesClient.createEmailIdentity(createEmailIdentityRequest);
      System.out.println("Email identity created: " + verifiedEmail);
    } catch (AlreadyExistsException e) {
      System.out.println("Email identity already exists, skipping creation: " +
verifiedEmail);
    } catch (NotFoundException e) {
      System.err.println("The provided email address is not verified: " +
verifiedEmail);
      throw e;
    } catch (LimitExceededException e) {
      System.err
          .println("You have reached the limit for email identities. Please
remove some identities and try again.");
      throw e;
    } catch (SesV2Exception e) {
```

```
            System.err.println("Error creating email identity: " + e.getMessage());
            throw e;
        }
```

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

Python

SDK for Python (Boto3)

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

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """
```

```
    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:

self.ses_client.create_email_identity(EmailIdentity=self.verified_email)
            print(f"Email identity '{self.verified_email}' created
successfully.")
        except ClientError as e:
            # If the email identity already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email identity '{self.verified_email}' already exists.")
            else:
                raise e
```

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

Rust

SDK for Rust

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

```
        match self
            .client
            .create_email_identity()
            .email_identity(self.verified_email.clone())
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email identity created
successfully.")?,
            Err(e) => match e.into_service_error() {
```

```
                CreateEmailIdentityError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email identity already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email identity: {}", e)),
            },
        }
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK **CreateEmailTemplate**で を使用する

以下のコード例は、CreateEmailTemplate の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ニュースレターのシナリオ

.NET

AWS SDK for .NET

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

```
    /// <summary>
    /// Creates an email template with the specified content.
    /// </summary>
```

```
    /// <param name="templateName">The name of the email template.</param>
    /// <param name="subject">The subject of the email template.</param>
    /// <param name="htmlContent">The HTML content of the email template.</param>
    /// <param name="textContent">The text content of the email template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string templateName, string
subject, string htmlContent, string textContent)
    {
        var request = new CreateEmailTemplateRequest
        {
            TemplateName = templateName,
            TemplateContent = new EmailTemplateContent
            {
                Subject = subject,
                Html = htmlContent,
                Text = textContent
            }
        };

        try
        {
            var response = await _sesClient.CreateEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email template with name {templateName} already
exists.");
            Console.WriteLine(ex.Message);
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email templates has been
exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
```

```
            Console.WriteLine($"An error occurred while creating the email
 template: {ex.Message}");
        }

        return false;
    }
```

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

Java

SDK for Java 2.x

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

```
    try {
      // Create an email template named "weekly-coupons"
      String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
      String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");

      CreateEmailTemplateRequest templateRequest =
 CreateEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .templateContent(EmailTemplateContent.builder()
              .subject("Weekly Coupons Newsletter")
              .html(newsletterHtml)
              .text(newsletterText)
              .build())
          .build();

      sesClient.createEmailTemplate(templateRequest);

      System.out.println("Email template created: " + TEMPLATE_NAME);
```

```
    } catch (AlreadyExistsException e) {
      // If the template already exists, skip this step and proceed with the next
      // operation
      System.out.println("Email template already exists, skipping creation...");
    } catch (LimitExceededException e) {
      // If the limit for email templates is exceeded, fail the workflow and
inform
      // the user
      System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
      throw e;
    } catch (Exception e) {
      System.err.println("Error occurred while creating email template: " +
e.getMessage());
      throw e;
    }
```

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

Python

SDK for Python (Boto3)

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

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
```

```
            workflow.send_coupon_newsletter()
            workflow.monitor_and_review()
        except ClientError as e:
            print_error(e)
        workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            template_content = {
                "Subject": "Weekly Coupons Newsletter",
                "Html": load_file_content("coupon-newsletter.html"),
                "Text": load_file_content("coupon-newsletter.txt"),
            }
            self.ses_client.create_email_template(
                TemplateName=TEMPLATE_NAME, TemplateContent=template_content
            )
            print(f"Email template '{TEMPLATE_NAME}' created successfully.")
        except ClientError as e:
            # If the template already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email template '{TEMPLATE_NAME}' already exists.")
            else:
                raise e
```

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

Rust

SDK for Rust

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

```
        let template_html =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.html")
                .unwrap_or_else(|_| "Missing coupon-
newsletter.html".to_string());
        let template_text =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.txt")
                .unwrap_or_else(|_| "Missing coupon-newsletter.txt".to_string());

        // Create the email template
        let template_content = EmailTemplateContent::builder()
            .subject("Weekly Coupons Newsletter")
            .html(template_html)
            .text(template_text)
            .build();

        match self
            .client
            .create_email_template()
            .template_name(TEMPLATE_NAME)
            .template_content(template_content)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template created
 successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailTemplateError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email template already exists, skipping creation."
```

```
                )?;
            }
            e => return Err(anyhow!("Error creating email template: {}", e)),
        },
    }
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK **DeleteContactList**で を使用する

以下のコード例は、DeleteContactList の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ニュースレターのシナリオ

.NET

AWS SDK for .NET

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

```
    /// <summary>
    /// Deletes a contact list and all contacts within it.
    /// </summary>
    /// <param name="contactListName">The name of the contact list to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteContactListAsync(string contactListName)
```

```
    {
        var request = new DeleteContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.DeleteContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The contact list {contactListName} is being
modified by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the contact
list: {ex.Message}");
        }

        return false;
    }
```

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

Java

SDK for Java 2.x

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

```
   try {
     // Delete the contact list
     DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
         .contactListName(CONTACT_LIST_NAME)
         .build();

     sesClient.deleteContactList(deleteContactListRequest);

     System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
   } catch (NotFoundException e) {
     // If the contact list does not exist, log the error and proceed
     System.out.println("Contact list not found. Skipping deletion...");
   } catch (Exception e) {
     System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
     e.printStackTrace();
   }
```

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

Python

SDK for Python (Boto3)

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

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:

 self.ses_client.delete_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' deleted successfully.")
```

```
        except ClientError as e:
            # If the contact list doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
            else:
                print(e)
```

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

Rust

SDK for Rust

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

```
        match self
            .client
            .delete_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list deleted
successfully.")?,
            Err(e) => return Err(anyhow!("Error deleting contact list: {e}")),
        }
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK **DeleteEmailIdentity**で を使用する

以下のコード例は、DeleteEmailIdentity の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ニュースレターのシナリオ

.NET

AWS SDK for .NET

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

```
    /// <summary>
    /// Deletes an email identity (email address or domain).
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailIdentityAsync(string emailIdentity)
    {
        var request = new DeleteEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.DeleteEmailIdentityAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being
 modified by another operation or thread.");
            Console.WriteLine(ex.Message);
```

```
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email
identity: {ex.Message}");
        }

        return false;
    }
```

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

Java

SDK for Java 2.x

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

```
     try {
       // Delete the email identity
       DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
            .emailIdentity(this.verifiedEmail)
```

```
            .build();

        sesClient.deleteEmailIdentity(deleteIdentityRequest);

        System.out.println("Email identity deleted: " + this.verifiedEmail);
      } catch (NotFoundException e) {
        // If the email identity does not exist, log the error and proceed
        System.out.println("Email identity not found. Skipping deletion...");
      } catch (Exception e) {
        System.err.println("Error occurred while deleting the email identity: " +
 e.getMessage());
        e.printStackTrace();
      }
    } else {
      System.out.println("Skipping email identity deletion.");
    }
```

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

Python

SDK for Python (Boto3)

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

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
```

```
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


            try:

 self.ses_client.delete_email_identity(EmailIdentity=self.verified_email)
                print(f"Email identity '{self.verified_email}' deleted
 successfully.")
            except ClientError as e:
                # If the email identity doesn't exist, skip and proceed
                if e.response["Error"]["Code"] == "NotFoundException":
                    print(f"Email identity '{self.verified_email}' does not
 exist.")
                else:
                    print(e)
```

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

Rust

SDK for Rust

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

```
            match self
                .client
                .delete_email_identity()
                .email_identity(self.verified_email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Email identity deleted
successfully.")?,
                Err(e) => {
                    return Err(anyhow!("Error deleting email identity: {}", e));
                }
            }
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK **DeleteEmailTemplate**で を使用する

以下のコード例は、DeleteEmailTemplate の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ニュースレターのシナリオ

.NET

AWS SDK for .NET

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

```
    /// <summary>
    /// Deletes an email template.
    /// </summary>
    /// <param name="templateName">The name of the email template to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
    {
        var request = new DeleteEmailTemplateRequest
        {
            TemplateName = templateName
        };

        try
        {
            var response = await _sesClient.DeleteEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email template {templateName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
```

```
            Console.WriteLine($"An error occurred while deleting the email
template: {ex.Message}");
        }

        return false;
    }
```

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

Java

SDK for Java 2.x

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

```
    try {
      // Delete the template
      DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .build();

      sesClient.deleteEmailTemplate(deleteTemplateRequest);

      System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
      // If the email template does not exist, log the error and proceed
      System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
      e.printStackTrace();
    }
```

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

Python

SDK for Python (Boto3)

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

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep
```

```
        try:
            self.ses_client.delete_email_template(TemplateName=TEMPLATE_NAME)
            print(f"Email template '{TEMPLATE_NAME}' deleted successfully.")
        except ClientError as e:
            # If the email template doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Email template '{TEMPLATE_NAME}' does not exist.")
            else:
                print(e)
```

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

Rust

SDK for Rust

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

```
        match self
            .client
            .delete_email_template()
            .template_name(TEMPLATE_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template deleted
successfully.")?,
            Err(e) => {
                return Err(anyhow!("Error deleting email template: {e}"));
            }
        }
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK **GetEmailIdentity**で を使用する

次のコード例は、GetEmailIdentity を使用する方法を示しています。

Rust

SDK for Rust

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

E メールアドレスが検証されたかどうかを判定します。

```
async fn is_verified(client: &Client, email: &str) -> Result<(), Error> {
    let resp = client
        .get_email_identity()
        .email_identity(email)
        .send()
        .await?;

    if resp.verified_for_sending_status() {
        println!("The address is verified");
    } else {
        println!("The address is not verified");
    }

    Ok(())
}
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK **ListContactLists**で を使用する

次のコード例は、ListContactLists を使用する方法を示しています。

Rust

SDK for Rust

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

```
async fn show_lists(client: &Client) -> Result<(), Error> {
    let resp = client.list_contact_lists().send().await?;

    println!("Contact lists:");

    for list in resp.contact_lists() {
        println!("  {}", list.contact_list_name().unwrap_or_default());
    }

    Ok(())
}
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK **ListContacts**で を使用する

以下のコード例は、ListContacts の使用方法を示しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- ニュースレターのシナリオ

.NET

AWS SDK for .NET

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

```
    /// <summary>
    /// Lists the contacts in the specified contact list.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The list of contacts response from the ListContacts operation.</
returns>
    public async Task<List<Contact>> ListContactsAsync(string contactListName)
    {
        var request = new ListContactsRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.ListContactsAsync(request);
            return response.Contacts;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
```

```
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing the contacts:
{ex.Message}");
        }

        return new List<Contact>();
    }
```

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

Java

SDK for Java 2.x

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

```
    ListContactsRequest contactListRequest = ListContactsRequest.builder()
        .contactListName(CONTACT_LIST_NAME)
        .build();

    List<String> contactEmails;
    try {
      ListContactsResponse contactListResponse =
sesClient.listContacts(contactListRequest);

      contactEmails = contactListResponse.contacts().stream()
          .map(Contact::emailAddress)
          .toList();
    } catch (Exception e) {
      // TODO: Remove when listContacts's GET body issue is resolved.
```

```
        contactEmails = this.contacts;
      }
```

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

Python

SDK for Python (Boto3)

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

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """
```

```
    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:
            contacts_response = self.ses_client.list_contacts(
                ContactListName=CONTACT_LIST_NAME
            )
        except ClientError as e:
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
                return
            else:
                raise e
```

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

Rust

SDK for Rust

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

```
async fn show_contacts(client: &Client, list: &str) -> Result<(), Error> {
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;

    println!("Contacts:");

    for contact in resp.contacts() {
        println!("  {}", contact.email_address().unwrap_or_default());
```

```
    }

    Ok(())
}
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK **SendEmail**で を使用する

以下のコード例は、SendEmail の使用方法を示しています。

.NET

AWS SDK for .NET

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

```
    /// <summary>
    /// Sends an email with the specified content and options.
    /// </summary>
    /// <param name="fromEmailAddress">The email address to send the email
from.</param>
    /// <param name="toEmailAddresses">The email addresses to send the email
to.</param>
    /// <param name="subject">The subject of the email.</param>
    /// <param name="htmlContent">The HTML content of the email.</param>
    /// <param name="textContent">The text content of the email.</param>
    /// <param name="templateName">The name of the email template to use
(optional).</param>
    /// <param name="templateData">The data to replace placeholders in the email
template (optional).</param>
```

```
   /// <param name="contactListName">The name of the contact list for
unsubscribe functionality (optional).</param>
   /// <returns>The MessageId response from the SendEmail operation.</returns>
   public async Task<string> SendEmailAsync(string fromEmailAddress,
List<string> toEmailAddresses, string? subject,
       string? htmlContent, string? textContent, string? templateName = null,
string? templateData = null, string? contactListName = null)
   {
       var request = new SendEmailRequest
       {
           FromEmailAddress = fromEmailAddress
       };

       if (toEmailAddresses.Any())
       {
           request.Destination = new Destination { ToAddresses =
toEmailAddresses };
       }

       if (!string.IsNullOrEmpty(templateName))
       {
           request.Content = new EmailContent()
           {
               Template = new Template
               {
                   TemplateName = templateName,
                   TemplateData = templateData
               }
           };
       }
       else
       {
           request.Content = new EmailContent
           {
               Simple = new Message
               {
                   Subject = new Content { Data = subject },
                   Body = new Body
                   {
                       Html = new Content { Data = htmlContent },
                       Text = new Content { Data = textContent }
                   }
               }
           };
```

```
        }

        if (!string.IsNullOrEmpty(contactListName))
        {
            request.ListManagementOptions = new ListManagementOptions
            {
                ContactListName = contactListName
            };
        }

        try
        {
            var response = await _sesClient.SendEmailAsync(request);
            return response.MessageId;
        }
        catch (AccountSuspendedException ex)
        {
            Console.WriteLine("The account's ability to send email has been
permanently restricted.");
            Console.WriteLine(ex.Message);
        }
        catch (MailFromDomainNotVerifiedException ex)
        {
            Console.WriteLine("The sending domain is not verified.");
            Console.WriteLine(ex.Message);
        }
        catch (MessageRejectedException ex)
        {
            Console.WriteLine("The message content is invalid.");
            Console.WriteLine(ex.Message);
        }
        catch (SendingPausedException ex)
        {
            Console.WriteLine("The account's ability to send email is currently
paused.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine($"An error occurred while sending the email:
 {ex.Message}");
        }

        return string.Empty;
    }
```

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

Java

SDK for Java 2.x

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

メッセージを送信します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */

public class SendEmail {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                    <sender> <recipient> <subject>\s

                                Where:
                                    sender - An email address that represents the
 sender.\s
                                    recipient - An email address that represents
 the recipient.\s
                                    subject - The subject line.\s
                                """;

                if (args.length != 3) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String sender = args[0];
                String recipient = args[1];
                String subject = args[2];

                Region region = Region.US_EAST_1;
                SesV2Client sesv2Client = SesV2Client.builder()
                                .region(region)
                                .build();

                // The HTML body of the email.
                String bodyHTML = "<html>" + "<head></head>" + "<body>" +
 "<h1>Hello!</h1>"
                                + "<p> See the list of customers.</p>" + "</
body>" + "</html>";

                send(sesv2Client, sender, recipient, subject, bodyHTML);
        }

        public static void send(SesV2Client client,
                        String sender,
                        String recipient,
                        String subject,
```

```
                        String bodyHTML) {

                Destination destination = Destination.builder()
                                .toAddresses(recipient)
                                .build();

                Content content = Content.builder()
                                .data(bodyHTML)
                                .build();

                Content sub = Content.builder()
                                .data(subject)
                                .build();

                Body body = Body.builder()
                                .html(content)
                                .build();

                Message msg = Message.builder()
                                .subject(sub)
                                .body(body)
                                .build();

                EmailContent emailContent = EmailContent.builder()
                                .simple(msg)
                                .build();

                SendEmailRequest emailRequest = SendEmailRequest.builder()
                                .destination(destination)
                                .content(emailContent)
                                .fromEmailAddress(sender)
                                .build();

                try {
                        System.out.println("Attempting to send an email through
Amazon SES "
                                        + "using the AWS SDK for Java...");
                        client.sendEmail(emailRequest);
                        System.out.println("email was sent");

                } catch (SesV2Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
```

```
        }
}
```

テンプレートを使用してメッセージを送信します。

```
      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
                    .templateName(TEMPLATE_NAME)
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
      }
```

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

Python

SDK for Python (Boto3)

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

連絡先リストのすべてのメンバーにメッセージを送信します。

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons
 Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
```

```
                    },
                )
                print(f"Welcome email sent to '{email}'.")
```

テンプレートを使用して、連絡先リストのメンバー全員にメッセージを送信します。

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email_address]},
                    Content={
                        "Template": {
                            "TemplateName": TEMPLATE_NAME,
                            "TemplateData": coupon_items,
                        }
```

```
                    },
                    ListManagementOptions={"ContactListName": CONTACT_LIST_NAME},
                )
```

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

Ruby

SDK for Ruby

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

```
require 'aws-sdk-sesv2'
require_relative 'config' # Recipient and sender email addresses.

# Set up the SESv2 client.
client = Aws::SESV2::Client.new(region: AWS_REGION)

def send_email(client, sender_email, recipient_email)
  response = client.send_email(
    {
      from_email_address: sender_email,
      destination: {
        to_addresses: [recipient_email]
      },
      content: {
        simple: {
          subject: {
            data: 'Test email subject'
          },
          body: {
            text: {
              data: 'Test email body'
            }
          }
```

```
        }
      }
    }
  )
  puts "Email sent from #{SENDER_EMAIL} to #{RECIPIENT_EMAIL} with message ID:
 #{response.message_id}"
end

send_email(client, SENDER_EMAIL, RECIPIENT_EMAIL)
```

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

Rust

SDK for Rust

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

連絡先リストのすべてのメンバーにメッセージを送信します。

```
async fn send_message(
    client: &Client,
    list: &str,
    from: &str,
    subject: &str,
    message: &str,
) -> Result<(), Error> {
    // Get list of email addresses from contact list.
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;

    let contacts = resp.contacts();
```

```
    let cs: Vec<String> = contacts
        .iter()
        .map(|i| i.email_address().unwrap_or_default().to_string())
        .collect();

    let mut dest: Destination = Destination::builder().build();
    dest.to_addresses = Some(cs);
    let subject_content = Content::builder()
        .data(subject)
        .charset("UTF-8")
        .build()
        .expect("building Content");
    let body_content = Content::builder()
        .data(message)
        .charset("UTF-8")
        .build()
        .expect("building Content");
    let body = Body::builder().text(body_content).build();

    let msg = Message::builder()
        .subject(subject_content)
        .body(body)
        .build();

    let email_content = EmailContent::builder().simple(msg).build();

    client
        .send_email()
        .from_email_address(from)
        .destination(dest)
        .content(email_content)
        .send()
        .await?;

    println!("Email sent to list");

    Ok(())
}
```

テンプレートを使用して、連絡先リストのメンバー全員にメッセージを送信します。

```
            let coupons = std::fs::read_to_string("../resources/newsletter/
sample_coupons.json")
                .unwrap_or_else(|_| r#"{"coupons":[]}"#.to_string());
            let email_content = EmailContent::builder()
                .template(
                    Template::builder()
                        .template_name(TEMPLATE_NAME)
                        .template_data(coupons)
                        .build(),
                )
                .build();

            match self
                .client
                .send_email()
                .from_email_address(self.verified_email.clone())

 .destination(Destination::builder().to_addresses(email.clone()).build())
                .content(email_content)
                .list_management_options(
                    ListManagementOptions::builder()
                        .contact_list_name(CONTACT_LIST_NAME)
                        .build()?,
                )
                .send()
                .await
            {
                Ok(output) => {
                    if let Some(message_id) = output.message_id {
                        writeln!(
                            self.stdout,
                            "Newsletter sent to {} with message ID {}",
                            email, message_id
                        )?;
                    } else {
                        writeln!(self.stdout, "Newsletter sent to {}", email)?;
                    }
                }
                Err(e) => return Err(anyhow!("Error sending newsletter to {}:
 {}", email, e)),
            }
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK を使用した Amazon SES API v2 のシナリオ AWS SDKs

次のコード例は、 AWS SDKs を使用して Amazon SES API v2 で一般的なシナリオを実装する方法を示しています。これらのシナリオでは、Amazon SES API v2 内で複数の関数を呼び出すか、その他の AWS のサービスと連携して特定のタスクを実行する方法を説明しています。各シナリオには、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

シナリオは、サービスアクションをコンテキストで理解するのに役立つ中級レベルの経験を対象としています。

例

- AWS SDK を使用した完全な Amazon SES API v2 ニュースレターシナリオ

AWS SDK を使用した完全な Amazon SES API v2 ニュースレターシナリオ

次のコード例は、Amazon SES API v2 ニュースレターシナリオを実行する方法を示しています。

.NET

AWS SDK for .NET

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

シナリオを実行します。

```
using System.Diagnostics;
using System.Text.RegularExpressions;
```

```
using Amazon.SimpleEmailV2;
using Amazon.SimpleEmailV2.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace Sesv2Scenario;

public static class NewsletterWorkflow
{
    /*
      This scenario demonstrates how to use the Amazon Simple Email Service (SES)
 v2 to send a coupon newsletter to a list of subscribers.
      The scenario performs the following tasks:

      1. Prepare the application:
         - Create a verified email identity for sending and replying to emails.
         - Create a contact list to store the subscribers' email addresses.
         - Create an email template for the coupon newsletter.

      2. Gather subscriber email addresses:
         - Prompt the user for a base email address.
         - Create 3 variants of the email address using subaddress extensions
 (e.g., user+ses-weekly-newsletter-1@example.com).
         - Add each variant as a contact to the contact list.
         - Send a welcome email to each new contact.

      3. Send the coupon newsletter:
         - Retrieve the list of contacts from the contact list.
         - Send the coupon newsletter using the email template to each contact.

      4. Monitor and review:
         - Provide instructions for the user to review the sending activity and
 metrics in the AWS console.

      5. Clean up resources:
         - Delete the contact list (which also deletes all contacts within it).
         - Delete the email template.
         - Optionally delete the verified email identity.

    */
```

```
    public static SESv2Wrapper _sesv2Wrapper;
    public static string? _baseEmailAddress = null;
    public static string? _verifiedEmail = null;
    private static string _contactListName = "weekly-coupons-newsletter";
    private static string _templateName = "weekly-coupons";
    private static string _subject = "Weekly Coupons Newsletter";
    private static string _htmlContentFile = "coupon-newsletter.html";
    private static string _textContentFile = "coupon-newsletter.txt";
    private static string _htmlWelcomeFile = "welcome.html";
    private static string _textWelcomeFile = "welcome.txt";
    private static string _couponsDataFile = "sample_coupons.json";

    // Relative location of the resources folder.
    private static string _resourcesFilePathLocation = "../../../../resources/";

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSimpleEmailServiceV2>()
                    .AddTransient<SESv2Wrapper>()
            )
            .Build();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon SES v2 Coupon Newsletter
Scenario.");
            Console.WriteLine("This scenario demonstrates how to use the Amazon
Simple Email Service (SES) v2 " +
                              "\r\nto send a coupon newsletter to a list of
subscribers.");
```

```
            // Prepare the application.
            var emailIdentity = await PrepareApplication();

            // Gather subscriber email addresses.
            await GatherSubscriberEmailAddresses(emailIdentity);

            // Send the coupon newsletter.
            await SendCouponNewsletter(emailIdentity);

            // Monitor and review.
            MonitorAndReview(true);

            // Clean up resources.
            await Cleanup(emailIdentity, true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon SES v2 Coupon Newsletter scenario is
complete.");
            Console.WriteLine(new string('-', 80));
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _sesv2Wrapper = host.Services.GetRequiredService<SESv2Wrapper>();
    }

    /// <summary>
    /// Set up the resources for the scenario.
    /// </summary>
    /// <returns>The email address of the verified identity.</returns>
    public static async Task<string?> PrepareApplication()
    {
        var htmlContent = await File.ReadAllTextAsync(_resourcesFilePathLocation
+ _htmlContentFile);
```

```
        var textContent = await File.ReadAllTextAsync(_resourcesFilePathLocation
+ _textContentFile);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("1. In this step, we will prepare the application:" +
                          "\r\n  - Create a verified email identity for sending
and replying to emails." +
                          "\r\n  - Create a contact list to store the
subscribers' email addresses." +
                          "\r\n  - Create an email template for the coupon
newsletter.\r\n");

        // Prompt the user for a verified email address.
        while (!IsEmail(_verifiedEmail))
        {
            Console.Write("Enter a verified email address or an email to verify:
");
            _verifiedEmail = Console.ReadLine();
        }

        try
        {
            // Create an email identity and start the verification process.
            await _sesv2Wrapper.CreateEmailIdentityAsync(_verifiedEmail);
            Console.WriteLine($"Identity {_verifiedEmail} created.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Identity {_verifiedEmail} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating email identity: {ex.Message}");
        }

        // Create a contact list.
        try
        {
            await _sesv2Wrapper.CreateContactListAsync(_contactListName);
            Console.WriteLine($"Contact list {_contactListName} created.");
        }
        catch (AlreadyExistsException)
        {
```

```
            Console.WriteLine($"Contact list {_contactListName} already
exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating contact list: {ex.Message}");
        }

        // Create an email template.
        try
        {
            await _sesv2Wrapper.CreateEmailTemplateAsync(_templateName, _subject,
htmlContent, textContent);
            Console.WriteLine($"Email template {_templateName} created.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine($"Email template {_templateName} already exists.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error creating email template: {ex.Message}");
        }

        return _verifiedEmail;
    }

    /// <summary>
    /// Generate subscriber addresses and send welcome emails.
    /// </summary>
    /// <param name="fromEmailAddress">The verified email address from
PrepareApplication.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> GatherSubscriberEmailAddresses(string
fromEmailAddress)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("2. In Step 2, we will gather subscriber email
addresses:" +
                          "\r\n  - Prompt the user for a base email address." +
                          "\r\n  - Create 3 variants of the email address using
subaddress extensions (e.g., user+ses-weekly-newsletter-1@example.com)." +
                          "\r\n  - Add each variant as a contact to the contact
list." +
```

```
                        "\r\n  - Send a welcome email to each new contact.\r
\n");

        // Prompt the user for a base email address.
        while (!IsEmail(_baseEmailAddress))
        {
            Console.Write("Enter a base email address (e.g., user@example.com):
 ");
            _baseEmailAddress = Console.ReadLine();
        }

        // Create 3 variants of the email address using +ses-weekly-newsletter-1,
 +ses-weekly-newsletter-2, etc.
        var baseEmailAddressParts = _baseEmailAddress!.Split("@");
        for (int i = 1; i <= 3; i++)
        {
            string emailAddress = $"{baseEmailAddressParts[0]}+ses-weekly-
newsletter-{i}@{baseEmailAddressParts[1]}";

            try
            {
                // Create a contact with the email address in the contact list.
                await _sesv2Wrapper.CreateContactAsync(emailAddress,
 _contactListName);
                Console.WriteLine($"Contact {emailAddress} added to the
 {_contactListName} contact list.");
            }
            catch (AlreadyExistsException)
            {
                Console.WriteLine($"Contact {emailAddress} already exists in the
 {_contactListName} contact list.");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error creating contact {emailAddress}:
 {ex.Message}");
                return false;
            }

            // Send a welcome email to the new contact.
            try
            {
                string subject = "Welcome to the Weekly Coupons Newsletter";
```

```
                string htmlContent = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _htmlWelcomeFile);
                string textContent = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _textWelcomeFile);

                await _sesv2Wrapper.SendEmailAsync(fromEmailAddress, new
List<string> { emailAddress }, subject, htmlContent, textContent);
                Console.WriteLine($"Welcome email sent to {emailAddress}.");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending welcome email to
{emailAddress}: {ex.Message}");
                return false;
            }

            // Wait 2 seconds before sending the next email (if the account is in
the SES Sandbox).
            await Task.Delay(2000);
        }

        return true;
    }

    /// <summary>
    ///  Send the coupon newsletter to the subscribers in the contact list.
    /// </summary>
    /// <param name="fromEmailAddress">The verified email address from
PrepareApplication.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> SendCouponNewsletter(string fromEmailAddress)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("3. In this step, we will send the coupon newsletter:"
+
                          "\r\n  - Retrieve the list of contacts from the contact
list." +
                          "\r\n  - Send the coupon newsletter using the email
template to each contact.\r\n");


        // Retrieve the list of contacts from the contact list.
        var contacts = await _sesv2Wrapper.ListContactsAsync(_contactListName);
        if (!contacts.Any())
```

```
        {
            Console.WriteLine($"No contacts found in the {_contactListName}
contact list.");
            return false;
        }

        // Load the coupon data from the sample_coupons.json file.
        string couponsData = await
File.ReadAllTextAsync(_resourcesFilePathLocation + _couponsDataFile);

        // Send the coupon newsletter to each contact using the email template.
        try
        {
            foreach (var contact in contacts)
            {
                // To use the Contact List for list management, send to only one
address at a time.
                await _sesv2Wrapper.SendEmailAsync(fromEmailAddress,
                    new List<string> { contact.EmailAddress },
                    null, null, null, _templateName, couponsData,
_contactListName);
            }

            Console.WriteLine($"Coupon newsletter sent to contact list
{_contactListName}.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error sending coupon newsletter to contact list
{_contactListName}: {ex.Message}");
            return false;
        }

        return true;
    }

    /// <summary>
    /// Provide instructions for monitoring sending activity and metrics.
    /// </summary>
    /// <param name="interactive">True to run in interactive mode.</param>
    /// <returns>True if successful.</returns>
    public static bool MonitorAndReview(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine("4. In step 4, we will monitor and review:" +
                          "\r\n  - Provide instructions for the user to review
the sending activity and metrics in the AWS console.\r\n");

        Console.WriteLine("Review your sending activity using the SES Homepage in
the AWS console.");
        Console.WriteLine("Press Enter to open the SES Homepage in your default
browser...");
        if (interactive)
        {
            Console.ReadLine();
            try
            {
                // Open the SES Homepage in the default browser.
                Process.Start(new ProcessStartInfo
                {
                    FileName = "https://console.aws.amazon.com/ses/home",
                    UseShellExecute = true
                });
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error opening the SES Homepage:
{ex.Message}");
                return false;
            }
        }

        Console.WriteLine("Review the sending activity and email metrics, then
press Enter to continue...");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Clean up the resources used in the scenario.
    /// </summary>
    /// <param name="verifiedEmailAddress">The verified email address from
PrepareApplication.</param>
    /// <param name="interactive">True if interactive.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Cleanup(string verifiedEmailAddress, bool
interactive)
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("5. Finally, we clean up resources:" +
                          "\r\n  - Delete the contact list (which also deletes
 all contacts within it)." +
                          "\r\n  - Delete the email template." +
                          "\r\n  - Optionally delete the verified email identity.
\r\n");

        Console.WriteLine("Cleaning up resources...");

        // Delete the contact list (this also deletes all contacts in the list).
        try
        {
            await _sesv2Wrapper.DeleteContactListAsync(_contactListName);
            Console.WriteLine($"Contact list {_contactListName} deleted.");
        }
        catch (NotFoundException)
        {
            Console.WriteLine($"Contact list {_contactListName} not found.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error deleting contact list {_contactListName}:
 {ex.Message}");
            return false;
        }

        // Delete the email template.
        try
        {
            await _sesv2Wrapper.DeleteEmailTemplateAsync(_templateName);
            Console.WriteLine($"Email template {_templateName} deleted.");
        }
        catch (NotFoundException)
        {
            Console.WriteLine($"Email template {_templateName} not found.");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error deleting email template {_templateName}:
 {ex.Message}");
            return false;
        }
```

```
        // Ask the user if they want to delete the email identity.
        var deleteIdentity = !interactive ||
            GetYesNoResponse(
                $"Do you want to delete the email identity
 {verifiedEmailAddress}? (y/n) ");
        if (deleteIdentity)
        {
            try
            {
                await
 _sesv2Wrapper.DeleteEmailIdentityAsync(verifiedEmailAddress);
                Console.WriteLine($"Email identity {verifiedEmailAddress}
 deleted.");
            }
            catch (NotFoundException)
            {
                Console.WriteLine(
                    $"Email identity {verifiedEmailAddress} not found.");
            }
            catch (Exception ex)
            {
                Console.WriteLine(
                    $"Error deleting email identity {verifiedEmailAddress}:
 {ex.Message}");
                return false;
            }
        }
        else
        {
            Console.WriteLine(
                $"Skipping deletion of email identity {verifiedEmailAddress}.");
        }

        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
```

```
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Simple check to verify a string is an email address.
    /// </summary>
    /// <param name="email">The string to verify.</param>
    /// <returns>True if a valid email.</returns>
    private static bool IsEmail(string? email)
    {
        if (string.IsNullOrEmpty(email))
            return false;
        return Regex.IsMatch(email, @"^[^@\s]+@[^@\s]+\.[^@\s]+$",
 RegexOptions.IgnoreCase);
    }
}
```

サービスオペレーション用のラッパー

```
using System.Net;
using Amazon.SimpleEmailV2;
using Amazon.SimpleEmailV2.Model;

namespace Sesv2Scenario;

/// <summary>
/// Wrapper class for Amazon Simple Email Service (SES) v2 operations.
/// </summary>
public class SESv2Wrapper
{

    private readonly IAmazonSimpleEmailServiceV2 _sesClient;

    /// <summary>
    /// Constructor for the SESv2Wrapper.
    /// </summary>
    /// <param name="sesClient">The injected SES v2 client.</param>
```

```
    public SESv2Wrapper(IAmazonSimpleEmailServiceV2 sesClient)
    {
        _sesClient = sesClient;
    }

    /// <summary>
    /// Creates a contact and adds it to the specified contact list.
    /// </summary>
    /// <param name="emailAddress">The email address of the contact.</param>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The response from the CreateContact operation.</returns>
    public async Task<bool> CreateContactAsync(string emailAddress, string
contactListName)
    {
        var request = new CreateContactRequest
        {
            EmailAddress = emailAddress,
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact with email address {emailAddress} already
exists in the contact list {contactListName}.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
```

```
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the contact:
{ex.Message}");
        }
        return false;
    }

    /// <summary>
    /// Creates a contact list with the specified name.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateContactListAsync(string contactListName)
    {
        var request = new CreateContactListRequest
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.CreateContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Contact list with name {contactListName} already
exists.");
            Console.WriteLine(ex.Message);
            return true;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for contact lists has been exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
```

```
        {
            Console.WriteLine($"An error occurred while creating the contact
list: {ex.Message}");
        }
        return false;
    }

    /// <summary>
    /// Creates an email identity (email address or domain) and starts the
verification process.
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to create and
verify.</param>
    /// <returns>The response from the CreateEmailIdentity operation.</returns>
    public async Task<CreateEmailIdentityResponse>
CreateEmailIdentityAsync(string emailIdentity)
    {
        var request = new CreateEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.CreateEmailIdentityAsync(request);
            return response;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email identity {emailIdentity} already exists.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being
modified by another operation or thread.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email identities has been
exceeded.");
```

```
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
exist.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
            throw;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email
identity: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Creates an email template with the specified content.
    /// </summary>
    /// <param name="templateName">The name of the email template.</param>
    /// <param name="subject">The subject of the email template.</param>
    /// <param name="htmlContent">The HTML content of the email template.</param>
    /// <param name="textContent">The text content of the email template.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateEmailTemplateAsync(string templateName, string
subject, string htmlContent, string textContent)
    {
        var request = new CreateEmailTemplateRequest
        {
            TemplateName = templateName,
            TemplateContent = new EmailTemplateContent
            {
                Subject = subject,
                Html = htmlContent,
                Text = textContent
```

```
            }
        };

        try
        {
            var response = await _sesClient.CreateEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AlreadyExistsException ex)
        {
            Console.WriteLine($"Email template with name {templateName} already
 exists.");
            Console.WriteLine(ex.Message);
        }
        catch (LimitExceededException ex)
        {
            Console.WriteLine("The limit for email templates has been
 exceeded.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while creating the email
 template: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Deletes a contact list and all contacts within it.
    /// </summary>
    /// <param name="contactListName">The name of the contact list to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteContactListAsync(string contactListName)
    {
        var request = new DeleteContactListRequest
```

```
        {
            ContactListName = contactListName
        };

        try
        {
            var response = await _sesClient.DeleteContactListAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The contact list {contactListName} is being
 modified by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the contact
 list: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Deletes an email identity (email address or domain).
    /// </summary>
    /// <param name="emailIdentity">The email address or domain to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailIdentityAsync(string emailIdentity)
    {
```

```
        var request = new DeleteEmailIdentityRequest
        {
            EmailIdentity = emailIdentity
        };

        try
        {
            var response = await _sesClient.DeleteEmailIdentityAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (ConcurrentModificationException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} is being
 modified by another operation or thread.");
            Console.WriteLine(ex.Message);
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email identity {emailIdentity} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email
 identity: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Deletes an email template.
    /// </summary>
    /// <param name="templateName">The name of the email template to delete.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteEmailTemplateAsync(string templateName)
```

```
    {
        var request = new DeleteEmailTemplateRequest
        {
            TemplateName = templateName
        };

        try
        {
            var response = await _sesClient.DeleteEmailTemplateAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The email template {templateName} does not
 exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
 later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while deleting the email
 template: {ex.Message}");
        }

        return false;
    }

    /// <summary>
    /// Lists the contacts in the specified contact list.
    /// </summary>
    /// <param name="contactListName">The name of the contact list.</param>
    /// <returns>The list of contacts response from the ListContacts operation.</
returns>
    public async Task<List<Contact>> ListContactsAsync(string contactListName)
    {
        var request = new ListContactsRequest
        {
            ContactListName = contactListName
        };
```

```
        try
        {
            var response = await _sesClient.ListContactsAsync(request);
            return response.Contacts;
        }
        catch (NotFoundException ex)
        {
            Console.WriteLine($"The contact list {contactListName} does not
exist.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while listing the contacts:
{ex.Message}");
        }

        return new List<Contact>();
    }

    /// <summary>
    /// Sends an email with the specified content and options.
    /// </summary>
    /// <param name="fromEmailAddress">The email address to send the email
from.</param>
    /// <param name="toEmailAddresses">The email addresses to send the email
to.</param>
    /// <param name="subject">The subject of the email.</param>
    /// <param name="htmlContent">The HTML content of the email.</param>
    /// <param name="textContent">The text content of the email.</param>
    /// <param name="templateName">The name of the email template to use
(optional).</param>
    /// <param name="templateData">The data to replace placeholders in the email
template (optional).</param>
    /// <param name="contactListName">The name of the contact list for
unsubscribe functionality (optional).</param>
    /// <returns>The MessageId response from the SendEmail operation.</returns>
```

```
   public async Task<string> SendEmailAsync(string fromEmailAddress,
List<string> toEmailAddresses, string? subject,
       string? htmlContent, string? textContent, string? templateName = null,
string? templateData = null, string? contactListName = null)
   {
       var request = new SendEmailRequest
       {
           FromEmailAddress = fromEmailAddress
       };

       if (toEmailAddresses.Any())
       {
           request.Destination = new Destination { ToAddresses =
toEmailAddresses };
       }

       if (!string.IsNullOrEmpty(templateName))
       {
           request.Content = new EmailContent()
           {
               Template = new Template
               {
                   TemplateName = templateName,
                   TemplateData = templateData
               }
           };
       }
       else
       {
           request.Content = new EmailContent
           {
               Simple = new Message
               {
                   Subject = new Content { Data = subject },
                   Body = new Body
                   {
                       Html = new Content { Data = htmlContent },
                       Text = new Content { Data = textContent }
                   }
               }
           };
       }

       if (!string.IsNullOrEmpty(contactListName))
```

```
        {
            request.ListManagementOptions = new ListManagementOptions
            {
                ContactListName = contactListName
            };
        }

        try
        {
            var response = await _sesClient.SendEmailAsync(request);
            return response.MessageId;
        }
        catch (AccountSuspendedException ex)
        {
            Console.WriteLine("The account's ability to send email has been
permanently restricted.");
            Console.WriteLine(ex.Message);
        }
        catch (MailFromDomainNotVerifiedException ex)
        {
            Console.WriteLine("The sending domain is not verified.");
            Console.WriteLine(ex.Message);
        }
        catch (MessageRejectedException ex)
        {
            Console.WriteLine("The message content is invalid.");
            Console.WriteLine(ex.Message);
        }
        catch (SendingPausedException ex)
        {
            Console.WriteLine("The account's ability to send email is currently
paused.");
            Console.WriteLine(ex.Message);
        }
        catch (TooManyRequestsException ex)
        {
            Console.WriteLine("Too many requests were made. Please try again
later.");
            Console.WriteLine(ex.Message);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred while sending the email:
{ex.Message}");
```

```
        }

        return string.Empty;
    }
}
```

- API の詳細については、『AWS SDK for .NET API リファレンス』の以下のトピックを参照してください。
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Java

SDK for Java 2.x

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

```
    try {
      // 2. Create a contact list
      String contactListName = CONTACT_LIST_NAME;
      CreateContactListRequest createContactListRequest =
CreateContactListRequest.builder()
          .contactListName(contactListName)
          .build();
```

```
      sesClient.createContactList(createContactListRequest);
      System.out.println("Contact list created: " + contactListName);
    } catch (AlreadyExistsException e) {
      System.out.println("Contact list already exists, skipping creation: weekly-
coupons-newsletter");
    } catch (LimitExceededException e) {
      System.err.println("Limit for contact lists has been exceeded.");
      throw e;
    } catch (SesV2Exception e) {
      System.err.println("Error creating contact list: " + e.getMessage());
      throw e;
    }

      try {
        // Create a new contact with the provided email address in the
        CreateContactRequest contactRequest = CreateContactRequest.builder()
            .contactListName(CONTACT_LIST_NAME)
            .emailAddress(emailAddress)
            .build();

        sesClient.createContact(contactRequest);
        contacts.add(emailAddress);

        System.out.println("Contact created: " + emailAddress);

        // Send a welcome email to the new contact
        String welcomeHtml = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.html"));
        String welcomeText = Files.readString(Paths.get("resources/
coupon_newsletter/welcome.txt"));

        SendEmailRequest welcomeEmailRequest = SendEmailRequest.builder()
            .fromEmailAddress(this.verifiedEmail)
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .simple(
                    Message.builder()
                        .subject(Content.builder().data("Welcome to the Weekly
 Coupons Newsletter").build())
                        .body(Body.builder()
                            .text(Content.builder().data(welcomeText).build())
                            .html(Content.builder().data(welcomeHtml).build())
                            .build())
                        .build())
```

```
                .build())
            .build();
        SendEmailResponse welcomeEmailResponse =
 sesClient.sendEmail(welcomeEmailRequest);
        System.out.println("Welcome email sent: " +
 welcomeEmailResponse.messageId());
      } catch (AlreadyExistsException e) {
        // If the contact already exists, skip this step for that contact and
 proceed
        // with the next contact
        System.out.println("Contact already exists, skipping creation...");
      } catch (Exception e) {
        System.err.println("Error occurred while processing email address " +
 emailAddress + ": " + e.getMessage());
        throw e;
      }
    }

      ListContactsRequest contactListRequest = ListContactsRequest.builder()
          .contactListName(CONTACT_LIST_NAME)
          .build();

      List<String> contactEmails;
      try {
        ListContactsResponse contactListResponse =
 sesClient.listContacts(contactListRequest);

        contactEmails = contactListResponse.contacts().stream()
            .map(Contact::emailAddress)
            .toList();
      } catch (Exception e) {
        // TODO: Remove when listContacts's GET body issue is resolved.
        contactEmails = this.contacts;
      }


      String coupons = Files.readString(Paths.get("resources/coupon_newsletter/
sample_coupons.json"));
      for (String emailAddress : contactEmails) {
        SendEmailRequest newsletterRequest = SendEmailRequest.builder()
            .destination(Destination.builder().toAddresses(emailAddress).build())
            .content(EmailContent.builder()
                .template(Template.builder()
                    .templateName(TEMPLATE_NAME)
```

```
                    .templateData(coupons)
                    .build())
                .build())
            .fromEmailAddress(this.verifiedEmail)
            .listManagementOptions(ListManagementOptions.builder()
                .contactListName(CONTACT_LIST_NAME)
                .build())
            .build();
        SendEmailResponse newsletterResponse =
 sesClient.sendEmail(newsletterRequest);
        System.out.println("Newsletter sent to " + emailAddress + ": " +
 newsletterResponse.messageId());
     }

   try {
     CreateEmailIdentityRequest createEmailIdentityRequest =
 CreateEmailIdentityRequest.builder()
         .emailIdentity(verifiedEmail)
         .build();
     sesClient.createEmailIdentity(createEmailIdentityRequest);
     System.out.println("Email identity created: " + verifiedEmail);
   } catch (AlreadyExistsException e) {
     System.out.println("Email identity already exists, skipping creation: " +
 verifiedEmail);
   } catch (NotFoundException e) {
     System.err.println("The provided email address is not verified: " +
 verifiedEmail);
     throw e;
   } catch (LimitExceededException e) {
     System.err
         .println("You have reached the limit for email identities. Please
 remove some identities and try again.");
     throw e;
   } catch (SesV2Exception e) {
     System.err.println("Error creating email identity: " + e.getMessage());
     throw e;
   }

   try {
     // Create an email template named "weekly-coupons"
     String newsletterHtml = loadFile("resources/coupon_newsletter/coupon-
newsletter.html");
     String newsletterText = loadFile("resources/coupon_newsletter/coupon-
newsletter.txt");
```

```
     CreateEmailTemplateRequest templateRequest =
CreateEmailTemplateRequest.builder()
         .templateName(TEMPLATE_NAME)
         .templateContent(EmailTemplateContent.builder()
             .subject("Weekly Coupons Newsletter")
             .html(newsletterHtml)
             .text(newsletterText)
             .build())
         .build();

     sesClient.createEmailTemplate(templateRequest);

     System.out.println("Email template created: " + TEMPLATE_NAME);
   } catch (AlreadyExistsException e) {
     // If the template already exists, skip this step and proceed with the next
     // operation
     System.out.println("Email template already exists, skipping creation...");
   } catch (LimitExceededException e) {
     // If the limit for email templates is exceeded, fail the workflow and
inform
     // the user
     System.err.println("You have reached the limit for email templates. Please
remove some templates and try again.");
     throw e;
   } catch (Exception e) {
     System.err.println("Error occurred while creating email template: " +
e.getMessage());
     throw e;
   }

   try {
     // Delete the contact list
     DeleteContactListRequest deleteContactListRequest =
DeleteContactListRequest.builder()
         .contactListName(CONTACT_LIST_NAME)
         .build();

     sesClient.deleteContactList(deleteContactListRequest);

     System.out.println("Contact list deleted: " + CONTACT_LIST_NAME);
   } catch (NotFoundException e) {
     // If the contact list does not exist, log the error and proceed
     System.out.println("Contact list not found. Skipping deletion...");
```

```
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the contact list: " +
e.getMessage());
      e.printStackTrace();
    }

      try {
        // Delete the email identity
        DeleteEmailIdentityRequest deleteIdentityRequest =
DeleteEmailIdentityRequest.builder()
            .emailIdentity(this.verifiedEmail)
            .build();

        sesClient.deleteEmailIdentity(deleteIdentityRequest);

        System.out.println("Email identity deleted: " + this.verifiedEmail);
      } catch (NotFoundException e) {
        // If the email identity does not exist, log the error and proceed
        System.out.println("Email identity not found. Skipping deletion...");
      } catch (Exception e) {
        System.err.println("Error occurred while deleting the email identity: " +
e.getMessage());
        e.printStackTrace();
      }
    } else {
      System.out.println("Skipping email identity deletion.");
    }

    try {
      // Delete the template
      DeleteEmailTemplateRequest deleteTemplateRequest =
DeleteEmailTemplateRequest.builder()
          .templateName(TEMPLATE_NAME)
          .build();

      sesClient.deleteEmailTemplate(deleteTemplateRequest);

      System.out.println("Email template deleted: " + TEMPLATE_NAME);
    } catch (NotFoundException e) {
      // If the email template does not exist, log the error and proceed
      System.out.println("Email template not found. Skipping deletion...");
    } catch (Exception e) {
      System.err.println("Error occurred while deleting the email template: " +
e.getMessage());
```

```
            e.printStackTrace();
        }
```

- API の詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Python

SDK for Python (Boto3)

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

```
def main():
    """
    The main function that orchestrates the execution of the workflow.
    """
    print(INTRO)
    ses_client = boto3.client("sesv2")
    workflow = SESv2Workflow(ses_client)
    try:
        workflow.prepare_application()
        workflow.gather_subscriber_email_addresses()
```

```
        workflow.send_coupon_newsletter()
        workflow.monitor_and_review()
    except ClientError as e:
        print_error(e)
    workflow.clean_up()



class SESv2Workflow:
    """
    A class to manage the SES v2 Coupon Newsletter Workflow.
    """

    def __init__(self, ses_client, sleep=True):
        self.ses_client = ses_client
        self.sleep = sleep


        try:

 self.ses_client.create_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' created successfully.")
        except ClientError as e:
            # If the contact list already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Contact list '{CONTACT_LIST_NAME}' already exists.")
            else:
                raise e

            try:
                # Create a new contact
                self.ses_client.create_contact(
                    ContactListName=CONTACT_LIST_NAME, EmailAddress=email
                )
                print(f"Contact with email '{email}' created successfully.")

                # Send the welcome email
                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
```

```
                                "Data": "Welcome to the Weekly Coupons
Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")
                if self.sleep:
                    # 1 email per second in sandbox mode, remove in production.
                    sleep(1.1)
            except ClientError as e:
                # If the contact already exists, skip and proceed
                if e.response["Error"]["Code"] == "AlreadyExistsException":
                    print(f"Contact with email '{email}' already exists.
Skipping...")
                else:
                    raise e

        try:
            contacts_response = self.ses_client.list_contacts(
                ContactListName=CONTACT_LIST_NAME
            )
        except ClientError as e:
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
                return
            else:
                raise e

                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email]},
                    Content={
                        "Simple": {
                            "Subject": {
                                "Data": "Welcome to the Weekly Coupons
Newsletter"
                            },
                            "Body": {
                                "Text": {"Data": welcome_text},
```

```
                                "Html": {"Data": welcome_html},
                            },
                        }
                    },
                )
                print(f"Welcome email sent to '{email}'.")

                self.ses_client.send_email(
                    FromEmailAddress=self.verified_email,
                    Destination={"ToAddresses": [email_address]},
                    Content={
                        "Template": {
                            "TemplateName": TEMPLATE_NAME,
                            "TemplateData": coupon_items,
                        }
                    },
                    ListManagementOptions={"ContactListName": CONTACT_LIST_NAME},
                )

        try:

self.ses_client.create_email_identity(EmailIdentity=self.verified_email)
            print(f"Email identity '{self.verified_email}' created
successfully.")
        except ClientError as e:
            # If the email identity already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
                print(f"Email identity '{self.verified_email}' already exists.")
            else:
                raise e

        try:
            template_content = {
                "Subject": "Weekly Coupons Newsletter",
                "Html": load_file_content("coupon-newsletter.html"),
                "Text": load_file_content("coupon-newsletter.txt"),
            }
            self.ses_client.create_email_template(
                TemplateName=TEMPLATE_NAME, TemplateContent=template_content
            )
            print(f"Email template '{TEMPLATE_NAME}' created successfully.")
        except ClientError as e:
            # If the template already exists, skip and proceed
            if e.response["Error"]["Code"] == "AlreadyExistsException":
```

```
                print(f"Email template '{TEMPLATE_NAME}' already exists.")
            else:
                raise e

        try:

self.ses_client.delete_contact_list(ContactListName=CONTACT_LIST_NAME)
            print(f"Contact list '{CONTACT_LIST_NAME}' deleted successfully.")
        except ClientError as e:
            # If the contact list doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Contact list '{CONTACT_LIST_NAME}' does not exist.")
            else:
                print(e)

            try:

self.ses_client.delete_email_identity(EmailIdentity=self.verified_email)
                print(f"Email identity '{self.verified_email}' deleted
successfully.")
            except ClientError as e:
                # If the email identity doesn't exist, skip and proceed
                if e.response["Error"]["Code"] == "NotFoundException":
                    print(f"Email identity '{self.verified_email}' does not
exist.")
                else:
                    print(e)

        try:
            self.ses_client.delete_email_template(TemplateName=TEMPLATE_NAME)
            print(f"Email template '{TEMPLATE_NAME}' deleted successfully.")
        except ClientError as e:
            # If the email template doesn't exist, skip and proceed
            if e.response["Error"]["Code"] == "NotFoundException":
                print(f"Email template '{TEMPLATE_NAME}' does not exist.")
            else:
                print(e)
```

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

- CreateContactList
- CreateEmailIdentity
- CreateEmailTemplate
- DeleteContactList
- DeleteEmailIdentity
- DeleteEmailTemplate
- ListContacts
- SendEmail.simple
- SendEmail.template

Rust

SDK for Rust

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

```
        match self
            .client
            .create_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list created
successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateContactListError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Contact list already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating contact list: {}", e)),
            },
```

```
        }

            match self
                .client
                .create_contact()
                .contact_list_name(CONTACT_LIST_NAME)
                .email_address(email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Contact created for {}", email)?,
                Err(e) => match e.into_service_error() {
                    CreateContactError::AlreadyExistsException(_) => writeln!(
                        self.stdout,
                        "Contact already exists for {}, skipping creation.",
                        email
                    )?,
                    e => return Err(anyhow!("Error creating contact for {}: {}",
 email, e)),
                },
            }

        let contacts: Vec<Contact> = match self
            .client
            .list_contacts()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(list_contacts_output) => {
                list_contacts_output.contacts.unwrap().into_iter().collect()
            }
            Err(e) => {
                return Err(anyhow!(
                    "Error retrieving contact list {}: {}",
                    CONTACT_LIST_NAME,
                    e
                ))
            }
        };

            let coupons = std::fs::read_to_string("../resources/newsletter/
sample_coupons.json")
                .unwrap_or_else(|_| r#"{"coupons":[]}"#.to_string());
```

```
            let email_content = EmailContent::builder()
                .template(
                    Template::builder()
                        .template_name(TEMPLATE_NAME)
                        .template_data(coupons)
                        .build(),
                )
                .build();

            match self
                .client
                .send_email()
                .from_email_address(self.verified_email.clone())

.destination(Destination::builder().to_addresses(email.clone()).build())
                .content(email_content)
                .list_management_options(
                    ListManagementOptions::builder()
                        .contact_list_name(CONTACT_LIST_NAME)
                        .build()?,
                )
                .send()
                .await
            {
                Ok(output) => {
                    if let Some(message_id) = output.message_id {
                        writeln!(
                            self.stdout,
                            "Newsletter sent to {} with message ID {}",
                            email, message_id
                        )?;
                    } else {
                        writeln!(self.stdout, "Newsletter sent to {}", email)?;
                    }
                }
                Err(e) => return Err(anyhow!("Error sending newsletter to {}:
{}", email, e)),
            }

       match self
            .client
            .create_email_identity()
            .email_identity(self.verified_email.clone())
            .send()
```

```
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email identity created
 successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailIdentityError::AlreadyExistsException(_) => {
                    writeln!(
                        self.stdout,
                        "Email identity already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email identity: {}", e)),
            },
        }

        let template_html =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.html")
                .unwrap_or_else(|_| "Missing coupon-
newsletter.html".to_string());
        let template_text =
            std::fs::read_to_string("../resources/newsletter/coupon-
newsletter.txt")
                .unwrap_or_else(|_| "Missing coupon-newsletter.txt".to_string());

        // Create the email template
        let template_content = EmailTemplateContent::builder()
            .subject("Weekly Coupons Newsletter")
            .html(template_html)
            .text(template_text)
            .build();

        match self
            .client
            .create_email_template()
            .template_name(TEMPLATE_NAME)
            .template_content(template_content)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template created
 successfully.")?,
            Err(e) => match e.into_service_error() {
                CreateEmailTemplateError::AlreadyExistsException(_) => {
```

```
                    writeln!(
                        self.stdout,
                        "Email template already exists, skipping creation."
                    )?;
                }
                e => return Err(anyhow!("Error creating email template: {}", e)),
            },
        }

        match self
            .client
            .delete_contact_list()
            .contact_list_name(CONTACT_LIST_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Contact list deleted
successfully.")?,
            Err(e) => return Err(anyhow!("Error deleting contact list: {e}")),
        }

            match self
                .client
                .delete_email_identity()
                .email_identity(self.verified_email.clone())
                .send()
                .await
            {
                Ok(_) => writeln!(self.stdout, "Email identity deleted
successfully.")?,
                Err(e) => {
                    return Err(anyhow!("Error deleting email identity: {}", e));
                }
            }

        match self
            .client
            .delete_email_template()
            .template_name(TEMPLATE_NAME)
            .send()
            .await
        {
            Ok(_) => writeln!(self.stdout, "Email template deleted
successfully.")?,
```

```
            Err(e) => {
                return Err(anyhow!("Error deleting email template: {e}"));
            }
        }
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の以下のトピックを参照してください。
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいSES での Amazon の使用 AWS SDK。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon Simple Email Serviceのセキュリティ

AWS ではクラウドセキュリティが最優先事項です。セキュリティを最も重視する組織の要件を満たすために構築された AWS のデータセンターとネットワークアーキテクチャは、お客様に大きく貢献します。

セキュリティは、AWS とお客様とが共有する責務です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ - AWS は、AWS Cloud で AWS のサービスを実行するインフラストラクチャを保護する責任を負います。また、AWSは、使用するサービスを安全に提供します。AWS コンプライアンスプログラムの一環として、サードパーティー監査者が定期的にセキュリティの有効性をテストおよび検証します。Amazon Simple Email Serviceに適用するコンプライアンスプログラムについては、の「コンプライアンスプログラムによる対象範囲の AWS サービス」を参照してください。
- クラウド内のセキュリティ – お客様の責任範囲は、ご使用の AWS のサービスに応じて異なります。また、お客様は、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、Amazon Simple Email Service使用時における責任共有モデルの適用法を理解するのに役立ちます。ここでは、セキュリティやコンプライアンスに関する目標を達成できるようにAmazon Simple Email Serviceを設定する方法について説明します。また、Amazon Simple Email Serviceのリソースのモニタリングとセキュリティ確保に役立つ、他の AWS のサービスを使用する方法も知ることができます。

> **Note**
>
> メール、スパム、マルウェアの配布などの AWS リソースの不正使用をレポートする場合、このデベロッパーガイドのページのリンクは使用しないでください。このフォームは、Trust & Safety ではなく AWS ドキュメンテーションチームが受信するためです。代わりに、AWS リソースの不正使用の報告方法資源ページでは、手順に従って、AWS Trust & Safety チームに連絡して、あらゆる種類の Amazon AWS不正使用を報告します。

内容

- Amazon での Identity and Access Management SES
- Amazon SES でのログ記録とモニタリング
- Amazon Simple Email Service のコンプライアンス検証
- Amazon Simple Email Serviceの耐障害性
- Amazon Simple Email Service での基盤セキュリティ
- Amazon SES を用いた VPC エンドポイントの設定

Amazon Simple Email Serviceにおけるデータ保護

AWS責任共有モデルは、Amazon Simple Email Serviceのデータ保護に適用されます。このモデルで説明されているように、AWS は、AWS クラウド のすべてを実行するグローバルインフラストラクチャを保護する責任があります。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。

データを保護するため、AWS アカウント 認証情報を保護し、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セットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要な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。
- AWS CloudTrail で API とユーザーアクティビティロギングをセットアップします。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「AWS CloudTrail ユーザーガイド」の「Working with CloudTrail trails」を参照してください。
- AWS のサービス 内のすべてのデフォルトセキュリティ管理に加え、AWS 暗号化ソリューションを使用します。
- 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 を使用して AWS にアクセスする際に FIPS 140-3 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) 140-3」を参照してください。

お客様の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、AWS CLI、または AWS SDK を使用して Amazon Simple Email Service または他の AWS のサービス を使用する場合も同様です。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。

内容

Amazon SES の保管中のデータ暗号化

Amazon SES は、保管中のすべてのデータをデフォルトで暗号化します。デフォルトの暗号化は、データの保護に伴う運用のオーバーヘッドと複雑な作業の軽減につながります。暗号化を使用すると、厳格な暗号化コンプライアンスと規制要件を満たす Mail Manager アーカイブを作成することもできます。

SES は、次の暗号化オプションを提供しています。

- AWS 所有キー – SES はデフォルトでこれらを使用します。AWS が所有するキーを表示、管理、使用したり、その使用を監査したりすることはできません。ただし、データを暗号化するキーを保護するために何らかの措置を講じたり、プログラムを変更したりする必要はありません。詳細については、AWS Key Management Service デベロッパーガイドの「AWS 所有キー」を参照してください。
- カスタマーマネージドキー – SES は、お客様が作成、所有、管理するカスタマーマネージド対称キーの使用をサポートしています。お客様が暗号化を完全に管理するため、以下のとおりのタスクを実行できます。
  - キーポリシーの策定と維持
  - IAM ポリシーとグラントの策定と維持
  - キーポリシーの有効化と無効化
  - キー暗号化マテリアルのローテーション
  - タグの追加
  - キーエイリアスの作成

- キー削除のスケジュール設定

独自のキーを使用するには、SES リソースを作成する際にカスタマーマネージドキーを選択します。

詳細については、「AWS Key Management Service デベロッパーガイド」の「カスタマーマネージドキー」を参照してください。

> **Note**
>
> SES は、AWS 所有キーを使用して、保管中の暗号化を無料で自動的に有効にします。ただし、カスタマーマネージドキーの使用には AWS KMS 料金が適用されます。料金の詳細については、「AWS Key Management Service 料金」を参照してください。

カスタマーマネージドキーを作成する

対称カスタマーマネージドキーを作成するには、AWS Management Console または AWS KMS API を使用します。

対称カスタマーマネージドキーを作成するには

「AWS Key Management Service デベロッパーガイド」の「対称暗号化 KMS キーの作成」の手順に従ってください。

> **Note**
>
> アーカイブするには、キーが以下の要件を満たしている必要があります。
>
> - KMS キーは対称である必要があります。
> - キーマテリアルのオリジンは `AWS_KMS` である必要があります。
> - キーの使用状況は、`ENCRYPT_DECRYPT` である必要があります。

キーポリシー

キーポリシーは、カスタマーマネージドキーへのアクセスを制御します。すべてのカスタマーマネージドキーには、キーポリシーが 1 つだけ必要です。このポリシーには、そのキーを使用できるユー

ザーとその使用方法を決定するステートメントが含まれています。カスタマーマネージドキーを作成する際に、キーポリシーを指定することができます。詳細については、 AWS Key Management Service デベロッパーガイド の「カスタマーマネージドキーへのアクセスの管理」を参照してください。

カスタマーマネージドキーを Mail Manager アーカイブで使用するには、キーポリシーで以下の API オペレーションを許可する必要があります。

- kms:DescribeKey – SES がキーを検証できるように、カスタマーマネージドキーの詳細を提供します。
- kms:GenerateDataKey – SES が保管中のデータを暗号化するためのデータキーを生成できるようにします。
- kms:Decrypt – SES が保存されたデータを API クライアントに返す前に復号化できるようにします。

一般的なキーポリシーの例は、以下のとおりです。

```
{
            "Sid": "Allow SES to encrypt/decrypt",
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
```

詳細については、「AWS Key Management Service デベロッパーガイド」の「ポリシーでのアクセス許可の指定」を参照してください。

トラブルシューティングの詳細については、「AWS Key Management Service デベロッパーガイド」の「キー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。

Mail Manager アーカイブ用のカスタマーマネージドキーの指定

AWS 所有キーを使用する代わりに、カスタマーマネージドキーを指定できます。アーカイブを作成する際に、KMS キー ARN を入力してデータキーを指定できます。この ARN は、Mail Manager アーカイブがアーカイブ内のすべての顧客データの暗号化に使用するものです。

- KMS key ARN – AWS KMS カスタマーマネージドキーのキー識別子です。キー ID、キー ARN、エイリアス名、またはエイリアス ARN を入力します。

Amazon SES 暗号化コンテキスト

暗号化コンテキストは、データに関する追加のコンテキスト情報が含まれたキーバリューペアのオプションのセットです。

AWS KMS は、暗号化コンテキストを追加の認証済みデータとして使用して、認証済み暗号化をサポートします。データの暗号化リクエストに暗号化コンテキストを組み込むと、AWS KMS は暗号化コンテキストを暗号化後のデータにバインドします。データを復号化するには、そのリクエストに (暗号化時と) 同じ暗号化コンテキストを含めます。

> **Note**
>
> Amazon SES は、アーカイブ作成の暗号化コンテキストをサポートしていません。代わりに、IAM または KMS ポリシーを使用します。ポリシーの例については、このセクションの後半の「アーカイブ作成ポリシー」を参照してください。

Amazon SES 暗号化コンテキスト

SES は、すべての AWS KMS 暗号化オペレーションで同じ暗号化コンテキストを使用します。キーは `aws:ses:arn` で、値はリソースの Amazon リソースネーム (ARN) です。

Example

```
"encryptionContext": {
    "aws:ses:arn": "arn:aws:ses:us-west-2:111122223333:ExampleResourceName/
ExampleResourceID"
}
```

モニタリングに暗号化コンテキストを使用する

対称カスタマーマネージドキーを使用して SES リソースを暗号化する場合、監査レコードとログで暗号化コンテキストを使用し、カスタマーマネージドキーが使用されている方法を特定することもできます。暗号化コンテキストは、AWS CloudTrail または Amazon CloudWatch Logs によって生成されたログにも表示されます。

暗号化コンテキストを使用してカスタマーマネージドキーへのアクセスを制御する

conditions が対称カスタマーマネージドキーへのアクセスを制御するための条件として、キーポリシーと IAM ポリシー内の暗号化コンテキストを使用することもできます。付与する際に、暗号化コンテキストの制約を使用することもできます。

SES は、許可で暗号化コンテキスト制約を使用して、アカウントまたはリージョン内のカスタマーマネージドキーへのアクセスを制御します。権限の制約では、権限によって許可されるオペレーションで指定された暗号化コンテキストを使用する必要があります。

Example

次に、特定の暗号化コンテキストのカスタマーマネージドキーへのアクセスを付与するキーポリシーステートメントの例を示します。このポリシーステートメントの条件では、権限に暗号化コンテキストを指定する暗号化コンテキスト制約が必要です。

```
{
    "Sid": "Enable DescribeKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
     },
     "Action": "kms:DescribeKey",
     "Resource": "*"
},
{
     "Sid": "Enable CreateGrant",
     "Effect": "Allow",
     "Principal": {
         "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
     },
     "Action": "kms:CreateGrant",
     "Resource": "*",
     "Condition": {
         "StringEquals": {
             "kms:EncryptionContext:aws:ses:arn": "arn:aws:ses:us-
west-2:111122223333:ExampleResourceName/ExampleResourceID"
          }
```

```
    }
}
```

アーカイブ作成ポリシー

次のポリシー例は、アーカイブ作成を有効にする方法を説明しています。このポリシーはすべてのアセットで機能します。

IAM ポリシー

```
{
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "ses:CreateArchive",
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "ses.us-east-1.amazonaws.com",
                    "kms:CallerAccount": "012345678910"
                }
            }
        }
```

AWS KMS ポリシー

```
{
            "Sid": "Allow SES to encrypt/decrypt",
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": [
```

```
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
```

Amazon SES の暗号キーのモニタリング

Amazon SES リソースで AWS KMS カスタマーマネージドキーを使用する場合、SES が AWS KMS に送信するリクエストの追跡に、AWS CloudTrail または Amazon CloudWatch Logs を使用できます。

次の例は、カスタマーマネージドキーが暗号化したデータにアクセスするために SES が呼び出す KMS オペレーションをモニタリングするための `GenerateDataKey`、`Decrypt`、`DescribeKey` の AWS CloudTrail イベントです。

GenerateDataKey

リソースの AWS KMS カスタマーマネージドキーを有効にすると、SES は一意のテーブルキーを作成します。リソースの AWS KMS カスタマーマネージドキーを指定する AWS KMS に `GenerateDataKey` リクエストを送信します。

Mail Manager アーカイブリソースの AWS KMS カスタマーマネージドキーを有効にすると、保管中のアーカイブデータを暗号化する際に `GenerateDataKey` が使用されます。

以下のイベント例では `GenerateDataKey` オペレーションを記録しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ses.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "encryptionContext": {
```

```
            "aws:ses:arn": "arn:aws:ses:us-west-2:111122223333:ExampleResourceName/
ExampleResourceID"
        },
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "57f5dbee-16da-413e-979f-2c4c6663475e"
}
```

Decrypt

ユーザーが暗号化されたリソースにアクセスすると、SES は Decrypt オペレーションを呼び出し、保存されている暗号化されたデータキーを使用して暗号化済みのデータにアクセスします。

以下のイベント例では Decrypt オペレーションを記録しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ses.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:10:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
```

```
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "encryptionContext": {
            "aws:ses:arn": "arn:aws:ses:us-west-2:111122223333:ExampleResourceName/
ExampleResourceID"
        },
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "dc129381-1d94-49bd-b522-f56a3482d088"
}
```

DescribeKey

SES は DescribeKey オペレーションを使用して、リソースに関連付けられている AWS KMS カスタマーマネージドキーがアカウントとリージョンに存在するかを確認します。

以下のイベント例では、DescribeKey オペレーションを記録しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
```

```
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-22T17:02:00Z"
            }
        },
        "invokedBy": "ses.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "keyId": "00dd0db0-0000-0000-ac00-b0c000SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

詳細はこちら

次のリソースは、保管時のデータ暗号化についての詳細を説明しています。

- AWS Key Management Service 基本概念の詳細については、「AWS Key Management Service デベロッパーガイド」を参照してください。
- 詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS Key Management Service 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。

転送中の暗号化

デフォルトで、Amazon SES は便宜的 TLSを使用します。つまり、Amazon SES は常に受信メールサーバーへの安全な接続を確立しようとします。安全な接続を確立できない場合、平文メッセージを送信します。Amazon SES は、安全な接続を確立できる受信用Eメールサーバーに対してのみメッセージを送信するように、この動作を変更できます。詳細については、「Amazon SES およびセキュリティプロトコル」を参照してください。

Amazon SES から個人データを削除する

使用方法によっては、個人用と見なされる可能性のあるデータが Amazon SES に保存される場合があります。例えば、Amazon SES を使用して E メールを送信するには、少なくとも 1 つの検証済み ID を指定する必要があります (メールアドレスまたはドメイン)。Amazon SES コンソールまたは Amazon SES API を使用して、この個人データを完全に削除できます。

この章では、個人用と見なされる可能性があるさまざまなタイプのデータを削除する手順について説明します。

内容

- カスタム検証 E メールテンプレートのデータを削除する
- AWS アカウントを解約することですべての個人データを削除する

アカウントレベルのサプレッションリストからメールアドレスを削除する

Amazon SES には、オプションのアカウントレベルのサプレッションリストが含まれています。この機能を有効にすると、バウンスや苦情が発生したときに、メールアドレスがサプレッションリストに自動的に追加されます。メールアドレスは、削除するまでこのリストに残ります。アカウントレベルのサプレッションリストの詳細については、「Amazon SESアカウントレベルのサプレッションリストの使用」を参照してください。

アカウントレベルのサプレッションリストからメールアドレスを削除するにはAmazon SES API v2で `DeleteSuppressedDestination` オペレーションを実行します。ここでは、AWS CLI を使用してメールアドレスを削除する手順について説明します。AWS CLI のインストールおよび設定の詳細については、「AWS Command Line Interface ユーザーガイド」を参照してください。

AWS CLI を使用してアカウントレベルのサプレッションリストからアドレスを削除するには

- コマンドラインで以下のコマンドを入力します。

  ```
  aws sesv2 delete-suppressed-destination --email-address recipient@example.com
  ```

  前のコマンドで、*recipient@example.com* を、アカウントレベルのサプレッションリストから削除するメールアドレスに置き換えます。

Amazon SES を使用して送信された E メールに関するデータを削除する

Amazon SESを使用してEメールを送信するとき、Eメールに関する情報を他のAWSサービスに送信することができます。例えば、E メールイベントについての情報 (配信、開封、クリックなど) を Firehose に送信できます。このイベントデータには通常、メールアドレスと、E メールの送信元の IP アドレスが含まれます。さらに、E メールが送信されたすべての受信者のメールアドレスも含まれます。

Firehose を使用すると、E メールのイベントデータを Amazon Simple Storage Service、Amazon OpenSearch Service、Amazon Redshift など、複数の送信先にストリーミングできます。このデータを削除するには、まず Firehose へのデータのストリーミングを停止した後に、既にストリーミングされたデータを削除する必要があります。Firehose への Amazon SES イベントデータのストリーミングを停止するには、Firehose イベントの送信先を削除する必要があります。

Amazon SES コンソールを使用して Firehose イベントの送信先を削除するには

1. Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. [Email Sending] で、[Configuration Sets] を選択します。
3. 設定セットのリストで、Firehose イベント送信先が含まれている設定セットを選択します。
4. 削除する Firehose イベント送信先のとなりにある [削除]
   ( )
   ボタンをクリックします。
5. 必要に応じて、Firehose が他のサービスに書き込んだデータを削除します。詳細については、「the section called “保存されたイベントデータを削除する”」を参照してください。

Amazon SES API を使用してイベントの送信先を削除することもできます。次の手順では、AWS Command Line Interface (AWS CLI) を使用して Amazon SES API を操作します。AWS SDK を使用したり、HTTP リクエストを直接行ったりすることで API を操作することもできます。

AWS CLI を使用して Firehose イベントの送信先を削除するには

1. コマンドラインから、以下のコマンドを入力します。

   ```
   aws sesv2 delete-configuration-set-event-destination --configuration-set-
   name configSet \
   --event-destination-name eventDestination
   ```

   このコマンドの *configSet* は、Firehose イベント送信先を含む設定セット名に置き換えます。*eventDestination* を Firehose イベント送信先名に置き換えます。
2. 必要に応じて、Firehose が他のサービスに書き込んだデータを削除します。詳細については、「the section called “保存されたイベントデータを削除する”」を参照してください。

保存されたイベントデータを削除する

他の AWS サービスから情報を削除する方法については、次のドキュメントを参照してください。

- Amazon Simple Storage Serviceユーザーガイドの「オブジェクトとバケットの削除」
- Amazon OpenSearch Service Developer Guide の「Delete an OpenSearch Service Domain」
- 『Amazon Redshift クラスター管理ガイド』の「クラスターの削除」

Firehose を使用すると、AWS でサポートされていない、または AWS Management Console で管理されないサードパーティーのサービスである Splunk に E メールデータをストリーミングすることもできます。Splunk からデータを削除する方法の詳細については、システム管理者に問い合わせるか、Splunk ウェブサイトでドキュメントを参照してください。

ID に関するデータを削除する

ID には、Amazon SES を使用して E メールを送信するために使用するメールアドレスとドメインが含まれています。管轄区域によっては、メールアドレスやドメインは個人を特定できるデータと見なされることがあります。

Amazon SES コンソールを使用して ID を削除するには

1. Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. [Identity Management] で、以下のいずれかを行います。
   - ドメインを削除する場合は、[Domains] を選択します。
   - E メールアドレスを削除する場合は、[Email Addresses] を選択します。
3. 削除する ID を選択し、[Remove] を選択します。
4. 確認ダイアログボックスで、[Yes, Delete Identity] を選択します。

Amazon SES API を使用して ID を削除することもできます。次の手順では、AWS Command Line Interface (AWS CLI) を使用して Amazon SES API を操作します。AWS SDK を使用したり、HTTP リクエストを直接行ったりすることで API を操作することもできます。

AWS CLI を使用して ID を削除するには

- コマンドラインから、以下のコマンドを入力します。

```
aws ses delete-identity --identity sender@example.com
```

このコマンドで、*sender@example.com* を、削除する ID に置き換えます。

送信者認証データを削除する

送信者認証とは、自分の代わりに別のユーザーが E メールを送信できるように Amazon SES を設定するプロセスを指します。送信者認可を有効にするには、「Amazon SES での送信承認の使用」で説明されているようにポリシーを作成する必要があります。これらのポリシーには、(自分の代わり

に E メールを送信するユーザーまたはグループに関連付けられている) AWS ID に加えて、(自分に属する) ID が含まれています。この個人データは、送信者認証ポリシーを変更または削除することで削除することができます。次の手順では、これらのポリシーを削除する方法を示します。

Amazon SES コンソールを使用して送信者認証ポリシーを削除するには

1. Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. [Identity Management] で、以下のいずれかを行います。
   - 削除する送信者認証ポリシーがドメインと関連付けられている場合は、[Domains] を選択します。
   - 削除する送信者認証ポリシーが E メールアドレスと関連付けられている場合は、[Email Addresses] を選択します。
3. [Identity Policies] で、削除するポリシーを選択し、[Remove Policy] を選択します。

Amazon SES API を使用して送信者認証ポリシーを削除することもできます。次の手順では、AWS Command Line Interface (AWS CLI) を使用して Amazon SES API を操作します。AWS SDK を使用したり、HTTP リクエストを直接行ったりすることで API を操作することもできます。

AWS CLI を使用して送信者認証ポリシーを削除するには

- コマンドラインから、以下のコマンドを入力します。

  ```
  aws ses delete-identity-policy --identity example.com --policy-name samplePolicy
  ```

  このコマンドで、*example.com* を、送信者認証ポリシーが含まれている ID に置き換えます。*samplePolicy* を送信者認証ポリシーの名前に置き換えます。

受信ルールに関連するデータを削除する

Amazon SES を使用して受信 E メールを受信する場合、1 つ以上の ID (E メールアドレスやドメイン) に適用される受信ルールを作成できます。これらのルールにより、指定された ID に送信された受信メールに対して Amazon SES が行う処理が決まります。

Amazon SES コンソールを使用して受信ルールを削除するには

1. Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. [Email Receiving] で [Rule Sets] を選択します。

3. 受信ルールがアクティブなルールセットの一部である場合、[View Active Rule Set] を選択します。それ以外の場合は、削除する受信ルールセットが含まれるルールセットを選択します。
4. 受信ルールのリストで、削除するルールを選択します。
5. [Actions] メニューで、[Delete] を選択します。
6. 確認ダイアログボックスで、[Delete] を選択します。

Amazon SES API を使用して受信ルールを削除することもできます。次の手順では、AWS Command Line Interface (AWS CLI) を使用して Amazon SES API を操作します。AWS SDK を使用したり、HTTP リクエストを直接行ったりすることで API を操作することもできます。

AWS CLI を使用して受信ルールを削除するには

- コマンドラインから、以下のコマンドを入力します。

```
aws ses delete-receipt-rule --rule-set myRuleSet --rule-name myReceiptRule
```

このコマンドで、*myRuleSet* を、受信ルールが含まれる受信ルールセットの名前に置き換えます。*myReceiptRule* を、削除する受信ルールの名前に置き換えます。

IP アドレスフィルターに関連するデータを削除する

Amazon SES を使用して受信メールを受信する場合、特定の IP アドレスから送信されるメッセージを明示的に許可またはブロックするフィルターを作成できます。

Amazon SES コンソールを使用して IP アドレスフィルターを削除するには

1. Amazon SES コンソール (https://console.aws.amazon.com/ses/) を開きます。
2. [Email Receiving] で [IP Address Filters] を選択します。
3. IP アドレスフィルターのリストで、削除するフィルターを選択し、[Delete] を選択します。

Amazon SES API を使用して IP アドレスフィルターを削除することもできます。次の手順では、AWS Command Line Interface (AWS CLI) を使用して Amazon SES API を操作します。AWS SDK を使用したり、HTTP リクエストを直接行ったりすることで API を操作することもできます。

AWS CLI を使用して IP アドレスフィルターを削除するには

- コマンドラインから、以下のコマンドを入力します。

```
aws ses delete-receipt-filter --filter-name IPfilter
```

このコマンドで、*IPfilter* を、削除する IP アドレスフィルターの名前に置き換えます。

E メールテンプレートのデータを削除する

E メールを送信するために E メールテンプレートを使用する場合、テンプレートを設定した方法によっては、テンプレートに個人データが含まれている可能性があります。たとえば、受信者が詳細を問い合わせることができるように E メールアドレスをテンプレートに追加した可能性があります。

E メールテンプレートを削除するには、Amazon SES API を使用する必要があります。

AWS CLI を使用して E メールテンプレートを削除するには

- コマンドラインから、以下のコマンドを入力します。

```
aws ses delete-template --template-name sampleTemplate
```

このコマンドで、*sampleTemplate* を、削除する E メールテンプレートの名前に置き換えます。

カスタム検証 E メールテンプレートのデータを削除する

新しい E メール送信アドレスを検証するためにカスタムテンプレートを使用する場合、テンプレートを設定した方法によっては、テンプレートに個人データが含まれている可能性があります。たとえば、受信者が詳細を問い合わせることができるように E メールアドレスを検証 E メールテンプレートに追加した可能性があります。

カスタム検証 E メールテンプレートを削除するには、Amazon SES API を使用する必要があります。

AWS CLI を使用してカスタム検証 E メールテンプレートを削除するには

- コマンドラインから、以下のコマンドを入力します。

```
aws ses delete-custom-verification-email-template --template-
name verificationEmailTemplate
```

このコマンドで、*verificationEmailTemplate* を、削除するカスタム検証 E メールテンプレートの名前に置き換えます。

AWS アカウントを解約することですべての個人データを削除する

AWSアカウントを解約することにより、Amazon SES に保存されたすべての個人データを削除することもできます。ただし、このアクションでは、他のすべての AWS サービスに保存した他のすべてのデータ (個人データと非個人データ) も削除されます。

AWS アカウントを解約すると、AWS アカウントのデータは 90 日間保持されます。その保持期間の後、完全に削除されて元に戻すことはできません。

AWS アカウントを解約するには

AWS アカウントを閉鎖する詳細な手順については、「AWS アカウントを閉鎖する」を参照してください。

Amazon での Identity and Access Management SES

Amazon Simple Email Service (Amazon IAM) で AWS Identity and Access Management （ SES) を使用して、ユーザー、グループ、またはロールが実行できるSESAPIアクションを指定できます。(このトピックでは、これらのエンティティをまとめて「ユーザー」と呼びます)。ユーザーが「From」、受取人、「Return-Path」の E メールアドレスに使用できる E メールアドレスをコントロールすることもできます。

たとえば、IAM ポリシーを作成して、組織内のユーザーはメールを送信できるが送信統計の確認などの管理作業を実行できないようにすることができます。別の例として、ユーザーが特定の「From」アドレスを使用している場合のみ、アカウントから SES を介して E メールを送信することを許可するポリシーを作成できます。

を使用するにはIAM、アクセス許可を明示的に定義するドキュメントである IAMポリシーを定義し、そのポリシーをユーザーにアタッチします。IAM ポリシーの作成方法については、「 IAMユーザーガイド」を参照してください。ポリシーで設定した制限が適用される以外に、ユーザーが SES を操作する方法や SES がリクエストを実行する方法に変更はありません。

> **Note**
>
> - アカウントがSESサンドボックスにある場合、その制限によりこれらのポリシーの一部が実装されなくなります。「」を参照してください本番稼働用アクセスのリクエスト。
> - 送信承認ポリシーを使用することで、SES へのアクセスを制御することもできます。IAM ポリシーは個々のユーザーが実行できる操作を制限しますが、送信承認ポリシーは個々の検証済み ID の使用方法を制限します。さらに、クロスアカウントアクセスを許可できるのは送信承認ポリシーのみです。送信承認の詳細については、「Amazon SES での送信承認の使用」を参照してください。

既存のユーザーのSESSMTP認証情報を生成する方法については、「」を参照してくださいAmazon SES SMTP 認証情報を取得。

SES にアクセスするための IAM ポリシーの作成

このセクションでは、特に SES で IAM ポリシーを使用する方法を説明します。IAM ポリシーを作成する一般的な方法については、「 IAMユーザーガイド」を参照してください。

SES で IAM を使用する理由は次の 3 つです。

- E メール送信アクションを制限するため。
- ユーザーが送信する E メールの「From」、受取人、「Return-Path」のアドレスを制限するため。
- API ユーザーが使用を許可された を呼び出すことができる期間などAPIs、使用の一般的な側面を制御する。

アクションの制限

ユーザーが実行できる SES アクションをコントロールするには、IAM ポリシーの `Action` 要素を使用します。API 名前の前に小文字の文字列 を付けることで、 `Action`要素を任意のSESAPIアクションに設定できます`ses:`。たとえば、`Action` を `ses:SendEmail`、`ses:GetSendStatistics`、または `ses:*` (すべてのアクションの場合) に設定できます。

次に、`Action` に応じて、次のように `Resource` 要素を指定します。

`Action`要素が E メール送信 APIs (つまり、 **`ses:SendEmail`**および/または **`ses:SendRawEmail`**) へのアクセスのみを許可する場合 :

- ユーザーが 内の任意の ID から送信できるようにするには AWS アカウント、 `Resource` を * に設定します。
- ユーザーが送信できる ID を制限するには、 `Resource`を、ユーザーに使用を許可する ID ARNsのに設定します。

`Action`要素がすべての へのアクセスを許可する場合APIs :

- ユーザーの送信元のアイデンティティを制限しない場合、`Resource` を * に設定します。
- ユーザーの送信元のアイデンティティを制限する場合、2 つのポリシー (または、1 つのポリシー内に 2 つのステートメント) を作成する必要があります。
  - を許可された non-email-sending の明示的なリスト`Action`に設定APIsし、 を * `Resource`に設定したもの
  - E メール送信 APIs (`ses:SendEmail` および/または `ses:SendRawEmail`) の 1 つに `Action` を設定し、ユーザーに使用を許可する ID の ARN(s) に `Resource` を設定します。

使用可能なSESアクションのリストについては、Amazon Simple Email Service APIリファレンスを参照してください。ユーザーが SMTPインターフェイスを使用する場合は、`ses:SendRawEmail`少なくとも へのアクセスを許可する必要があります。

E メールアドレスの制限

ユーザーを特定の E メールアドレスに制限する場合、`Condition` ブロックを使用できます。`Condition` ブロックでは、「IAMユーザーガイド」で説明されているように、条件キーを使用して条件を指定します。条件キーを使用して、次の E メールアドレスをコントロールできます。

> **Note**
>
> これらの E メールアドレス条件キーは、次の表APIsに記載されている にのみ適用されます。

条件キー	説明	API
`ses:Recipients`	To:、「CC」、「」アドレスを含む受信者BCCアドレスを制限します。	`SendEmail` , `SendRawEmail`

条件キー	説明	API
ses:FromAddress	「From」アドレスを制限します。	SendEmail , SendRawEmail , SendBounce
ses:FromDisplayName	表示名として使用される「From」アドレスを制限します。	SendEmail , SendRawEmail
ses:FeedbackAddress	"Return-Path" アドレス (E メールのフィードバック転送によりバウンスや苦情を送信できるアドレス) を制限します。E メールのフィードバック転送の詳細については、「E メールを介したAmazon SES に関する通知の受信」を参照してください。	SendEmail , SendRawEmail
ses:MultiRegionEndpointId	E メールを送信するときに使用するエンドポイント ID を制御できます。	SendEmail , SendBulkEmail

SES API バージョンによる制限

条件で ses:ApiVersionキーを使用すると、 のバージョンSESに基づいて SES へのアクセスを制限できますAPI。

> **Note**
>
> SES SMTP インターフェイスは SESAPIのバージョン 2 を使用しますses:SendRawEmail。

一般的なAPI使用の制限

条件に AWS全体のキーを使用することで、ユーザーが へのアクセスを許可されている日時などの側面SESに基づいて、 へのアクセスを制限できますAPIs。 は、次の AWS全体のポリシーキーのみSESを実装します。

- `aws:CurrentTime`
- `aws:EpochTime`
- `aws:SecureTransport`
- `aws:SourceIp`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:UserAgent`
- `aws:VpcSourceIp`

これらのキーの詳細については、「IAMユーザーガイド」を参照してください。

SES の IAM ポリシー例

このトピックでは、特定の条件下でのみ SES へのユーザーアクセスを許可するポリシーの例を示します。

このセクションのポリシーの例：

- すべての SES アクションへのフルアクセスを許可
- SES API バージョン 2 のみへのアクセスの許可
- E メール送信アクションへのアクセスのみを許可
- 送信期間の制限
- 受取人アドレスの制限
- "From" アドレスの制限
- E メール送信者の表示名の制限
- バウンスや苦情のフィードバックの制限と送信先

すべての SES アクションへのフルアクセスを許可

次のポリシーでは、任意の SES アクションの呼び出しをユーザーに許可します。

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

SES API バージョン 2 のみへのアクセスの許可

次のポリシーでは、ユーザーはAPIバージョン 2 のSESアクションのみを呼び出すことができます。

```
{
                    "Version":"2012-10-17",
                    "Statement":[
                        {
                            "Effect":"Allow",
                            "Action":[
                            "ses:*"
                            ],
                            "Resource":"*",
                            "Condition": {
                                "StringEquals" : {
                                "ses:ApiVersion" : "2"
                                }
                            }
                        }
                    ]
                }
```

E メール送信アクションへのアクセスのみを許可

次のポリシーでは、SES を使用した E メールの送信をユーザーに許可しますが、SES 送信統計へのアクセスなどの管理作業を実行することは許可しません。

```
{
```

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

送信期間の制限

次のポリシーでは、2018 年 9 月中にAPIsのみ E SESメール送信を呼び出すことをユーザーに許可します。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Resource":"*",
      "Condition":{
        "DateGreaterThan":{
          "aws:CurrentTime":"2018-08-31T12:00Z"
        },
        "DateLessThan":{
          "aws:CurrentTime":"2018-10-01T12:00Z"
        }
      }
    }
  ]
}
```

受取人アドレスの制限

次のポリシーでは、ユーザーが E SESメール送信 を呼び出すことを許可しますがAPIs、ドメイン example.com の受信者アドレスに対してのみ許可します (StringLike では大文字と小文字が区別されます)。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Resource":"*",
      "Condition":{
        "ForAllValues:StringLike":{
          "ses:Recipients":[
            "*@example.com"
          ]
        }
      }
    }
  ]
}
```

"From" アドレスの制限

次のポリシーでは、「From」アドレスが marketing@example.com の場合にのみAPIs、E SESメール送信元の を呼び出すことをユーザーに許可します。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Resource":"*",
```

```
      "Condition":{
        "StringEquals":{
          "ses:FromAddress":"marketing@example.com"
        }
      }
    }
  ]
}
```

次のポリシーでは、「From」アドレスが SendBounce bounce@example.com の場合にのみAPI、ユーザーに の呼び出しを許可します。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "ses:SendBounce"
      ],
      "Resource":"*",
      "Condition":{
        "StringEquals":{
          "ses:FromAddress":"bounce@example.com"
        }
      }
    }
  ]
}
```

E メール送信者の表示名の制限

次のポリシーでは、「送信元」アドレスの表示名に Marketing が含まれている場合にのみ (StringLike では大文字と小文字が区別されます) APIs、ユーザーが SES E メール送信 を呼び出すことを許可します。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
```

```
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Resource":"*",
      "Condition":{
        "StringLike":{
          "ses:FromDisplayName":"Marketing"
        }
      }
    }
  ]
}
```

バウンスや苦情のフィードバックの制限と送信先

次のポリシーでは、E SESメールの送信元 を呼び出すことをユーザーに許可します。ただしAPIs、E メールの「Return-Path」が feedback@example.com に設定されている場合に限ります。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Resource":"*",
      "Condition":{
        "StringEquals":{
          "ses:FeedbackAddress":"feedback@example.com"
        }
      }
    }
  ]
}
```

Amazon Simple Email Service の AWS マネージドポリシー

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 マネージドポリシーの詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS のサービスは、AWS マネージドポリシーを維持および更新します。AWS マネージドポリシーの許可を変更することはできません。サービスでは、新しい機能を利用できるようにするために、AWS マネージドポリシーに権限が追加されることがあります。この種類の更新は、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、グループおよびロール) に影響を与えます。新しい機能が立ち上げられた場合や、新しいオペレーションが使用可能になった場合に、各サービスが 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 マネージドポリシーから権限を削除しないため、ポリシーの更新によって既存の権限が破棄されることはありません。

さらに、AWS は、複数のサービスにまたがるジョブ機能の特徴に対するマネージドポリシーもサポートしています。例えば、ReadOnlyAccess AWS マネージドポリシーでは、すべての AWS のサービスおよびリソースへの読み取り専用アクセスを許可します。サービスが新しい機能を起動する場合、AWS は、新たなオペレーションとリソース用に、読み取り専用の許可を追加します。ジョブ機能ポリシーのリストと説明については、IAM ユーザーガイドのジョブ機能の AWS 管理ポリシーを参照してください。

AWS マネージドポリシー: AmazonSESFullAccess

`AmazonSESFullAccess` ポリシーは IAM ID にアタッチできます。Amazon SES へのフルアクセスを提供します。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の「AmazonSESFullAccess」を参照してください。

AWS マネージドポリシー: AmazonSESReadOnlyAccess

`AmazonSESReadOnlyAccess` ポリシーは IAM ID にアタッチできます。Amazon SES への読み取り専用アクセスを提供します。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の「AmazonSESReadOnlyAccess」を参照してください。

AWS マネージドポリシー: AmazonSESServiceRolePolicy

IAM エンティティに `AmazonSESServiceRolePolicy` ポリシーをアタッチすることはできません。このポリシーは、Amazon SES がユーザーに代わってアクションを実行することを許可する

サービスにリンクされたロールにアタッチされます。詳細については、「Amazon SES 向けのサービスにリンクされたロール」を参照してください。

このポリシーのアクセス許可を確認するには、「AWS マネージドポリシーリファレンス」の「AmazonSESServiceRolePolicy」を参照してください。

Amazon Simple Email Service での AWS マネージドポリシーの更新

Amazon Simple Email Service が変更を追跡を開始して以来の AWS マネージドポリシーへの更新に関する詳細は、以下のとおりです。

変更	説明	日付
Amazon Simple Email Service に新しい管理ポリシーが追加されました	Amazon Simple Email Service で、サービスにリンクされたロール `AWSServiceRoleForAmazonSES` に `AmazonSESServiceRolePolicy` が追加されました。これにより、SES がユーザーに代わってアクションを実行できるようになりました。	2024 年 5 月 13 日
Amazon Simple Email Service がポリシーの定義を更新しました	Amazon Simple Email Service でこのテーブル (以下の行) の以前のエントリを明確化。Amazon Simple Email Service で、AmazonSESReadOnlyAccess マネージドポリシーに `ses:BatchGetMetricData` が追加されました。これにより、SES API BatchGetMetricData にアクセス権が付与されます。	2024 年 4 月 30 日

変更	説明	日付
Amazon Simple Email Service がポリシーの定義を更新しました	Amazon Simple Email Service で、AmazonSESReadOnlyAccess マネージドポリシーに `ses:BatchGet*` が追加されました。これにより、SES API BatchGetMetricData にアクセス権が付与されます。	2024 年 2 月 16 日
Amazon Simple Email Service で 2 つのポリシー定義を変更	Amazon Simple Email Service で、AmazonSESFullAccess 定義と AmazonSESReadOnlyAccess 定義の末尾から「AWS マネジメントコンソール経由」が削除されました	2023 年 5 月 3 日
Amazon Simple Email Service が変更の追跡を開始	Amazon Simple Email Service が AWS マネージドポリシーへの変更の追跡を開始しました	2023 年 4 月 5 日

Amazon SES 向けのサービスにリンクされたロールの使用

Amazon Simple Email Service (SES) は、AWS Identity and Access Management (IAM) の サービスにリンクされたロール を使用します。サービスにリンクされたロールは、Amazon SES に直接リンクされた独自のタイプの IAM ロールです。サービスにリンクされたロールは、SES が事前定義しており、このサービスがお客様の代わりにその他の AWS サービスを呼び出すのに必要なアクセス許可がすべて含まれています。

サービスにリンクされたロールを使用することで、必要なアクセス許可を手動で追加する必要がなくなるため、SES の設定が容易になります。SES は、サービスにリンクされたロールのアクセス許可を定義します。特に定義されている場合を除き、これらのロールを引き受けることができるのは、SES のみです。定義したアクセス許可には、信頼ポリシーと許可ポリシーが含まれます。この許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールは、まずその関連リソースを削除しなければ削除できません。これにより、リソースへのアクセス許可の不用意な削除が回避されるため、SES リソースは保護されます。

サービスリンクロールをサポートする他のサービスについては、「IAM と連動する AWS のサービス」を参照し、[Service-linked role (サービスリンクロール)] の列内で [Yes (はい)] と表記されたサービスを確認してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。

Amazon SES 向けのサービスにリンクされたロール

SES は、AWSServiceRoleForAmazonSES という名前のサービスにリンクされたロールを使用します。このロールを使用して、SES は SES リソースに代わって Amazon CloudWatch の基本モニタリングメトリクスを発行できます。

サービスにリンクされたロール AWSServiceRoleForAmazonSES は、以下のサービスを信頼して、このロールを引き受けます。

- `ses.amazonaws.com`

AmazonSESServiceRolePolicy という名前のロールアクセス許可ポリシーは、指定されたリソースに対して SES が以下のアクションを実行できるようにする AWS マネージドポリシーです。

- アクション: `AWS/SES` CloudWatch 名前空間における `cloudwatch:PutMetricData` このアクションは、SES にメトリクスデータを CloudWatch `AWS/SES` 名前空間に配置するアクセス許可を付与します。CloudWatch で利用できる SES メトリクスの詳細については、「Amazon SES でのログ記録とモニタリング」を参照してください。
- アクション: `AWS/SES/MailManager` CloudWatch 名前空間における `cloudwatch:PutMetricData` このアクションは、SES にメトリクスデータを CloudWatch `AWS/SES/MailManager` 名前空間に配置するアクセス許可を付与します。CloudWatch で利用できる SES メトリクスの詳細については、「Amazon SES でのログ記録とモニタリング」を参照してください。
- アクション: `AWS/SES/Addons` CloudWatch 名前空間における `cloudwatch:PutMetricData` このアクションは、SES にメトリクスデータを CloudWatch `AWS/SES/Addons` 名前空間に配置するアクセス許可を付与します。CloudWatch で利用できる SES メトリクスの詳細については、「Amazon SES でのログ記録とモニタリング」を参照してください。

ユーザー、グループ、ロールなどがサービスにリンクされたロールを作成、編集、削除できるようにするには、アクセス権限を設定する必要があります。詳細については、 IAM ユーザーガイド の「サービスリンクロールのアクセス許可」を参照してください。

Amazon SES 向けのサービスにリンクされたロールの作成

サービスリンクロールを手動で作成する必要はありません。AWS Management Console、AWS CLI、または AWS API で SES リソースを作成すると、SES が自動的にサービスにリンクされたロールを作成します。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は、同じ方法でアカウントにロールを再作成できます。SES リソースを作成する際も、SES が自動的にサービスにリンクされたロールを作成します。

Amazon SES 向けのサービスにリンクされたロールの編集

SES では、サービスにリンクされたロール AWSServiceRoleForAmazonSES の編集をユーザーに許可していません。サービスリンクロールを作成した後は、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロールの説明の編集はできます。

SES 向けのサービスにリンクされたロールの削除

サービスリンクロールが必要な機能またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、モニタリングや保守が積極的に行われていない未使用のエンティティを排除できます。ただし、手動で削除する前に、サービスリンクロールをクリーンアップする必要があります。

サービスにリンクされたロールのクリーンアップ

IAM を使用してサービスにリンクされたロールを削除する前に、まずすべての SES リソースを削除する必要があります。

> **Note**
>
> リソースを削除する際に、SES サービスで該当ロールが使用されている場合、削除が失敗する場合があります。失敗した場合は、数分待ってから操作を再試行してください。

サービスにリンクされたロールを手動で削除する

IAM コンソール、AWS CLI、または AWS API を使用して、サービスにリンクされたロール AWSServiceRoleForAmazonSES を削除します。詳細については、 IAM ユーザーガイド の「サービスにリンクされたロールの削除」を参照してください。

Amazon SES のサービスにリンクされたロールをサポートするリージョン

SES サービスが利用可能なすべてのリージョンでサービスにリンクされたロールの使用がサポートされているとは限りません。AWSServiceRoleForAmazonSES ロールは、以下のリージョンで利用できます。

リージョン名	リージョン識別子	SES でのサポート
米国東部 (バージニア北部)	us-east-1	はい
米国東部 (オハイオ)	us-east-2	可能
アジアパシフィック (シドニー)	ap-southeast-2	はい
アジアパシフィック (東京)	ap-northeast-1	可能
欧州 (フランクフルト)	eu-central-1	はい
欧州 (アイルランド)	eu-west-1	可能

Amazon SES でのログ記録とモニタリング

モニタリングは、Amazon SES の信頼性、可用性、パフォーマンスと AWS ソリューションを維持する上で重要な部分です。AWS は、Amazon SES を監視し、潜在的なインシデントに対して対応するツールを提供することができます。

- Amazon CloudWatch は、AWS のリソースおよび AWS で実行しているアプリケーションをリアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームの設定を行うことができます。詳細については、CloudWatch から Amazon SES イベントデータの取得およびCloudWatch を使用して評価モニタリングアラームを作成するを参照してください。

- AWS CloudTrail は、AWS アカウント により、またはそのアカウントに代わって行われた API コールや関連イベントを取得し、指定した Amazon S3 バケットにログファイルを配信します。AWS を呼び出したユーザーとアカウント、呼び出し元の IP アドレス、および呼び出し日時を特定できます。詳細については、「を使用した Amazon SESAPIコールのログ記録 AWS CloudTrail」を参照してください。
- Amazon SESのE メール送信イベント を使用すると、Eメール送信戦略を微調整できます。Amazon SES は、送信、配信、オープン、クリック、バウンス、苦情、拒否の数など、詳細な情報を捉えます。詳細については、「送信アクティビティのモニタリング」を参照してください。
- Amazon SESの評価メトリクス は、アカウントのバウンス率と苦情率を追跡します。詳細については、「送信者評価のモニタリング」を参照してください。

を使用した Amazon SESAPIコールのログ記録 AWS CloudTrail

Amazon SESは AWS CloudTrail、Amazon . CloudTrail captures のユーザー、ロール、または AWS のサービスによって実行されたアクションを記録するサービスである と統合されていますSES。 は Amazon をイベントSESとしてAPI呼び出します。キャプチャされた呼び出しには、Amazon SESコンソールからの呼び出しと、Amazon SESAPIオペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、Amazon の CloudTrail イベントなど、Amazon S3 バケットへのイベントの継続的な配信を有効にすることができますSES。 Amazon S3 証跡を設定しない場合でも、 CloudTrail コンソールのイベント履歴で最新のイベントを表示できます。によって収集された情報を使用して CloudTrail、Amazon に対するリクエストSES、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

設定と有効化の方法などの詳細については CloudTrail、「 AWS CloudTrail ユーザーガイド」を参照してください。

の Amazon SES 情報 CloudTrail

CloudTrail アカウントを作成する AWS アカウント と、 は で有効になります。Amazon でサポートされているイベントアクティビティが発生するとSES、そのアクティビティは 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。で最近のイベントを表示、検索、ダウンロードできます AWS アカウント。詳細については、「イベント履歴を使用した CloudTrail イベントの表示」を参照してください。

Amazon のイベントなど AWS アカウント、 のイベントの継続的な記録についてはSES、証跡を作成します。証跡により CloudTrail 、 はログファイルを Amazon S3 バケットに配信できます。デ

フォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントを口グに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、 CloudTrail 口グで収集されたイベントデータをより詳細に分析し、それに基づいて行動するように、他の AWS サービスを設定できます。詳細については、次を参照してください:

- 証跡の作成のための概要
- CloudTrail サポートされているサービスと統合
- の Amazon SNS 通知の設定 CloudTrail
- 複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからのログファイルの受信 CloudTrail

Amazon SES は、 SESAPIリファレンスおよび SES API v2 リファレンスにリストされているすべてのアクションを、以下のノートボックスにリストされているアクションを除き、 CloudTrail ログファイルのイベントとしてログ記録をサポートしています。

> **Note**
>
> Amazon SESは管理イベントを に配信します CloudTrail。管理イベントには、 内のリソースの作成と管理に関連するアクションが含まれます AWS アカウント。Amazon ではSES、管理イベントには ID や受信ルールの作成や削除などのアクションが含まれます。
> 管理イベントは、データイベントとは異なります。データイベントは、 内のデータへのアクセスと操作に関連するイベントです AWS アカウント。Amazon ではSES、データイベントには E メールの送信などのアクションが含まれます。
> Amazon は管理イベントSESのみを配信するため CloudTrail、以下のイベントは記録されません CloudTrail。
>
> - SendEmail
> - SendRawEmail
> - SendTemplatedEmail
> - SendBulkTemplatedEmail
>
> E メール送信に関連するイベントを記録するイベント発行を使用できます。詳細については、「Amazon SES イベント発行を使用して E メール送信をモニタリングする」を参照してください。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストが root または AWS Identity and Access Management （ IAM) ユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、CloudTrail userIdentity「 要素」を参照してください。

例: Amazon SESログファイルエントリ

証跡は、指定した Amazon S3 バケットにイベントをログファイルとして配信できるようにする設定です。 CloudTrail ログファイルには 1 つ以上のログエントリが含まれます。イベントは任意のソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。 CloudTrail ログファイルはパブリックAPIコールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、 `DeleteIdentity`および `VerifyEmailIdentity`アクションを示す CloudTrail ログエントリを示しています。

```
{
  "Records":[
    {
      "awsRegion":"us-west-2",
      "eventID":"0ffa308d-1467-4259-8be3-c749753be325",
      "eventName":"DeleteIdentity",
      "eventSource":"ses.amazonaws.com",
      "eventTime":"2018-02-02T21:34:50Z",
      "eventType":"AwsApiCall",
      "eventVersion":"1.02",
      "recipientAccountId":"111122223333",
      "requestID":"50b87bfe-ab23-11e4-9106-5b36376f9d12",
      "requestParameters":{
        "identity":"amazon.com"
      },
      "responseElements":null,
      "sourceIPAddress":"192.0.2.0",
      "userAgent":"aws-sdk-java/unknown-version",
```

```
      "userIdentity":{
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "accountId":"111122223333",
        "arn":"arn:aws:iam::111122223333:root",
        "principalId":"111122223333",
        "type":"Root"
      }
    },
    {
      "awsRegion":"us-west-2",
      "eventID":"5613b0ff-d6c6-4526-9b53-a603a9231725",
      "eventName":"VerifyEmailIdentity",
      "eventSource":"ses.amazonaws.com",
      "eventTime":"2018-02-04T01:05:33Z",
      "eventType":"AwsApiCall",
      "eventVersion":"1.02",
      "recipientAccountId":"111122223333",
      "requestID":"eb2ff803-ac09-11e4-8ff5-a56a3119e253",
      "requestParameters":{
        "emailAddress":"sender@example.com"
      },
      "responseElements":null,
      "sourceIPAddress":"192.0.2.0",
      "userAgent":"aws-sdk-java/unknown-version",
      "userIdentity":{
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "accountId":"111122223333",
        "arn":"arn:aws:iam::111122223333:root",
        "principalId":"111122223333",
        "type":"Root"
      }
    }
  ]
}
```

Amazon Simple Email Service のコンプライアンス検証

Amazon Simple Email Serviceのセキュリティとコンプライアンスは、AWS のさまざまなコンプライアンスプログラムのパートとして、第三者の監査機関によって評価されます。これらのプログラムには、SOC、PCI、FedRAMP、HIPAA などがあります。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスプログラムによる対象範囲内の AWS サービス」を参照してください。一般的な情報については、「AWS コンプライアンスプログラム」を参照してください。

AWS Artifact を使用して、サードパーティーの監査レポートをダウンロードできます。については、AWS Artifactのレポートのダウンロードを参照してください。

Amazon Simple Email Serviceを使用する際のコンプライアンス責任は、データの機密性、企業のコンプライアンス目標、適用法規や規則によって決まります。AWS ではコンプライアンスに役立つ以下のリソースを用意しています。

- Security and Compliance Quick Start Guides – これらのデプロイガイドには、アーキテクチャ上の考慮事項の説明と、AWSでセキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイするためのステップが記載されています。
- HIPAA セキュリティおよびコンプライアンスホワイトペーパーのアーキテクチャの設計 - このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWSコンプライアンスのリソース– このワークブックおよびガイドのコレクションは、お客様の業界と拠点に適用されるものである場合があります。
- AWS Configデベロッパーガイドのルールでのリソースの評価 – AWS Configは、リソース設定が、社内のプラクティス、業界のガイドラインそして規制にどの程度適合しているのかを評価します。
- AWS Security Hub– AWSのこのサービスは、 AWS内でのユーザーのセキュリティ状態に関する包括的な見解を提供し、業界のセキュリティ標準、およびベストプラクティスに対するコンプライアンスを確認するために役立ちます。

Amazon Simple Email Serviceの耐障害性

AWS の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として構築されています。リージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立および隔離されたアベイラビリティーゾーンがあります。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、耐障害性、および拡張性が優れています。

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 Simple Email Service での基盤セキュリティ

Amazon Simple Email Service はマネージドサービスとして、AWS グローバルネットワークセキュリティによって保護されています。AWSセキュリティサービスと AWS がインフラストラクチャを保護する方法については、「AWSクラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、セキュリティの柱 - AWS Well-Architected Frameworkのインフラストラクチャ保護を参照してください。

AWS が公開している API コールを使用し、ネットワーク経由で Amazon Simple Email Service にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2 は必須で TLS 1.3 がお勧めです。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token ServiceAWS 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。

Amazon SES を用いた VPC エンドポイントの設定

Amazon SES のお客様の多くは、会社のポリシーによって、内部システムからパブリックインターネットに接続することを制限されています。これらのポリシーは、パブリック Amazon SES エンドポイントの使用を防ぎます。

同様のポリシーがある場合は、Amazon Virtual Private Cloud を使用して、これらの制限内で作業できます。Amazon VPC を使うと、AWS クラウド の分離されたエリアに存在する仮想ネットワークに AWS リソースをデプロイできます。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 エンドポイントを経由して、安全かつスケーラブルな方法で Amazon VPC から SES に直接接続できます。インターフェイス VPC エンドポイントを使用すると、アウトバウンドトラフィックのファイアウォールを開く必要がないため、セキュリティ体制が強化されるだけでなく、Amazon VPC エンドポイントを使用する他の利点も得られます。

VPC エンドポイントを使用すると、SES へのトラフィックはインターネットに転送されることなく、Amazon ネットワーク内に留まるため、ネットワークトラフィックに対する可用性のリスクや

帯域幅の制限に影響されることなく、VPC を安全に SES に接続できます。インターネットゲートウェイを利用することなく、マルチアカウントインフラストラクチャ全体で SES を一元化し、これをサービスとして各アカウントに提供できます。

> **制約事項**
>
> - SES が VPC エンドポイントをサポートしていないアベイラビリティーゾーンは、`use1-az2`、`use1-az3`、`use1-az5`、`usw1-az2`、`usw2-az4`、`apne2-az4`、`cac1-az3`、`cac1-az4` です。
> - VPC 内で使用される SMTP エンドポイントは、アカウントで現在使用している AWS リージョンに限定されます。

Amazon VPC で SES を設定するウォークスルー例

前提条件

このセクションの手順を実行する前に、以下の手順を完了してください。

- 既存の仮想プライベートクラウド (VPC) を使用するか、新しい VPC を作成します。手順については、「Amazon VPC の使用を開始する」を参照してください。
- 後のステップで作成する VPC エンドポイントへの接続をテストするために、VPC で Amazon EC2 インスタンスを起動します。詳細については、「デフォルト VPC」を参照してください。

> **Note**
>
> SES の VPC エンドポイントは任意のリソースで使用できますが、テスト方法を簡単にするために、この例では EC2 インスタンスをリソースとして使用します。Amazon EC2 はデフォルトでポート 25 経由の E メール トラフィックを制限するため、TCP 25 以外の別のポート (TCP 465、587、2465、2587 など) を使用する必要があります。

Amazon VPC での SES の設定

SES で使用する VPC エンドポイントを設定するプロセスは、いくつかの個別のステップで構成されています。最初に、SMTP ポートとの通信をインスタンスに 許可するセキュリティグループを作成します。次に Amazon SES の VPC エンドポイントを作成します。最後に VPC エンドポイントへの接続をテストして、正しく設定されていることを確認します。

ステップ 1: セキュリティグループを作成する

このステップでは、これから作成する VPC インターフェイスエンドポイントとの通信を Amazon EC2 インスタンスに許可するセキュリティグループを作成します。

セキュリティグループを作成するには

1. Amazon EC2 コンソールのナビゲーションペインで、[ネットワークとセキュリティ] の下にある [セキュリティグループ] を選択します。
2. [セキュリティグループの作成] を選択します。
3. [基本的な詳細] で、次の操作を行います。
   - [セキュリティグループ名] に、セキュリティグループの一意の名前を入力します。
   - [説明] に、セキュリティグループの目的を説明するテキストを入力します。
   - [VPC] で、Amazon SES で使用する VPC を選択します。
4. [インバウンドルール] で、[ルールの追加] を選択します。
5. 新しいインバウンドルールで、次の操作を実行します。
   - [タイプ] で [カスタム TCP] を選択します。
   - [ポート範囲] に、E メールの送信に使用するポート番号を入力します。ポート番号として、**465**、**587**、**2465**、**2587** のいずれでも使用できます。
   - [Source タイプ] で、[Custom] を選択します。
   - [ソース] に、SES サービスと VPC エンドポイント経由で通信するリソースを含むプライベート IP CIDR 範囲またはその他のセキュリティグループ ID を入力します。
   - (アクセスを許可する CIDR 範囲またはセキュリティグループごとに、手順 4 ~ 5 を繰り返します。)
6. 完了したら、[Create security group] を選択します。

ステップ 2: VPC エンドポイントを作成する

Amazon VPC では、VPC エンドポイント を使用して、サポートされている AWS サービスに VPC を接続できます。この例では、Amazon EC2 セキュリティグループが Amazon SES に接続できるように、Amazon VPC を設定します。

VPC エンドポイントを作成するには

1. Amazon VPC コンソール (https://console.aws.amazon.com/vpc/) を開きます。

2. [仮想プライベートクラウド] 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] を選択して、[エンドポイントの作成] ページを開きます。
4. (オプション) [Endpoint settings] (エンドポイントの設定) パネルで、[Name tag] (名前タグ) フィールドにタグを作成します。
5. [Service category] (サービスカテゴリ) で、[AWS のservices] (サービス) を選択します。
6. [Services] (サービス) の検索バーで smtp をフィルタリングし、そのラジオボタンを選択します。
7. [VPC] パネルで検索バー内をクリックし、リストボックスから VPC を選択します (「the section called “前提条件”」を参照)。
8. [Subnets] (サブネット) パネルで、[Availability Zones] (アベイラビリティーゾーン) と [Sbnet ID] (サブネット ID) を選択します。

   > **Note**
   >
   > 次の [アベイラビリティーゾーン] では、Amazon SES は VPC エンドポイントをサポートしていません。`use1-az2`、`use1-az3`、`use1-az5`、`usw1-az2`、`usw2-az4`、`apne2-az4`、`cac1-az3`、および `cac1-az4`。

9. [セキュリティグループ] パネルで、前に作成したセキュリティグループを選択します。
10. (オプション) [タグ] パネルで、1 つ以上のタグを作成できます。
11. [Create endpoint (エンドポイントの作成)] を選択します。Amazon VPC がエンドポイントを作成している間、5 分ほど待ちます。エンドポイントの準備ができると、[ステータス] 列の値が [Available] (利用可能) に変わります。

(オプション) ステップ 3: VPC エンドポイントへの接続をテストする

VPC エンドポイントの設定プロセスが完了したら、VPC エンドポイントが正しく設定されていることを確認するために、接続をテストできます。接続のテストには、ほとんどのオペレーティングシステムに用意されているコマンドラインツールを使用できます。

VPC エンドポイントへの接続をテストするには

1. email-smtp VPC エンドポイントを作成したのと同じ VPC で Amazon EC2 インスタンスを起動します。

Linux インスタンスに接続する方法については、「Amazon EC2 ユーザーガイド」の「Linux インスタンスへの接続」を参照してください。

Windows インスタンスへの接続の詳細については、「Amazon EC2 ユーザーガイド」の「開始方法のチュートリアル」を参照してください。

2. 例えば、SES SMTP インターフェイスを使用してテスト E メールを送信します。

> **Note**
>
> Amazon SES 経由で E メールを送信する前に、メールアドレスまたはドメインを検証する必要があります。検証の方法については、「Amazon での ID の作成と検証 SES」を参照してください。

Amazon SESの問題に関するラブルシューティング

このセクションは、問題が発生した場合に役に立つ次のトピックから構成されます。

- ドメインの確認時に発生する可能性がある問題の詳細については、ドメインおよび E メールアドレス検証の問題 を参照してください。
- DKIM 関連の問題に対する解決策については、Amazon SES の DKIM 問題のトラブルシューティング を参照してください。
- E メール送信時に発生する可能性がある一般的な配信に関連する問題のリストとその対処方法については、Amazon SES 配信の問題 を参照してください。
- Amazon SES から送信された E メールの受信者側での表示に関する問題については、「Amazon SES から受け取った E メールに関する問題」を参照してください。
- バウンス、苦情、および配信通知に関する問題の解決方法については、Amazon SES 通知の問題を参照してください。
- Amazon SES で E メールを送信するときに発生する可能性があるエラーのリストについては、「Amazon SES の E メール送信エラー」を参照してください。
- API または SMTP インターフェイスを使用して Amazon SES への複数の呼び出しを行うときに E メール送信速度を向上させるためのヒントについては、「Amazon SES のスループットを上げる」を参照してください。
- Simple Mail Transfer Protocol (SMTP、簡易メール転送プロトコル) インターフェイスを介して Amazon SES を使用するときに発生する可能性がある一般的な問題の解決方法と、Amazon SES から返される SMTP 応答コードの一覧については、Amazon SES SMTP の問題 を参照してください。
- Amazon SESのAPI v2から返されるエラーコードのリストについては、「よくあるエラー」を参照してください。
- レビューの送信プロセスに関連する一般的な問題の説明とその対処方法については、Amazon SES 送信レビュープロセスに関するよくある質問 を参照してください。
- DNS ベースのブラックホールリスト (DNSBL) が Amazon SES を使用した送信にどのように影響を与えるかについては、「DNS ブラックホールリスト (DNSBL) に関するよくある質問」を参照してください。

Amazon SES API を直接呼び出している場合は、Amazon Simple Email Service API リファレンスを参照して、発生する可能性がある HTTP エラーを確認してください。

> **Note**
>
> テクニカルサポートをリクエストする必要がある場合、このデベロッパーガイドの任意のページに記載されたフィードバックリンクは使用しないでください。このフォームは AWS サポートではなく、AWS ドキュメントチームに送信されます。代わりに、お問い合わせページで、利用可能なさまざまなサポートオプションを確認してください。

内容

一般的な Amazon SES の問題

このページで情報は、Amazon SES を使用する上で遭遇する可能性のある問題を説明しており、診断に役立ちます。

行った変更がすぐに表示されない

世界中のデータセンター内のコンピュータを介してアクセスされるサービスとして、Amazon SES は、結果整合性と呼ばれる分散コンピューティングモデルを採用しています。Amazon SES (または他のAWSサービス) で行った変更は、すべての可能なエンドポイントから認識されるまでに時間がかかります。この遅延は、サーバー間および世界中のリージョン間でのデータ送信にかかる時間から発生している場合もあります。ほとんどのケースでは、この遅延が数分以上かかることはありません。

遅延が生じる可能性のあるエリアには以下が含まれます。

- 設定セットの作成と変更 - 設定セットを作成または変更すると (例えば、専用 IP プールを既存の設定セットと関連付ける場合)、作成や変更の時点から変更が有効になるまでに少しの間があります。
- イベント発生先の作成と変更 - イベント発生先の作成または変更をすると (たとえば、Amazon SES に E メールの送信データを別のAWSのサービス に送信するように伝える場合)、イベント発生先の作成または変更をした時点から E メールの送信イベントが実際に指定した先に届くまでの間に遅延が生じることがあります。

ドメインおよび E メールアドレス検証の問題

Amazon SES でドメインまたは E メールアドレスを検証するには、Amazon SES コンソールまたは Amazon SES API を使用してプロセスを開始します。このセクションには、認証プロセスに関する問題の解決に役立つ情報が含まれます。

> **Note**
>
> 以下の手順での DNS レコードへの参照は、使用した DKIM の形式に応じて、CNAME レコードまたは TXT レコードを参照している場合があります。Easy DKIM は CNAME レコードを使用し、Bring Your Own DKIM (BYODKIM) は TXT レコードを使用します。Easy DKIM または BYODKIM それぞれの詳細な検証手順が説明されています。

ドメインの検証に関する一般的な問題

「the section called “ドメイン ID の検証”」の手順を使用してドメインを確認しようとしたときに問題が発生した場合は、次に示す考えられる原因と解決策を確認してください。

- 自分が所有していないドメインを確認しようとしています – 自分が所有していないドメインを確認することはできません。例えば、Amazon SES 経由で、gmail.comドメインから E メールを送信する場合は、そのE メールアドレスを明確に確認する必要があります。gmail.com ドメイン全体を確認することはできません。
- プライベートドメインの検証を試みている場合 – DNS レコードをパブリック DNS 上で解決できない場合は、ドメインを検証できません。
- DNS プロバイダーにより、DNS レコード名にアンダースコアを含めることが許可されない – 少数の DNS プロバイダーは、レコード名にアンダースコア (_) 文字を含めることを許可していません。ただし、DKIM レコード名には下線が必要となります。DNS プロバイダーがレコード名に下線を含めることを許可しない場合、プロバイダーのカスタマーサポートにお問い合わせください。

- DNS プロバイダーによって DNS レコードの末尾にドメイン名が追加された – 一部の DNS プロバイダーは、ドメインの名前を自動的に DNS レコードの属性名に追加します。例えば、属性名が _domainkey.example.com であるレコードを作成した場合、プロバイダーがドメイン名を追加し、_domainkey.example.com.example.com となります)。ドメイン名の重複を避けるには、DNS レコードを入力するときに、ドメイン名の末尾にピリオドを追加します。このステップでは、ドメイン名をレコードに追加する必要はないことを DNS プロバイダーに伝えます。
- DNS プロバイダが DNS レコードの値を変更した場合 – 一部のプロバイダは、小文字のみを使用するように DNS レコードの値を自動的に変更します。Amazon SES がドメインを検証するのは、ドメインの所有権の検証プロセスをスタートしたときに Amazon SES が提供した値と属性値が正確に一致する検証レコードが検出された場合のみです。ドメインの DNS プロバイダーが小文字のみを使用するように DNS レコード値を変更した場合は、DNS プロバイダーにお問い合わせください。
- 同じドメインを複数回確認したい - 異なる地域で送信しようとしている場合や、同じドメインを使って複数の AWS アカウントから送信しようとしている場合は、複数回ドメインを確認する必要があります。DNS プロバイダーが同じ属性名の複数の DNS レコードを持つことを許可しない場合、2 つのドメインを検証できることがあります。DNS プロバイダーによって許可される場合、同じ DNS レコードに複数の属性値を割り当てることができます。例えば、DNS が Amazon Route 53 によって管理されている場合、以下のステップを実行して、同じ CNAME レコードに対して複数の値をセットアップできます。

  1. Route 53 コンソールで、最初のリージョンのドメインを検証したときに作成した CNAME レコードを選択します。
  2. [Value (値)] ボックスで、既存の属性値の末尾に移動し、Enter キーを押します。
  3. 追加のリージョンの属性値を追加し、レコードセットを保存します。

  DNS プロバイダーが、同じ DNS レコードへの複数の値の割り当てを許可していない場合、DNS レコードの属性名の _domainkey で 1 回、属性名から削除された _domainkey で再度ドメインを検証することができます。このソリューションの欠点は、同じドメインを 2 回しか確認できないことです。

ドメイン検証設定の確認

以下の手順を使用して、Amazon SES ドメイン検証 DNS レコードが適切に DNS サーバーに発行されたことを確認できます。この手順では、Windows および Linux で使用できる nslookup ツールを使用します。Linux では、dig を使用することもできます。

これらの手順に示すコマンドは、Windows 7 で実行されています。使用されているサンプルのドメインは、CNAME レコードを使用する Easy DKIM で設定された ses-example.com です。

この手順では、最初にドメインにサービスを提供する DNS サーバーを見つけます。次に、これらのサーバーに対して、CNAME レコードを表示するためのクエリを実行します。ドメインにサービスを提供する DNS サーバーに対してクエリを実行する理由は、これらのサーバーには他の DNS サーバーに伝達されるまでに時間のかかるドメインの最新情報が格納されているためです。

ドメイン検証 CNAME レコードが DNS サーバーに公開されていることを検証するには

1. 次のステップを実行して、ドメインのネームサーバーを見つけます。

   a. コマンドラインに移動します。Windows 7 でコマンドラインに移動するには、[Start] を選択し、cmd と入力します。Linux ベースのオペレーティングシステムでは、ターミナルウィンドウを開きます。

   b. コマンドプロンプトで、次のように入力します。ここで、<domain> はドメインを示します。これにより、ドメインにサービスを提供しているすべてのネームサーバーが表示されます。

      ```
      nslookup -type=NS <domain>
      ```

      ドメインが ses-example.com の場合、このコマンドは次のようになります。

      ```
      nslookup -type=NS ses-example.com
      ```

      コマンドの出力に、ドメインにサービスを提供しているネームサーバーのリストが表示されます。次のステップでは、これらのサーバーの 1 つに対してクエリを実行します。

2. 以下のステップを実行して、CNAME レコードが適切に発行されていることを検証します。Amazon SES は Easy DKIM 認証のために 3 つの CNAME レコードを生成するため、3 つのそれぞれについて以下の手順を繰り返します。

   a. コマンドプロンプトで、次のように入力します。ここで、<random string> は SES で生成された CNAME 名、<domain> はドメイン、<name server> はステップ 1 で見つけたネームサーバーの 1 つを示します。

      ```
      nslookup -type=CNAME <random string>_domainkey.<domain> <name server>
      ```

ses-example.com の例で、ステップ 1。で見つけたネームサーバーが ns1.name-server.net で、SES によって生成された <random string> が 4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz の場合は、次のように入力します。

```
nslookup -type=CNAME 4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz_domainkey.ses-example.com
 ns1.name-server.net
```

b. コマンドの出力の `canonical name =` に続く文字列が、Amazon SES コンソールのアイデンティティリストでドメインを選択すると表示される CNAME 値と一致することを検証します。

この例では、4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz_domainkey.ses-example.com の下で、値が 4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz.dkim.amazonses.com の CNAME レコードを探しています。レコードが正しく発行されている場合、次のようなコマンド出力が得られます。

```
4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz_domainkey.ses-example.com canonical name =
 "4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz.dkim.amazonses.com"
```

E メール認証のよくある問題

- 確認 E メールが届かない – E メールアドレス ID の検証 の手順を完了しても数分以内に確認 E メールが届かない場合は、以下のステップを実行します。
  - 認証しようとしている E メールアドレスのスパムまたは迷惑メールフォルダを確認します。
  - 認証しようとしている E メールアドレスで E メールを受信できることを確かめます。別の E メールアドレス (個人の E メールアドレスなど) を使用して、認証しようとしているアドレスに、テスト E メールを送信します。
  - Amazon SES コンソールの認証されたアドレスのリストをチェックします。認証しようとしている E メールアドレスに間違いがないことを確認します。

Amazon SES の DKIM 問題のトラブルシューティング

このセクションでは、Amazon SES で DKIM 認証を設定するときに発生する可能性のある問題について説明します。DKIM を設定しようとしたときに問題が発生した場合は、以下の考えられる原因と解決策を確認してください。

DKIM を正常にセットアップしたが、メッセージが DKIM 署名されていない

Easy DKIM または BYODKIM を使用してドメインの DKIM を設定したが、送信するメッセージが DKIM 署名されていない場合は、次の手順を実行します。

- 適切な ID に対して DKIM が有効になっていることを確認します。Amazon SES コンソールで ID に対して DKIM を有効にするには、ID リストで E メールドメインを選択します。ドメインの詳細ページで、[DKIM] を展開し、[Enable] を選択して DKIM を有効にします。
- 同じドメインの確認済み E メールアドレスから送信していないことを確認します。ドメインに DKIM を設定した場合は、そのドメインから送信するすべてのメッセージが DKIM 署名されます。ただし、個別に確認した E メールアドレスは除外されます。個別に確認された E メールアドレスには、別の設定が使用されます。たとえば、ドメイン example.com に DKIM を設定し、E メールアドレス mary@example.com を個別に検証した場合 (ただし、アドレスに DKIM を設定していない場合)、mary@example.com から送信する E メールは DKIM 認証なしで送信されます。この問題は、アカウントの ID リストから E メールアドレス ID を削除することで解決できます。
- 複数の AWS リージョンで同じ ID を使用する場合は、リージョンごとに個別に DKIM を設定する必要があります。同様に、同じドメインを複数の AWS アカウントで使用する場合は、アカウントごとに DKIM を設定する必要があります。特定のリージョンやアカウントに必要な DNS レコードを削除すると、Amazon SES はそのリージョンやアカウントの DKIM 署名を無効にします。DKIM 署名が無効になると、Amazon SES は E メールで通知を送信します。

Amazon SES コンソールのユーザードメインの DKIM 詳細は DKIM: waiting on sender verification... を示します。DKIM 検証ステータス: 検証の保留中。

簡単 DKIM または BYODKIM - 自分のDKIM を使用する の手順を実行してドメインの DKIM を設定しても、Amazon SES コンソールに DKIM 検証が保留中であることが示される場合は、次の手順を実行します。

- 最長で 72 時間待ちます。まれに、DNS レコードが Amazon SES に表示されるまでに時間がかかることがあります。
- CNAME レコード (Easy DKIM の場合) または TXT レコード (BYODKIM の場合) に正しい名前が使用されていることを確認します。DNS プロバイダーによっては、作成するレコードにドメイン名が自動的に付加される場合があります。たとえば、`example._domainkey.example.com` の [Name] でレコードを作成した場合、DNS プロバイダーはこの文字列の末尾にドメインの名前を追加して、`example._domainkey.example.com.example.com` という名前にします。詳細については、DNS プロバイダーのドキュメントを参照してください。

Amazon SES から、DKIM の設定が取り消された (または取り消される) という E メールが届きます。

これは、Amazon SES が DNS サーバー上で必要な CNAME レコード (Easy DKIM を使用している場合) または必要な TXT レコード (BYODKIM を使用している場合) を検出できなくなっていることを意味します。通知 E メールには、DNS レコードを再発行することで DKIM のセットアップステータスが失効して DKIM 署名が無効化されるのを回避できる残りの期間が記載されています。DKIM のセットアップが失効した場合は、DKIM のセットアップ手順を最初から繰り返す必要があります。

BYODKIM を設定しようとすると、DKIM 検証プロセスに失敗します。

プライベートキーが正しい形式を使用していることを確認してください。プライベートキーは PKCS #1 または PKCS #8 形式で、1024 または 2048 ビットの RSA 暗号化を使用する必要があります。さらに、プライベートキーには base64 エンコードを適用する必要があります。

BYODKIM のセットアップ中にドメインのパブリックキーを指定しようとすると **`BadRequestException`** エラーが表示されます。

`BadRequestException` エラーが発生した場合は、次の操作を行います。

- パブリックキーに指定するセレクタに、1 ~ 63 文字の英数字が含まれていることを確認します。セレクタに、ピリオド、その他の記号、句読点を含めることはできません。
- パブリックキーからヘッダー行とフッター行を削除し、パブリックキーからすべての改行を削除したことを確認します。

Easy DKIM を使用すると、DNS サーバーは Amazon SES DKIM CNAME レコードを正常に返しますが、ドメイン検証 TXT レコードに対しては **`SERVFAIL`** を返します。

DNS プロバイダーが CNAME レコードをリダイレクトできない場合があります。TXT レコードの Amazon SES と ISP クエリ。DKIM の仕様に準拠するには、DNS サーバーが CNAME レコードに対するクエリだけでなく、TXT レコードに対するクエリにも応答できる必要があります。DNS プロバイダーが TXT レコードのクエリに応答できない場合は、DNS ホスティングプロバイダーとして Route 53 を使用することもできます。

E メールに 2 つの DKIM 署名が含まれている

追加の DKIM 署名 (`d=amazonses.com` を含む) は、Amazon SES によって自動的に追加されます。このメッセージは無視できます。

Amazon SES 配信の問題

Amazon SES へのリクエストが成功すると、メッセージは多くの場合、すぐに送信されます。ただし場合によっては、短い遅延が生じることがあります。いずれにせよ、E メールが送信されることは保証されます。

ただし、Amazon SES からメッセージが送信されるときにいくつかの要因で正常な配信が妨げられることがあります。場合によっては、送信したメッセージが到着しないことを知って初めて配信が失敗したことに気付く結果になります。この状況を解決するには、次のプロセスを使用します。

E メールが到着しない場合は、次の操作を試してください。

- 対象の E メールに SendEmail または SendRawEmail リクエストを行っていて、成功を示す応答を受け取っていることを確認します これらのリクエストをプログラムで行っている場合は、ソフトウェアのログを調べて、プログラムがリクエストを実行し、成功を示す応答を受け取っていることを確認します。
- ブログ記事「SES を使用してメールを送信したときに遅延が発生する可能性がある 3 つの場所」を参照してください。実際は、E メールが配信されていないのではなく、遅延が生じている可能性があります。
- 送信者の E メールアドレス (「From」アドレス) が有効であることを確認します。バウンスメッセージが送信される「Return-Path」アドレスも確認します。メールがバウンスされた場合は、説明のエラーメッセージが返されます。
- [AWSService Health Dashboardにアクセスして、Amazon SES に既知の問題がないことを確認してください。
- E メールの受取人または受取人の ISP に問い合わせます。受取人が正しい E メールアドレスを使用していることを確認し、受取人の ISP において配信に関する既知の問題があるかどうかを問い合わせます。また、実際は到着した E メールがスパムとしてフィルタリングされていないかどうかを確認します。
- 有償の AWS サポートプランを契約している場合は、新しい技術サポートケースを開くことができます。お問い合わせの際は、SendEmail または SendRawEmail 応答から返されたリクエスト ID またはメッセージ ID に加えて、該当する受取人のアドレスをお知らせください。
- 原因が実際は配信時の遅延であり永続的なエラーでないか、しばらく観察します。スパム発信者への対策として、一部の ISP は不明な送信メールサーバーからの受信メッセージを一時的に拒否します。グレーリストと呼ばれるこのプロセスにより配信時に遅延が生じることがあります。Amazon SES はこれらのメッセージを再試行します。グレーリストが原因となっている場合は、再試行される E メールを ISP が許可することができます。

- お客様の利益を最優先にしておりますが、メッセージの配信性能に影響するような状況が発生する場合があります。E メールメッセージを対象者に確実に届けるためには、「the section called “良好な送信者評価の維持”」を参照してください。

Amazon SES から受け取った E メールに関する問題

このセクションでは、Amazon SES から送信された E メールを受信するときに発生する可能性がある一般的な問題について説明します。

E メールクライアントでは、E メールのソースとして「amazonses.com 経由で送信されました」と表示されます

一部のメールクライアントでは、送信者のドメインが E メールの送信元のドメイン (この場合は amazonses.com) と一致しない場合、「via」ドメインが表示されます。詳細については、Gmail サポートウェブサイトの 「送信者の名前の横にあるその他の情報」を参照してください。または、ドメインキーアイデンティファイドメール (DKIM) を設定することもできます。DKIM を使用してメールを認証する場合、通常、E メールクライアントには "via" ドメインが表示されません。これは、DKIM の署名には、送信元として示されているドメインがメールの送信元として示されるためです。DKIM の設定の詳細については、「Amazon DKIMでの による E メールの認証 SES」を参照してください。

> **Note**
>
> SES ユーザーからスパムやその他の未承諾メッセージを受信した場合は、E メールクライアントのスパム報告ツールを使用し、[お問い合わせ] に一覧表示されている、SES メールの不正使用を報告する手順に従ってください。

メッセージに文字化けまたは意味のない文字が含まれています

メッセージに ASCII 文字セットに含まれていない文字 (アクセント付きラテン文字、中国文字、アラビア文字など) が含まれている場合は、HTML 文字エンコーディングを使用してこれらの文字をエンコードする必要があります。ウェブベースのツールを使用して、E メール内の文字をエンコードできます。たとえば、Email On Acid ウェブサイトの HTML 文字コンバーター など。

または、MIME メッセージを自分で組み立てることもできます。MIME メッセージでは、メッセージに UTF-8 エンコーディングを使用するように指定できます。UTF-8 エンコーディングを使

用する場合、非 ASCII 文字をメッセージに直接使用できます。MIME メッセージの作成が完了したら、SendRawEmail API または SendMail API v2を使用して送信できます。

この問題の一般的な原因の 1 つは、Microsoft Word のスマート引用符機能です。Word からコンテンツを頻繁にコピーして E メールに貼り付ける場合は、この問題が発生する可能性があります。スマート引用符機能は、ストレート引用文字 (「...」) を中引用符文字 (「...」) に置き換えます。中引用符文字は標準 ASCII 文字ではありません。その結果、一部の E メールクライアントでは「??」として表記されます。あるいは「â€œ」などの文字群として表記されます。この問題を解決するには、Word でスマート引用符機能を無効にすることができます。または、前の段落の SendRawEmail ソリューションを使用することもできます。この機能を無効にする方法については、Microsoft Office サポートウェブサイトの「Word のスマート引用符」を参照してください。

Amazon SES 通知の問題

バウンス、苦情、または配信通知に関する問題が発生した場合は、以下の考えられる原因と解決方法を確認します。

- バウンス通知を Amazon SNS 経由で受け取ったが、どの受信者が通知と対応しているかわからない – 今後、バウンス通知を特定の受信者に関連付けるには、以下のオプションを使用できます。
  - Amazon SES は追加したカスタムメッセージ ID を保持しないので、Amazon SES が E メールを受け付けるときにお客様に渡す Amazon SES メッセージ ID と識別子のマッピングを保存します。
  - Amazon SES を呼び出すたびに、単一のメッセージを複数の受信者に送信する代わりに、単一の受取人に送信します。
  - バウンスに関するメッセージ全文をお客様に転送する、E メールによるフィードバックの転送を有効にすることができます。
- 苦情または配信の通知を Amazon SNS 経由または E メールによるフィードバックの転送で受け取ったが、どの受信者が通知と対応しているかわからない – 一部の ISP は、苦情通知を Amazon SES に渡す前に、苦情を送信した受信者の E メールアドレスを編集します。受信者の E メールアドレスを特定できるようにする最善の方法は、Amazon SES が E メールを受け付けるときにお客様に渡す Amazon SES メッセージ ID と識別子のマッピングを保存することです。Amazon SES は追加したカスタムメッセージ ID を保持しないことに注意してください。
- 自分が所有していない Amazon SNS トピックに通知が送信されるように設定したい - このトピックの所有者は、お客様のアカウントがこの所有者のトピックで `SNS:Publish` アクションを呼び出すことを許可する アクセスポリシーを設定する必要があります。IAM ポリシーを使用して Amazon SNS トピックへのアクセスをコントロールする方法については、Amazon Simple

Notification Service デベロッパーガイドの「Amazon SNS トピックへのアクセスの管理」を参照してください。

Amazon SES の E メール送信エラー

このトピックでは、Amazon SES 経由で E メールを送信するときに発生する可能性がある E メール送信に固有のエラーのタイプについて説明します。Amazon SES 経由でメールを送信しようとしたときに Amazon SES の呼び出しに失敗すると、Amazon SES からアプリケーションにエラーメッセージが返され、E メールは送信されません。このエラーメッセージがどのように表示されるかは、Amazon SES を呼び出す方法によって異なります。

- Amazon SES API を直接呼び出す場合は、Query アクションによってエラーが返されます。エラーは「`MessageRejected`」あるいは、Amazon Simple Email Service API リファレンスの「一般的なエラー」トピック内で指定されたエラーの一つである可能性があります。
- 例外をサポートするプログラミング言語を使用するAWSSDK を使用して Amazon SES を呼び出した場合、Amazon SES は例外を取り除きます。例外のタイプは、SDK とエラーによって異なります。例えば、Amazon SES`MessageRejectedException`の例外 (実際の名前は SDK によって異なります) または一般的な AWSの例外が取り除かれます。例外のタイプにかかわらず、例外のエラータイプとエラーメッセージからより多くの情報が得られます。
- SMTP インターフェイスを介して Amazon SES を呼び出した場合、エラーがどのように示されるかはアプリケーションによって異なります。アプリケーションによって、特定のエラーメッセージが表示される場合もあれば、表示されない場合もあります。Amazon SES から返される SMTP 応答コードのリストについては、「Amazon SES から返される SMTP 応答コード」を参照してください。

> **Note**
>
> E メール送信のための Amazon SES 呼び出しが失敗した場合、対象の E メールについては課金されません。

E メールを送信しようとしたときに Amazon SES がエラーを返す原因となる、Amazon SES に固有の問題のタイプを次に示します。これらのエラーは、Amazon Simple Email Service API Referenceの「一般的なエラー」トピックに指定されている`MalformedQueryString`などの一般的なAWSエラーとは別のエラーです。

- メールアドレスが検証されていません。以下のアイデンティティが、リージョン内でチェックできませんでした。リージョン: identity1、identity2、identity3。- Amazon SES で検証されていない E メールアドレスまたはドメインから E メールを送信しようとしています。このエラーは、「From」、「Source」、「Sender」、または「Return-Path」のアドレスに該当する場合があります。アカウントが Amazon SES サンドボックスにまだある場合は、Amazon SES のメールボックスシミュレーターから提供されているアドレスを除く、すべての受信者のアドレスも検証する必要があります。Amazon SES が失敗した ID をすべて表示できない場合は、エラーメッセージが省略符号で終了します。

  > **Note**
  >
  > Amazon SES は複数のAWS リージョンにエンドポイントを持ち、E メールアドレスの検証ステータスはAWS リージョンごとに別個に扱われます。使用するAWS リージョンの各送信者について、検証プロセスを完了する必要があります。

- Account is paused - アカウントによる E メール送信機能を一時停止します。Amazon SES コンソールには引き続きアクセスでき、ほとんどの操作を実行できます。ただし、E メールを送信しようとした場合、このメッセージが表示されます。

  アカウントの E メール送信機能を一時停止する場合、AWS アカウントに関連付けられている E メールアドレスに通知を送信します。詳細については、「the section called “送信レビュープロセスに関するよくある質問”」を参照してください。

- スロットリング - アプリケーションが 1 秒あたりに送信しようとしているメッセージが多すぎるか、過去 24 時間に送信した E メールが多すぎる可能性があります。このような場合は、エラーメッセージは次の例のようになります。

  - Daily message quota exceeded - 24 時間の期間に送信することが許可されたメッセージの最大数に達しました。日次クォータを超過した場合、次の 24 時間の期間にならないとそれ以上 E メールを送信できません。
  - Maximum sending rate exceeded - 送信を試みた 1 秒あたりの E メールの件数が、許可された最大送信レートを超えました。送信レートを超過した場合、E メールの送信を継続できますが、送信レートを引き下げる必要があります。詳細については、「AWSメッセージングとターゲティングブログ」の「"Throttling – Maximum sending rate exceeded (スロットリング – 最大送信レートの超過)" エラーの対処法」を参照してください。
  - Sigv2 SMTP の最大送信率の超過 — 2019 年 1 月 10 日より前に作成された SMTP 認証情報を使用してメッセージを送信しようとしています。SMTP 認証情報は古いバージョンのAWS署名を使用して作成されています。セキュリティのため、この日付よりも前に作成した認証情報を削除

して、新しい認証情報に置き換える必要があります。古い認証情報は、IAM コンソールを使用して削除できます。認証情報ファイルの作成の詳細については、「the section called “SMTP 認証情報の取得”」を参照してください。

送信アクティビティを定期的に監視して、送信クォータにどれだけ近づいているかを確認する必要があります。詳細については、「Amazon SES 送信クォータのモニタリング」を参照してください。送信クォータに関する一般的な情報については、「Amazon SES 送信制限の管理」を参照してください。送信クォータを引き上げる方法については、「Amazon SES 送信クォータの引き上げ」を参照してください。

> **Important**
>
> スロットリングエラーを説明するエラーテキストが日次クォータまたは最大送信レートの超過に関係ない場合は、システム全体の問題が原因で送信機能が制限されている可能性があります。サービスステータスについては、「AWSService Health Dashboardを参照してください。

- There are no recipients specified – 受信者が指定されていません。
- There are non-ASCII characters in the email address - E メールアドレス文字列は 7 ビット ASCII である必要があります。送信先または送信元の E メールアドレス内で、ドメインの部分に Unicode 文字が含まれる場合は、Punycode を使用してドメインをエンコードする必要があります。Punycode は E メールアドレスのローカル部分 (@ 記号の前の部分) では許可されていません。また、「差出人」名にも許可されていません。「差出人」名に Unicode 文字を使用する場合は、Amazon SES API v2 を使用した raw E メールの送信 に説明されているとおりに MIME encoded-word 構文を使用して「差出人」名をエンコードする必要があります。Punycode の詳細については、RFC 3492 を参照してください。
- Mail FROM domain is not verified - Amazon SES は、指定された MAIL FROM ドメインを使用するために必要な MX レコードを読み取ることができませんでした。カスタム MAIL FROM ドメインの設定については、カスタムMAILFROMドメインの使用 を参照してください。
- Configuration set does not exist - 指定した設定セットが存在しません。設定セットは、メール送信イベントの発行に使用するオプションのパラメータです。詳細については、「Amazon SES イベント発行を使用して E メール送信をモニタリングする」を参照してください。

Amazon SES のスループットを上げる

E メールを送信するときは、最大送信レートで許可される頻度で Amazon SES を呼び出すことができます。(最大送信レートの詳細については、「Amazon SES 送信制限の管理」を参照してください)。ただし、それぞれの Amazon SES の呼び出しが完了するまで時間がかかります。

Amazon SES API または SMTP インターフェイスを使用して Amazon SES を複数回呼び出す場合は、次のヒントを考慮するとスループットを高めることができます。

- 現在のパフォーマンスを測定してボトルネックを識別する - 考えられるパフォーマンステストには、アプリケーションのコードループ内で複数のテスト E メールをできる限り早く送信する操作が含まれます。各 `SendEmail` リクエストのラウンドトリップレイテンシーを測定します。次に、同じマシン上でアプリケーションの追加インスタンスを増分的に起動して、ネットワークレイテンシーへの影響を調べます。また、このテストを複数のマシンおよび異なるネットワークで実行すると、考えられるあらゆるマシンリソースボトルネックや存在する可能性があるネットワークボトルネックを特定するのに役立ちます。
- (API のみ) 永続的な HTTP 接続の使用を検討する - 永続的な HTTP 接続を使用して、API リクエストごとに別個の新しい HTTP 接続を確立するオーバーヘッドを回避します。つまり、複数の API リクエストに対して同じ HTTP 接続を再利用します。
- 複数のスレッドの使用を検討する — アプリケーションが単一のスレッドを使用する場合、アプリケーションコードは Amazon SES API を呼び出し、API 応答を同期的に待機します。通常、E メール送信では I/O 負荷が高いため、複数のスレッドからこの操作をする方がよりスループットが向上します。任意の数の実行スレッドを使用して、同時にメールを送信できます。
- 複数のプロセスを使用することを検討する - 複数のプロセスを使用すると、Amazon SES に対してより多くの同時アクティブ接続を持つことになるため、スループットが向上します。たとえば、目的のメールを複数のバケットにセグメント化した後、E メール送信スクリプトの複数のインスタンスを同時に実行できます。
- ローカルメールリレーを使用することを検討する - アプリケーションは、ローカルメールサーバーにメッセージを迅速に送信できます。次に、ローカルメールサーバーを使用して、メッセージをバッファ処理し、非同期的に Amazon SES に送信します。一部のメールサーバーでは、同時配信がサポートされています。つまり、アプリケーションがメールサーバーに送る E メールがシングルスレッドで生成される場合でも、メールサーバーから Amazon SES に E メールを送信するときは複数のスレッドが使用されます。詳細については、「Amazon SES を既存の E メールサーバーと統合します」を参照してください。
- Amazon SES API エンドポイントに近い場所でアプリケーションをホストすることを検討する - Amazon SES API エンドポイントに近いデータセンターか、Amazon SES API エンドポイントと

同じAWS地域の Amazon EC2 インスタンスでアプリケーションをホストすることを検討してください。これにより、アプリケーションと Amazon SES の間のネットワークレイテンシーが小さくなり、スループットが向上する可能性があります。Amazon SES を使用できるリージョンのリストについては、「AWS 全般のリファレンス」の「Amazon Simple Email Service (Amazon SES)」を参照してください。

- 複数のマシンを使用することを検討する - ホストマシンのシステム構成によっては、単一 IP アドレスへの同時 HTTP 接続の数に制限がある場合があります。つまり、単一マシンでの特定の同時接続数を超えた場合に並列処理の利点が制限されることになります。これがボトルネックとなる場合は、複数のマシンを使用して同時 Amazon SES リクエストを実行することを検討してください。
- SMTP エンドポイントではなく Amazon SES クエリ API を使用することを検討する - Amazon SES SMTP エンドポイントとのやり取りには複数のネットワークリクエスト (たとえば、EHLO、MAIL FROM、RCPT TO、DATA、QUIT) から構成される SMTP 対話が含まれるのに対し、 クエリ API を使用すると、単一のネットワーク呼び出しを使用して E メール送信リクエストを送信できます。Amazon SES クエリ API の詳細については、Amazon SES API を使用して E メールを送信する を参照してください。
- Amazon SES メールボックスシミュレーターを使用して最大スループットをテストする - 実装した変更をテストするために、メールボックスシミュレーターを使用できます。メールボックスシミュレーターを使用すると、日次送信クォータを使い果たすことなくシステムの最大スループットを判定できます。メールボックスシミュレータの詳細については、「手動でメールボックスシミュレーターを使用する」を参照してください。

SMTP インターフェイスを介して Amazon SES にアクセスする場合は、「Amazon SES SMTP の問題スループットに影響を与える可能性がある特定の SMTP 関連の問題」を確認してください。

Amazon SES SMTP の問題

このセクションでは、SMTP (Amazon SES Simple Mail Transfer Protocol) インターフェイスを使用した E メールの送信に関連するいくつかの一般的な問題の解決策について説明します。また、Amazon SES から返される SMTP 応答コードも一覧表示します。

Amazon SES SMTP インターフェイス経由の E メール送信の詳細については、「Amazon SES SMTP インターフェイスを使用して E メールを送信」を参照してください。

- Amazon SES SMTP エンドポイントに接続できない。

Amazon SES SMTP エンドポイントへの接続の問題は、ほとんどの場合以下の問題が関係しています。

- 認証情報の誤り - SMTP エンドポイントへの接続に使用する認証情報がAWS 認証情報と異なっています。SMTP 認証情報を取得する方法については、「Amazon SES SMTP 認証情報を取得」を参照してください。認証情報の詳細については、「Amazon SES 認証情報の種類」を参照してください。
- ネットワークまたはファイアウォールの問題 - ネットワークにより、E メールを送信しようとしているポートでのアウトバウンド接続がブロックされている可能性があります。ローカルネットワークが原因で接続の問題が発生しているかどうかを確認するには、コマンドラインに次のコマンドを入力します。*port* は、使用するポート (通常は 465、587、2465 または 2587) に置き換えてください。`telnet email-smtp.us-west-2.amazonaws.com` *port*

  このコマンドを使用して SMTP サーバーに接続でき、さらに TLS ラッパーまたは STARTTLS を使用して Amazon SES に接続しようとしている場合は、コマンドラインを使用して、Amazon SES SMTP インターフェイスへの接続をテストする に示されている手順を実行してください。

  `telnet` または `openssl` を使用して Amazon SES SMTP エンドポイントに接続できない場合は、ネットワークのどこか (ファイアウォールなど) で、使用しようとしているポートでのアウトバウンド接続がブロックされているということです。ネットワーク管理者と協力して、問題を診断および解決してください。

- ポート 25 を使用して Amazon EC2 インスタンスから Amazon SES に送信しているときに、タイムアウトエラーを受信する。

  Amazon EC2 はデフォルトでポート 25 を制限します。これらの制限を削除するためには、E メール送信制限解除申請に Amazon EC2 リクエストを送信します。制限されないポート 465 または 587 を使用して Amazon SES に接続することもできます。

- ネットワークエラーが原因で E メールのドロップが発生している。

  アプリケーションが Amazon SES SMTP エンドポイントに接続するときに再試行ロジックを使用していること、およびアプリケーションがネットワークエラーを検出しメッセージ配信を再試行できることを確認します。SMTP は冗長プロトコルで、このプロトコルを使用して E メールを送信する場合は複数のネットワークラウンドトリップが必要になります。SMTP の性質上、ネットワークエラーの可能性は高くなります。

- SMTP エンドポイントとの接続が失われる。

ほとんどの場合、接続の切断は以下の問題によって発生します。

- MTU サイズ - タイムアウトエラーメッセージが表示された場合、Amazon SES SMTP インターフェイスとの接続に使用しているコンピュータのネットワークインターフェイスの最大送信単位 (MTU) が大きすぎる可能性があります。この問題を解決するには、そのコンピュータの MTU サイズを 1500 バイトに設定します。

  Windows、Linux、macOS の各オペレーティングシステムで MTU サイズを設定する方法の詳細については、Amazon Redshift 管理ガイドの「クライアントでクエリがハングして、クラスターに到達しない場合がある」を参照してください。

  Amazon EC2 の MTU サイズ設定の詳細については、「Amazon EC2 ユーザーガイド」の「EC2 インスタンスのネットワークの最大送信単位 (MTU)」を参照してください。

- 存続時間の長い接続 - Amazon SES SMTP エンドポイントは、Elastic Load Balancer (ELB) の背後の複数の Amazon EC2 インスタンスのフリートで実行されます。システムの最新の状態と耐障害性を維持するために、アクティブな Amazon EC2 インスタンスは定期的に終了され、新しいインスタンスと置き換えられます。アプリケーションは ELB を介して Amazon EC2 インスタンスに接続するため、 Amazon EC2 インスタンスが終了すると接続が無効になります。単一の SMTP 接続を使用して所定の数のメッセージを配信した後、または SMTP 接続がある程度の時間にわたってアクティブであった場合は、新しい SMTP 接続を確立する必要があります。アプリケーションがホストされている場所やアプリケーションがどのように E メールを Amazon SES に送信するかに応じて、いろいろな値を試しながら適切なしきい値を見つける必要があります。

- ネットワークで IP アドレスを許可リストに登録できるように、Amazon SES の SMTP メールサーバーの IP アドレスを知りたい。

  Amazon SES SMTP エンドポイントの IP アドレスは、ロードバランサーの背後に存在します。その結果、これらの IP アドレスは頻繁に変更されます。Amazon SES エンドポイントのすべての IP アドレスの明確なリストを提供することはできません。個々の IP アドレスを許可リストに登録せずに、`amazonses.com` ドメインを許可リストに登録することをお勧めします。

Amazon SES から返される SMTP 応答コード

このセクションでは、Amazon SES SMTP インターフェイスから返される応答コードを一覧表示します。

400 エラーを受け取った SMTP リクエストは、再試行する必要があります。待機時間を少しずつ増やしながらリクエストを再試行するシステムを実装することをお勧めします (たとえば、待機時間を 5 秒、10 秒、30 秒と増やしながら再試行します)。 3 回目の再試行が失敗した場合は、20 分待ってからプロセスを繰り返してください。 段階的な再試行ポリシーを使用する実装例については、AWS メッセージングとターゲティングブログ で「"Throttling – Maximum sending rate exceeded (スロットリング – 最大送信レートの超過)" エラーの対処法」を参照してください。

> **Note**
>
> AWS SDK は、再試行ロジックを自動的に実装しますが、SMTP の代わりに HTTPS インターフェイスを使用します。

500 エラーが発生した場合は、もう一度リクエストを送信する前に、リクエストを修正して問題を修正する必要があります。たとえば、AWS 認証情報が無効な場合は、要求を再度送信する前に正しい認証情報を使用するようにアプリケーションを更新する必要があります。

説明	Response Code (レスポンスコード)	詳細情報
認証に成功	`235 Authentication successful`	SMTP クライアントは SMTP サーバーに正常に接続してサインインしました。
正常に配信	`250 Ok` *`MessageID`*	*`##### ID`* は、Amazon SES がメッセージを一意に識別するために使用する文字列です。
Service unavailable	`421 Too many concurrent SMTP connections`	SMTP サーバーへの接続が現在多すぎるため、Amazon SES はリクエストを処理できません。
ローカル処理エラー	`451 Temporary service failure`	Amazon SES はリクエストを処理できませんでした。リクエストを処理できないという問題がある可能性があります。

説明	Response Code (レスポンスコード)	詳細情報
タイムアウト	`451 Timeout waiting for data from client`	リクエストの間の経過時間が長すぎるため、SMTP サーバーが接続を閉じました。
日次送信クォータの超過	`454 Throttling failure: Daily message quota exceeded`	24 時間の期間に送信することが Amazon SES によって許可されたメッセージの最大数を超えました。詳細については、「Amazon SES 送信制限の管理」を参照してください。
最大送信レートの超過	`454 Throttling failure: Maximum sending rate exceeded`	Amazon SES によって許可された、1 秒あたりに送信できる E メールの最大数を超えました。詳細については、「Amazon SES 送信制限の管理」を参照してください。

説明	Response Code (レスポンスコード)	詳細情報
SMTP 認証情報を検証するときの Amazon SES の問題	`454 Temporary authentication failure`	この問題の原因となる可能性のある問題には、以下が含まれます (ただし、これらに限定されません)。 • E メール送信アプリケーションと Amazon SES の間で暗号化に関する問題が発生しています。Amazon SES に接続するときは暗号化された接続を使用する必要があることに注意してください。詳細については、「Amazon SES SMTP エンドポイントへの接続」を参照してください。 • Amazon SES に問題が発生している可能性があります。更新のために、AWS Service Health Dashboardをチェックします。
リクエストの受信に関する問題	`454 Temporary service failure`	Amazon SES はリクエストを受信できませんでした。 その結果、メッセージは送信されませんでした。
認証情報の誤り	`530 Authentication required`	E メールの送信に使用するアプリケーションは、Amazon SES SMTP インターフェイスに接続したときに認証を試みませんでした。

説明	Response Code (レスポンスコード)	詳細情報
無効な認証情報	`535 Authentication Credentials Invalid`	E メールを送信するのに使用するアプリケーションは Amazon SES に正しい SMTP 認証情報を提供しませんでした。SMTP 認証情報は AWS 認証情報と同じではありません。詳細については、「Amazon SES SMTP 認証情報を取得」を参照してください。
Amazon SES にサブスクライブされないアカウント	`535 Account not subscribed to SES`	SMTP 認証情報を所有する AWS アカウント は、Amazon SES にサインアップされていません。
メッセージが長すぎます	`552 Message is too long.`	送信しようとしているメッセージのサイズが最大メッセージサイズを超えています。
Amazon SES にサブスクライブされないアカウント	`535 Account not subscribed to SES`	SMTP 認証情報を所有する AWS アカウント は、Amazon SES にサインアップされていません。
MAIL FROM 構文エラー	`553 <email-address > Invalid email address`	SMTP メッセージの MAIL FROM 部分に構文エラーがあります。正しい形式に従っていることを確認し、電子メールアドレスを「<>」で囲むことを忘れないでください。

説明	Response Code (レスポンスコード)	詳細情報
RCPT TO 構文エラー	`553 <email-address > address unknown`	SMTP メッセージの RCPT TO 部分に構文エラーがあります。正しい形式に従っていることを確認し、電子メールアドレスを「<>」で囲むことを忘れないでください。
Amazon SES SMTP エンドポイントを呼び出す許可が与えられていないユーザー	`554 Access denied: User UserARN is not authorized to perform ses:SendRawEmail on resource IdentityARN`	SMTP 認証情報を所有するユーザーの AWS Identity and Access Management (IAM) ポリシーまたは Amazon SES 送信認可ポリシーでは、 SMTP エンドポイントを呼び出すことが許可されません。

説明	Response Code (レスポンスコード)	詳細情報
未確認の E メールアドレス	`554 Message rejected: Email address is not verified. The following identities failed the check in region` *`region`*`: `*`identity0`*` , `*`identity1`*` , `*`identity2`*	Amazon SES アカウントからのメール送信が確認されていないメールアドレスまたはドメインからメールを送信しようとしています。 このエラーは、「From」、「Source」、「Sender」、または「Return-Path」のアドレスに該当する場合があります。アカウントがサンドボックスにまだある場合は、Amazon SES メールボックスシミュレーターから提供されているアドレスを除く、すべての受信者のアドレスも確認する必要があります。Amazon SES が検証チェックに失敗したすべての ID を表示できない場合、エラーメッセージは 3 つのピリオド (...) で終わります。 **Note** Amazon SES は複数の AWS リージョンにエンドポイントを持ち、E メールアドレスの検証ステータスはAWS リージョンごとに別個に扱われます。使用する AWS リージョン の各送信者について、検証プロセスを完了する必要があります。

> **Note**
>
> このページのトラブルシューティングに記載のない SMTP の問題については、[お問い合わせ] に一覧表示されている SES サポートオプションをお試しください。

Amazon SES に関するよくある質問 (FAQ)

このセクションでは、Amazon SES の使用に関するよくある質問に対する回答を記載しています。

このセクションには、以下のトピックに関するよくある質問が含まれています。

- Amazon SES 送信レビュープロセスに関するよくある質問
- DNS ブラックホールリスト (DNSBL) に関するよくある質問
- Amazon SES E メール送信メトリクスに関するよくある質問

Amazon SES 送信レビュープロセスに関するよくある質問

Amazon SES から送信された E メールをモニタリングして、このサービスが悪意のあるコンテンツや迷惑なコンテンツ、または質の低い E メールの送信に使用されていないことを確認しています。ユーザーがこのようなカテゴリに分類されるコンテンツを送信していると判断した場合、そのアカウントに対してアクションを起こします。このプロセスを送信レビュープロセスと呼びます。

多くの場合、アカウントの問題を検出すると、そのアカウントをレビューします。アカウントの E メール送信機能を一時停止する場合もあります。これらの対処を行うのは、各アカウントの送信者の評価を保護するためであり、他の SES ユーザーが、サービスの中断や配信性能に関する問題に直面することを回避するためです。

内容

- アカウントのレビューに関するよくある質問
- 送信一時停止に関するよくある質問
- バウンスに関するよくある質問
- 苦情に関するよくある質問
- スパムトラップに関するよくある質問
- 手動調査に関するよくある質問

アカウントのレビューに関するよくある質問

Q1.「お客様のアカウントはレビュー対象です」というメッセージを受け取りました。これはどういう意味ですか?

アカウントから送信された E メールに関係した問題が見つかりました。問題を修正するまでには猶予期間があります。引き続き通常どおり E メールを送信できますが、お客様のアカウントがレビュー対象の原因となった問題を修正する必要があります。レビュー期間が終了するまでに問題を修正しない場合は、以後の E メールの送信機能が一時停止される場合があります。

Q2. アカウントがレビュー対象になるときは、必ず通知されますか?

はい。AWS アカウントに関連付けられた E メールアドレスに通知が送信されます。

Q3. アカウントがレビュー対象であるという通知を受け取らなかったのはなぜですか?

アカウントがレビュー対象になると、AWS アカウントに関連付けられた E メールアドレスに通知が自動的に送信されます。この E メールアドレスは、AWS アカウントの作成時に指定したものです。このメールアドレスは、SES を使用して E メールを送信する際に使用するアドレスとは異なる場合があります。

評価メトリクスを定期的に参照して、送信者の評価をモニタリングすることをお勧めします。また、Amazon CloudWatch で自動アラームを設定できます。これらのアラームは、評価メトリクスが特定のしきい値を超えたときに通知を送信します。Amazon CloudWatch は、携帯電話にテキストメッセージを送信するなど、他の方法で連絡するように設定することもできます。

Q4. SES アカウントがレビュー対象になると、その他の AWS サービスの使用に影響が及びますか。

SES のアカウントがレビュー中であっても、その他の AWS サービスは利用できます。ただし、アウトバウンド通信を送信するようなその他の AWS サービス (Amazon SNS など) のサービスクォータの引き上げをリクエストする場合、SES アカウントのレビューが解除されるまでは、リクエストが却下される場合があります。

Q5. アカウントがレビュー対象である場合、どうすればよいですか?

次のことを実行してください。

- お客様の状況で可能な場合は、問題が解決されるまでメールの送信を停止します。アカウントのレビュー中も E メールの送信を続行できます。ただし、変更を加えずにメールを送信し続けると、不用意に問題を悪化させる可能性があります。
- 当社から受け取る E メールをご覧になり、問題の概要について確認してください。
- 送信内容を調べて、どのような送信が原因で問題が発生したのかを具体的に特定します。
- 問題解決のために変更した後に、AWSコンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージでは、問題を解決するために行ったステップの詳細情報を提供し、これらのステップでどのように今後問題の再発を防ぐことができるかを説明します。
- 当社が具体的に要求するすべての情報を必ず提供してください。お客様の状況を評価するために、これらの情報が必要になります。

Q6. 見直しのリクエストをするにはどうすればよいですか?

レビュー対象にするという決定の見直しをリクエストできます。見直しをリクエストするために、AWSコンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。

リクエストで以下の情報を提供してください。

- アカウントがレビュー対象となった根本原因に関する情報。
- 問題修正のために行った変更のリスト。実行済みのステップのみを含め、今後実行する予定のステップは含めないでください。
- これらの変更により今後どのように同じ問題の再発が防止されるかに関する情報。

アカウントがレビュー対象となったイベントの性質によっては、追加情報を提出していただく場合があります。リクエストに含める情報のリストについては、発生した問題に関する、よくある質問のトピックを参照してください。

Q7. 見直しリクエストが受け入れられない場合どうなりますか?

当社はお客様にリクエストを受け入れなかった理由に関する情報を回答します。問題を解決し、その変更により今後問題の再発を防げることを示すことができれば、場合によっては、別のリクエストを送信することができます。

Q8. 問題を診断する際にサポートを受けることはできますか?

通常は、問題の概要だけが伝えられます (たとえば、バウンスに関する問題があるなど)。根本原因は、お客様の側で調査する必要があります。

Q9. アカウントがレビュー対象でなくなったことを確認するには、どうすればよいですか?

評価メトリクスにはアカウントの現在のステータスに関する情報が含まれます。詳細については、「評価メトリクスを使用して返送率と苦情率を追跡する」を参照してください。

Q10. 問題が存在する場合は必ずアカウントがレビュー対象になりますか。

いいえ。状況によっては、アカウントをまずレビュー対象にするのではなく、E メール送信機能を一時停止する場合があります。以下に例を示します。

- 問題が非常に深刻な場合。
- お客様のアカウントが過去に何度も同じ問題でレビュー対象になっている場合。そのため、レビュー対象の原因となった個別の問題を解決するだけではなく、根本的な問題に対処することが重要となります。たとえば、特定のキャンペーンが原因でアカウントがレビュー対象になっているのであれば、そのキャンペーンを単に停止する以上のことを実行する必要があります。キャンペーンのどのような特徴が問題となっていたかを特定し、今後のキャンペーンで同じ問題が発生しないようにするプロセスを構築する必要があります。

いずれの場合でも、アカウントの E メール送信機能を一時停止する際には、自動的に通知を送信します。

Q11. レビュー対象の期限が切れる直前に問題を解決した場合どうなりますか?

AWS Management Consoleにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。ケースへの返信で、問題が解決したことを通知してください。

Q12. AWS の担当者やプレミアムサポートからサポートを受けることはできますか?

AWS のアカウント担当者がすでに決まっている場合は、お客様のアカウントがレビュー対象になると、その担当者に自動的に通知されます。アカウント担当者は、問題の把握に役立つ追加情報をお客様に提供できる場合があります。プレミアムサポートを利用している場合は、そのチームに連絡してさらなるサポートを受ける必要もあります。

送信一時停止に関するよくある質問

Q1. 「アカウントの E メール送信機能を一時停止します」というメッセージを受信しました。これはどういう意味ですか?

お客様が送信した E メールに重大な問題があるため、お客様のアカウントの E メール送信機能を一時停止しました。ほとんどの場合、アカウントの一時停止は、次のいずれかの理由によります。

- 以前にアカウントがレビュー対象になっていた。アカウントがレビュー対象になった原因がレビュー期間の終了までに修正されなかったため、アカウントの E メール送信を一時停止しました。
- アカウントは同じ問題で何度もレビュー対象になっている。
- アカウントは AWS サービス利用規約に違反した E メールを送信した。これらの違反が深刻な場合、アカウントをまずレビュー対象にするのではなく、E メール送信機能を一時停止する可能性があります。

Q2. アカウントの E メール送信機能が一時停止されるときは、必ず通知されますか。

はい。AWS アカウントに関連付けられた E メールアドレスに通知が送信されます。

Q3. アカウントの E メール送信機能が一時停止されています。なぜ通知がなかったのでしょうか?

アカウントの E メール送信機能を一時停止する場合は、そのアカウントに関連付けられている E メールアドレスに通知を送信します。

> **Note**
>
> AWS アカウントを作成するときに、E メールアドレスを指定する必要があります。このアドレスはいつでも変更できます。AWS アカウントに関連付けられているアドレスを変更する詳しい方法については、「AWS アカウントの管理」 (AWS Billing and Cost Management ユーザーガイド) を参照してください。

バウンス率や苦情率が高すぎるときに通知するアラームを Amazon CloudWatch で作成できます。アラームを作成すると、アカウントの E メール送信機能の一時停止に至るような要因について早い段階で警告を受けることができます。ただし、バウンスや苦情以外にも E メール送信機能を一時停止

する要因があります。CloudWatch アラームの作成の詳細については、「CloudWatch を使用して評価モニタリングアラームを作成する」を参照してください。

アカウントダッシュボードを使用してアカウントの現在のステータスを確認することもできます。例えば、アカウントの E メール送信機能が現在一時停止になっている場合、アカウントダッシュボードの [Account status] (アカウントステータス) セクションに、ステータスとして [Paused] (一時停止) と表示されます。アカウントで正常に E メールを送信できる場合は、[Healthy] (正常) と表示されます。

最後に、https://phd.aws.amazon.com/ で AWS Health Dashboard (PHD) をチェックし、アカウントの E メール送信機能が現在一時停止されているかどうかを確認できます。アカウントの E メール送信機能が一時停止されると、PHD の [Event log (イベントログ)] セクションに [SES sending paused (SES 送信一時停止)] イベントが自動的に追加されます。[SES sending paused (SES 送信一時停止)] イベントのステータスは、アカウントの E メール送信機能が現在一時停止されているかどうかに関係なく、常に [Closed (クローズ)] です。イベントログには、送信一時停止イベントの発生時に AWS アカウントに関連付けられていた E メールアドレスに送信された E メールのコピーも含まれます。

個人用 Health ダッシュボードに新しいイベントが表示されたときにアラートするアラームを CloudWatch で作成できます。詳細については、AWS Health ユーザーガイドの「 CloudWatch Eventsを使用した AWS Health イベントのモニタリング」を参照してください。

Q4. アカウントの E メール送信機能が一時停止されています。これによって、他の AWS サービスの使用に影響がありますか?

アカウントの E メール送信機能の一時停止中にも他の AWS サービスを使用することもできます。ただし、アウトバウンド通信を送信する他のAWSのサービス (Amazon SNS など) で service quota の引き上げを申請した場合、アカウントの E メール送信機能が回復するまで、その申請は拒否されることがあります。

Q5. アカウントの E メール送信機能が一時停止されるときは、どうすればよいですか?

次のことを実行してください。

- 当社から受け取る E メールをご覧になり、問題の概要について確認してください。
- 送信内容を調べて、どのような送信が原因で問題が発生したのかを具体的に特定します。
- 問題解決のために変更した後に、AWSコンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージでは、問題を解決するために行っ

たステップの詳細情報を提供し、これらのステップでどのように今後問題の再発を防ぐことができるかを説明します。

- 当社が具体的に要求するすべての情報を必ず提供してください。お客様の状況を評価するために、これらの情報が必要になります。

Q6. レビューとは何ですか?

レビュー対象にするという当社の決定を見直すようにリクエストできます。見直しのリクエストに関する詳細については、次の質問を参照してください。

Q7. 見直しのリクエストをするにはどうすればよいですか?

見直しをリクエストするために、AWSコンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。

リクエストで以下の情報を提供してください。

- この問題の原因に関する情報。
- 問題修正のために行った変更のリスト。実行済みのステップのみを含め、今後実行する予定のステップは含めないでください。
- これらの変更により今後どのように同じ問題の再発を防ぐことができるかに関する情報。

アカウントの E メール送信機能が一時停止となったイベントの性質によっては、追加情報を提出していただく場合があります。リクエストに含める情報のリストについては、発生した問題に関する、よくある質問のトピックを参照してください。

Q8. リクエストが受け入れられない場合どうなりますか?

当社はお客様にリクエストを受け入れなかった理由に関する情報を回答します。問題を解決し、その変更により今後問題の再発を防げることを示すことができれば、場合によっては、別のリクエストを送信することができます。

Q9. 問題を診断する際にサポートを受けることはできますか?

通常は、問題の概要だけが伝えられます (たとえば、バウンスに関する問題があるなど)。問題を解決するのは、お客様の責任です。

Q10. アカウントの E メール送信機能が回復したことはどうすればわかりますか?

評価メトリクスにはアカウントの現在のステータスに関する情報が含まれます。詳細については、「評価メトリクスを使用して返送率と苦情率を追跡する」を参照してください。

Q11. AWS の担当者やプレミアムサポートからサポートを受けることはできますか?

AWS のアカウント担当者がすでに決まっている場合は、お客様のアカウントの E メール送信機能が一時停止になると、その担当者に自動的に通知されます。アカウント担当者は、問題の把握に役立つ追加情報をお客様に提供できる場合があります。プレミアムサポートを利用している場合は、そのチームに連絡してさらなるサポートを受ける必要もあります。

バウンスに関するよくある質問

Q1. なぜバウンスを重視しているのですか?

高いバウンス率は、E メールプロバイダーやアンチ スパム組織などのエンティティで、不良な E メール送信プラクティスに関与している送信者を検出するためによく使用されます。高いバウンス率は、E メールが受信トレイではなく迷惑メールフォルダに送信される原因になる可能性があります。

Q2. アカウントのバウンス率により、アカウントがレビュー対象になっている、または送信が一時停止になっている、という通知を受信した場合、どうすればよいですか?

問題の原因を特定し、修正します。問題解決のために変更した後に、AWSコンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージでは、問題を解決するために行ったステップの詳細情報を提供し、これらのステップでどのように今後問題の再発を防ぐことができるかを説明します。次の情報も含めてください。

- バウンスの追跡で使用する方法。
- 新しい受信者へ送信する前に、新しい受信者の E メールアドレスが有効であることを検証する方法。たとえば、「Q11. バウンスを最小限に抑えるにはどうすればよいですか?」で実行する推奨事項などがあります。

Q3. どのような種類のバウンスがバウンス率に含まれますか?

バウンス率には、まだ確認していないドメインに対するハードバウンスのみが含まれます。ハードバウンスは、「アドレスは存在しません」などの永続的な配信障害です。「メールボックスがいっぱいです」 などの一時的かつ断続的な障害や、IP アドレスのブロックによるバウンスは、バウンス率にカウントされません。

Q4. アカウントのレビューや、送信の一時停止の原因になりえるバウンス率は公開されていますか？

最良の結果を得るには、バウンス率を 2% 未満に維持する必要があります。これより高いバウンス率は、E メールの配信に影響する可能性があります。

バウンス率が 5% 以上になると、アカウントはレビュー対象になります。バウンス率が 10% 以上の場合は、高いバウンス率の原因となった問題が解決するまで、以後の E メール送信を一時停止することがあります。

Q5. バウンス率の計算対象となる期間はどれくらいですか?

バウンス率の計算は固定期間に基づいて行われるわけではありません。これは、さまざまな送信者が異なる割合で送信を行うためです。代わりに、代表ボリューム (典型的な通常の E メール送信量) を調べます。大量のメールの送信者と少量のメールの送信者を公平に扱うために、代表的ボリュームはユーザーごとに異なっており、ユーザーの送信パターンの変化に伴い代表的ボリュームも変わります。

Q6. SES コンソールや GetSendStatistics API の情報を使用して、独自にバウンス率を計算することはできますか。

いいえ。バウンス率は、代表ボリュームを使用して計算されます (「Q5. バウンス率の計算対象となる期間はどれくらいですか?」を参照)。送信率によっては、SES コンソールや `GetSendStatistics` で取得できる期間より以前の期間を含めたバウンス率である場合があります。また、バウンス率を計算するときに考慮されるのは、検証されていないドメインへの E メールのみです。ただし、これらのメソッドを使用して定期的にバウンス率をモニタリングしている場合は、問題が発生してアカウントがレビュー対象になる、または、アカウントの E メール送信機能が一時停止される前に問題を見つける適切な指標になります。

Q7. どの E メールアドレスがバウンスされたかを確認するには、どうすればよいですか?

SES から送信されたバウンス通知を調べます。SES が通知を転送する先の E メールアドレスは、「E メールを介したAmazon SES に関する通知の受信」に説明されているとおり、元のメッセージの送信方法によって異なります。また、「Amazon SES のイベント通知の設定」で説明されているように、Amazon Simple Notification Service (Amazon SNS) からバウンス通知をセットアップすることもできます。バウンスされたアドレスを詳しく調べずにリストから削除すると、根本的な問題が解決できなくなる場合があるので、注意してください。バウンスを減らすために実行できる作業については、「Q11. バウンスを最小限に抑えるにはどうすればよいですか?」を参照してください。

Q8. バウンスをモニタリングしていない場合、バウンスされたアドレスのリストは提供されますか?

いいえ。バウンスされたアドレスの完全なリストを提供することはできません。アカウントのバウンスをモニタリングし、それに応じて行動するのは、お客様の責任です。

Q9. バウンスはどのように処理すればよいですか?

バウンスされたアドレスをメーリングリストから削除し、それらのアドレスへのメール送信をすぐに停止する必要があります。送信メールが少量であれば、場合によっては、E メールを使用してバウンスをモニタリングし、バウンスされたアドレスをメーリングリストから手動で削除するだけで十分です。もしボリュームが多ければ、場合によっては、このプロセスのオートメーションをセットアップする必要があります。そのためには、バウンスを受信したメールボックスをプログラムで処理するか、Amazon SNS を経由するようにバウンス通知を設定します。詳細については、「Amazon SES のイベント通知の設定」を参照してください。

Q10. 送信クォータに達したために、E メールがバウンスされる場合はありますか?

いいえ。バウンスは送信クォータとは関係ありません。送信クォータを超えた E メールを送信しようとすると、SES API または SMTP インターフェイスからエラーが返されます。

Q11. バウンスを最小限に抑えるにはどうすればよいですか?

まず、バウンスの状況について把握してください (「Q7. どの E メールアドレスがバウンスされたかを確認するには、どうすればよいですか?」を参照)。そして、次のガイドラインに従ってください。

- E メールアドレスの購入、貸与、共有は行わないでください。E メールの受信を明示的にリクエストした受信者にのみ E メールを送信します。
- バウンスされた E メールアドレスをリストから削除します。
- ユーザーは、ウェブのフォームで、E メールアドレスを 2 回入力してフォーム送信前に両方のアドレスが一致していることをチェックするように求められます。
- ダブルオプトインを使用して、新しいユーザーをサインアップします。つまり、新しいユーザーは、サインアップ後に確認メールを受信します。その確認メールで確認のクリックをすると、その他のメールを受け取ることができます。これにより、他のユーザーのサインアップをしたり、誤ってサインアップしたりすることを回避できます。
- 最近メールを送信していないアドレスに送信する必要がある場合 (つまり、そのアドレスがまだ有効であるかどうかを確認できていない場合)、すべての送信内容のごく一部だけを、そのアドレス

に送信します。詳細については、ブログ投稿「古いアドレスには送信しないこと。ただし、送信する必要がある場合はどうするか?」を参照してください。

- 架空のアドレスの使用を推奨するようなサインアップを構築していないことを確認します。たとえば、付加価値や特典は受信者の確認済みアドレスにのみ与えるようにしてください。
- 「友だちにメールを送る」機能を取り入れている場合は、CAPTCHA や同様のメカニズムを使用して、この機能が自動的に使用されることを防いでください。また、ユーザーが自由にコンテンツを挿入することを許可しないでください。
- システム通知のために SES を使用している場合は、メールを受信できる実在のアドレスに通知を送信していることを確認してください。また、不要な通知は無効にすることを検討してください。
- 新しいシステムをテストしている場合は、E メールを受信できる実在のアドレスに送信していること、または SES メールボックスシミュレーターを使用していることを確認してください。詳細については、「手動でメールボックスシミュレーターを使用する」を参照してください。

苦情に関するよくある質問

Q1. 苦情とは何ですか?

受信者が E メールの受信をリクエストしていないことを報告した場合に、苦情が発生します。受信者が E メールクライアントで [スパムの報告] などのボタンをクリックした場合、E メールプロバイダーに苦情を報告した場合、SES に直接またはその他の方法で通知した場合などが該当します。このトピックには、苦情に関する全般的な情報が含まれています。通知に、苦情のソースに関する特定の情報が含まれている場合は、以下の関連トピックも参照してください。

- フィードバックループを介した SES の苦情に関するよくある質問
- 受信者から直接 SES に寄せられた苦情に関するよくある質問
- E メールプロバイダーを介した SES の苦情に関するよくある質問

Q2. なぜ苦情を重視しているのですか?

高い苦情率は、E メールプロバイダー、アンチスパム組織などの団体によって頻繁に使用されます。このような苦情率は、E メールの受信にサインアップしていない受信者に送信者がメールを送ったことや、送信者が送ったコンテンツは受信者がサインアップしたコンテンツのタイプと異なっていることを示す指標として利用されます。

Q3. 苦情が絡む問題により、アカウントがレビュー対象になるという通知や、送信が一時停止になるという通知を受信した場合、どうすればよいですか?

お客様のリスト取得プロセスと E メールの内容を確認して、受取人がお客様からの E メールを受け取りたくない理由を確認します。問題の原因を特定し、修正します。問題解決のために変更した後に、AWSコンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージでは、問題を解決するために行ったステップの詳細情報を提供し、これらのステップでどのように今後問題の再発を防ぐことができるかを説明します。

Q4. 苦情を最小限に抑えるにはどうすればよいですか?

まず、必ず SES から通知される苦情をモニタリングします。SES は、フィードバックループ (「フィードバックループを介した SES の苦情に関するよくある質問」を参照) を介してこのような苦情を受け取ります。そして、次のガイドラインに従ってください。

- E メールアドレスの購入、貸与、共有は行わないでください。お客様からのメールを具体的にリクエストしたアドレスのみを使用します。
- ダブルオプトインを使用して、新しいユーザーをサインアップします。つまり、ユーザーは、サインアップ後に確認メールを受信します。その確認メールで確認のクリックをすると、その他のメールを受け取ることができます。これにより、他のユーザーのサインアップをしたり、誤ってサインアップしたりすることを回避できます。
- 送信したメールへの対応状況をモニタリングし、メッセージを開かなかった受信者やクリックしなかった受信者への送信を停止します。
- 新しいユーザーがサインアップするとき、ユーザーが受信する E メールの種類を明確にします。また、ユーザーがサインアップしたときに指定した種類のメールだけを送信するようにします。たとえば、ユーザーがニュースの更新にサインアップした場合は、広告を送信しないでください。
- メールが適切な形式になっており、見た目の質も高いことを確認します。
- お客様からのメールであることが明確で、他のメールと混同されないことを確認します。
- ユーザーがメールの受信登録を解除するとき、わかりやすく簡単な方法で実行できるようにします。

フィードバックループを介した SES の苦情に関するよくある質問

このトピックでは、フィードバックループを介して SES が E メールプロバイダーから受け取る苦情に関する情報を提供します。すべての種類の苦情に適用される一般的情報については、「苦情に関するよくある質問」を参照してください。

Q1. この種類の苦情はどのように報告されていますか?

ほどんどの E メールクライアントプログラムには、メッセージをスパムフォルダに移動して E メールプロバイダーに転送するためのボタン ([Mark as Spam (スパムとしてマーク)] など) が用意されています。また、ほとんどのプロバイダーでは、ユーザーが不要な E メールを転送して E メールプロバイダーによる防止策をリクエストできる迷惑メール用アドレス (abuse@example.com など) を用意しています。E メールプロバイダーとのフィードバックループ (FBL) を SES で設定している場合、苦情は SES に転送されます。

> **Note**
>
> メッセージを送信すると、SES は自動的にフィードバック ID ヘッダーを設定し、メールボックスプロバイダーが苦情やスパム率などの配信統計を集約できるようにして、ユーザーも統計を利用できるようにします。SES が提供するフィードバック ID ヘッダー値の構成は、以下のとおりです。
>
> - `FeedBackId:((SESInternalID):(AmazonSES))`。ここで、
>   - SESInternalID は、SES が苦情情報を収集するために使用する識別子です。
>   - AmazonSES は、SES を送信プラットフォームとして識別する静的タグです。
>
> 必要に応じて、SES が提供する標準のフィードバック ID ヘッダー値に加えて、`ses:feedback-id-a` メッセージタグと `ses:feedback-id-b` メッセージタグ (最大 2 つまで) を使用して独自のカスタマイズしたフィードバック ID を指定することもできます。「the section called “E メールキャンペーンのきめ細かなフィードバック”」を参照してください。

Q2. このような苦情は、SES コンソールにある苦情率の統計に含まれていて、GetSendStatistics API で返されますか。

はい。ただし、苦情率の統計には SES にフィードバックを提供しない E メールプロバイダーからの苦情は含まれません。フィードバックを提供するドメインでの苦情率は、他の送信を代表しているものと考えることができます。

Q3. これらの苦情はどのようにして通知されますか?

E メールまたは Amazon SNS 通知を通じて通知を受けることができます。セットアップ手順については、「Amazon SES のイベント通知の設定」を参照してください。

Q4. E メールまたは Amazon SNS を通じて苦情の通知を受け取った場合はどうすればよいですか?

最初に、苦情が発生したアドレスをメーリングリストから削除し、それらのアドレスへのメール送信をすぐに停止する必要があります。受信解除のリクエストを受け取ったことを示す E メールの送信も行わないでください。このプロセスのオートメーションのセットアップを検討してください。そのためには、苦情を受信したメールボックスをプログラムで処理するか、Amazon SNS を経由して苦情通知を設定するかします。詳細については、「Amazon SES のイベント通知の設定」を参照してください。

次に、送信状況を詳細に調べて、お客様から送信されたメールを受信者が快く思わない理由を判断し、根本的な問題に対処します。苦情を持つすべてのユーザーの中には、お客様のメールを快く思わなかったけれども、苦情を申し出なかった (または苦情を申し出ることができなかった) 受信者が多数いる可能性があります。実際に苦情を申し出た受信者を削除しただけでは、根本的な問題に対処したことにはなりません。

Q5. アカウントがレビュー対象になったり、アカウントの E メール送信機能が一時停止されたりする原因となり得る SES 苦情率は公開されていますか。

最良の結果を得るには、苦情率を 0.1% 未満に維持する必要があります。これより高い苦情率は、E メールの配信に影響する可能性があります。

苦情率が 0.1% 以上になると、アカウントはレビュー対象になります。苦情率が 0.5% 以上の場合は、高い苦情率の原因となった問題が解決するまで、以後の E メール送信を一時停止することがあります。

Q6. 苦情率の計算対象となる期間はどれくらいですか?

苦情率の計算は固定期間に基づいて行われるわけではありません。これは、さまざまな送信者が異なる割合で送信を行うためです。代わりに、代表ボリューム (典型的な通常のメール送信量) を調べます。大量のメールの送信者と少量のメールの送信者を公平に扱うために、代表的ボリュームはユーザーごとに異なっており、ユーザーの送信パターンの変化に伴い代表的ボリュームも変わります。また、苦情率はすべての E メールに基づいて計算されるわけではありません。代わりに、苦情率は、苦情のフィードバックを SES に送信するドメインを対象としており、これらのドメインに送信されたメールに対する苦情の割合として計算されます。

Q7. SES コンソールや GetSendStatistics API からのメトリクスを使用して、独自に苦情率を計算することはできますか。

いいえ。その主な理由には、次の 2 つがあります。

- 苦情率は、代表ボリュームを使用して計算されます (「Q6. 苦情率の計算対象となる期間はどれくらいですか?」を参照) 。送信率によっては、SES コンソールや GetSendStatistics API で取得できる期間より以前の期間を含めた苦情率である場合があります。このため、これらのメソッドを定期的に使用して、アカウントの苦情率をモニタリングすることをお勧めします。このように苦情率をモニタリングすることで、E メールの配信に影響を及ぼす前に、問題を特定するための必要な情報を取得できます。
- 苦情率を計算するとき、すべての E メールが計算対象になるわけではありません。苦情率は、苦情のフィードバックを SES に送信するドメインを対象としており、これらのドメインに送信されたメールに対する苦情の割合として計算されます。

Q8. どの E メールアドレスに苦情があったかを確認するには、どうすればよいですか?

E メールまたは Amazon SNS を介して SES から受信した苦情の通知を確認します (「Amazon SES のイベント通知の設定」を参照)。ただし、E メールプロバイダーから提供される情報の量は異なり、プロバイダーによっては苦情を送信した受取人の E メールアドレスを編集してから苦情通知を SES に渡す場合もあります。今後、受取人の E メールアドレスを特定できるようにする最善の方法は、SES が E メールを受け付ける際にお客様に渡す SES メッセージ ID と識別子のマッピングを保存しておくことです。カスタムメッセージ ID を追加しても、SES には保持されないことに注意してください。

Q9. 苦情をモニタリングしていない場合、苦情があったアドレスのリストは提供されますか?

残念ながら、弊社では、包括的なリストを提供することはできません。ただし、E メールや Amazon SNS を利用して、将来の苦情をモニタリングすることはできます。

Q10. E メールのサンプルを入手できますか?

リクエストに応じて E メールのサンプルをお送りすることはできませんが、この情報は苦情通知に含まれている場合があります。詳細については、「Q8. どの E メールアドレスに苦情があったかを確認するには、どうすればよいですか?」を参照してください。

受信者から直接 SES に寄せられた苦情に関するよくある質問

このトピックでは、受信者から直接 SES に寄せられた苦情に関する情報を提供します。すべての種類の苦情に適用される一般的情報については、「苦情に関するよくある質問」を参照してください。

Q1. この種類の苦情はどのように報告されていますか?

E メールやその他の方法を介して、お客様が送信した E メールに関して複数の受信者から直接 SES に連絡がありました。

Q2. このような苦情は、SES コンソールにある苦情率の統計に含まれていて、GetSendStatistics API で返されますか。

いいえ。SES コンソールや `GetSendStatistics` API を使用して取得する苦情率の統計に含まれるのは、フィードバックループを通じて SES が受け取った苦情のみです。これらの苦情のタイプの詳細については、「フィードバックループを介した SES の苦情に関するよくある質問」を参照してください。

Q3. E メールのフィードバック通知または Amazon SNS を通じて、これらの苦情について通知されないのはなぜですか?

E メールのフィードバック転送および Amazon SNS 通知には、フィードバックループを介して SES が受け取る苦情のみが含まれます。受信者から直接 SES に寄せられた苦情についての通知は送信されません。

Q4. どの E メールアドレスに苦情があったかを確認するには、どうすればよいですか?

苦情元の受信者の ID を保護するために、お客様の E メールについて苦情を申し出た受信者の E メールアドレスのリストを提供することはできません。

お客様のリストからそのような受信者を削除することに注力するのではなく、苦情が寄せられる原因となった問題を特定することをお勧めします。まず、顧客獲得プロセスを見直し、お客様からの E メールの受信を明示的にリクエストしていない顧客をリストから削除することをお勧めします。E メールの内容を分析して、受信者が苦情を申し出ている理由の把握に努めてください。

Q5. E メールのサンプルを入手できますか?

苦情元の受信者の ID を保護するために、受信者が苦情を申し出る原因となった E メールのコピーを提供することはできません。

Q6. 直接的な苦情により、アカウントがレビュー対象である、または送信が一時停止になるという通知を受信した場合、どうすればよいですか?

送信プロセスをすぐに変更して、お客様からのメッセージの受信にサインアップした受信者にのみメッセージを送信するようにします。また、受信者がサインアップした受信コンテンツタイプを送信するようにしてください。問題解決のために変更した後に、AWSコンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージでは、問題を解決するために行ったステップの詳細情報を提供し、これらのステップでどのように今後問題の再発を防ぐことができるかを説明します。

お客様が3 週間以内に見直しをリクエストせず、かつ受信者から直接苦情が送信され続ける場合、お客様のアカウントの E メール送信機能を一時停止することがあります。

E メールプロバイダーを介した SES の苦情に関するよくある質問

このトピックでは、SES が E メールプロバイダー (別称メールボックスプロバイダー) を介して受け取る苦情に関する情報を提供します。すべての種類の苦情に適用される一般的情報については、「苦情に関するよくある質問」を参照してください。

Q1. この種類の苦情はどのように報告されていますか?

非常に多くの数の顧客が、お客様の E メールをスパムとマークしたことが E メールプロバイダーから SES に報告されました。この報告は、「フィードバックループを介した SES の苦情に関するよくある質問」で説明されるフィードバックループ以外の方法で SES に提供されました。

Q2. このような苦情は、SES コンソールにある苦情率の統計に含まれていて、GetSendStatistics API で返されますか。

いいえ。SES コンソールや `GetSendStatistics` API を使用して取得する苦情率の統計に含まれるのは、フィードバックループを通じて SES が受け取った苦情のみです。

Q3. E メールのフィードバック通知または Amazon SNS を通じて、これらの苦情について通知されないのはなぜですか?

E メールのフィードバック転送および Amazon SNS 通知には、フィードバックループを介して SES が受け取る苦情のみが含まれます。

Q4. どの E メールアドレスに苦情があったかを確認するには、どうすればよいですか?

通常、E メールプロバイダーはこの情報を公開していません。ただし、リストから個別の受信者を削除することではなく、根本的な問題の発見と修正に重点を置く必要があります。お客様のリスト取得プロセスと E メールの内容を確認して、受取人がお客様の E メールを受け取りたくない理由を特定することから開始します。

Q5. E メールのサンプルを入手できますか?

いいえ。通常、E メールプロバイダーはサンプルの E メールを公開していません。

Q6. E メールプロバイダーの苦情により、アカウントがレビュー対象である、または送信が一時停止になるという通知を受信した場合、どうすればよいですか?

問題の原因を特定し、修正します。問題解決のために変更した後に、AWSコンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージでは、問題を解決するために行ったステップの詳細情報を提供し、これらのステップでどのように今後問題の再発を防ぐことができるかを説明します。お客様が3 週間以内に見直しをリクエストせず、か

つ引き続き E メールプロバイダーから苦情が送信され続ける場合、お客様のアカウントの以後の E メール送信を一時停止することがあります。

スパムトラップに関するよくある質問

Q1. スパムトラップとは何ですか?

スパムトラップは、インターネットサービスプロバイダー (ISP) 、E メールプロバイダー、またはアンチスパム組織によって管理される特別な E メールアドレスです。アドレスが E メール受信のサインアップを正当に行っていないため、スパムトラップを管理する組織は、アドレスにメールを送信している者が 不審なメール行為を実施しているようであると判断しています。

Q2. スパムトラップはどのようにセットアップされますか?

スパムトラップのアドレスは複数の方法でセットアップされます。これらのアドレスは、以前は有効であったアドレスから変換されたものですが、長期間にわたって未使用 (およびバウンスの状態) になっています。また、これらのアドレスはスパムトラップとして使用されることのみを目的としてセットアップされています。スパムトラップのアドレスは、推測しにくい通常とは異なるアドレスで、実際のアドレスに近いアドレスになっている場合もあります (たとえば、一般的なドメイン名のスペルを変えたアドレスなど)。常にというわけではありませんが、スパムトラップがさまざまな方法でインターネット上に配置され、世界中に「仕掛けられる」ことがよくあります。

Q3. スパムトラップに送信した場合、SES はどのように把握しますか。

SES の送信者がスパムトラップにヒットすると、スパムトラップを運営する特定の組織が SES に通知を送信します。

Q4. SES はどのようにスパムトラップレポートを使用するのですか。

レポートを確認します。アカウントがスパムトラップに E メールを送信していると判断された場合、アカウントはレビュー対象になり、根本的な問題を解決するよう求められます。レビュー期間が終了するまでに問題を修正しない場合、アカウントの E メール送信機能が一時停止されることがあります。スパムトラップの問題が非常に深刻である場合は、アカウントをまずレビュー対象にするのではなく、ただちに E メール送信機能を一時停止する可能性があります。

Q5. スパムトラップの問題により、アカウントがレビュー対象である、または送信が一時停止になるという通知を受信した場合、どうすればよいですか?

まず、アカウントがレビュー対象になる、または、アカウントの E メール送信機能が一時停止される原因となった問題に対処します。次に、AWSコンソールにサインインし、サポートセンターに移

動します。代理でオープンしたケースに返信してください。メッセージでは、問題を解決するために行ったステップの詳細情報を提供し、これらのステップでどのように今後問題の再発を防ぐことができるかを説明します。お客様が加えた変更により問題が適切に対処されたと見なされた場合は、アカウントのレビュー期間をキャンセル、または一時停止を解除します。

スパムトラップのヒットが報告される方法に起因して、お客様の加えた変更で問題が解決されたかどうかを判断するのに 3 週間以上かかる場合があります。

Q6. アカウントがレビュー対象になる、または、アカウントの E メール送信機能が一時停止されるまで許容されるスパムトラップのヒットは何件ですか?

アカウントに対してアクションを実行するまでのスパムトラップの特定のヒット件数は公開されていません。ただし、重要な注意点として、スパムトラップのヒットは少数でも、お客様の送信者としての評価に悪影響を及ぼす可能性があります。そのため、スパムトラップレポートを重視する必要があります。

Q7. スパムトラップのアドレスは公開されていますか?

いいえ。スパムトラップが有効であるためには、スパムトラップが公開されていないことが重要です。スパムトラップの組織は、スパムトラップのヒットが発生したことだけを公開しており、実際のスパムトラップのアドレスは公開していません。

Q8. スパムトラップへの送信を回避するにはどうすればよいですか?

スパムトラップに送信するリスクを軽減するには、次のガイドラインに従ってください。

- E メールアドレスの購入、貸与、共有は行わないでください。お客様からのメールを具体的にリクエストしたアドレスのみを使用します。
- ユーザーは、ウェブフォームで、E メールアドレスを 2 回入力してフォーム送信前に両方のアドレスが一致していることを確認するように求められます。
- ダブルオプトインを使用して、新しいユーザーをサインアップします。つまり、ユーザーは、サインアップ後に確認メールを受信します。その確認メールで確認のクリックをすると、その他のメールを受け取ることができます。
- ハードバウンスが発生するアドレスをリストから削除します。これにより、それらのアドレスは、スパムトラップに変換されるよりもかなり前に削除されることになります。
- 受信者の関与の状況をモニタリングし、最近 E メールやウェブサイトを使用していない受信者への送信を停止します。「使用しているユーザー」を決める期間はお客様のユースケースによって異なりますが、一般的に、ユーザーが数か月にわたって E メールを開いていないまたはクリックし

ていない場合は、そのようなユーザーを削除することを検討してください (ただし、そのユーザーが実際にはメールを希望していることが明確な場合は除きます)。

- リエンゲージメントキャンペーンとして、最近やり取りが途絶えている相手に意図的に連絡する場合は、細心の注意が必要です。こうした活動はリスクが高く、スパムトラップへの送信だけでなく、バウンスや苦情に関連した問題を引き起こす場合が多くあります。
- メーリングリスト全体にオプトインメッセージを送信し、検証リンクをクリックした受取人のみを維持します。この手順では、リストからアクティブでない受取人を削除することに加え、スパムトラップアドレスを削除するのにも役立ちます。ただし、メーリングリストに多くの不正なアドレスが含まれているか、バウンスの問題がすでに発生していると思われる場合、この手法の使用はお勧めしません。アカウントのバウンス率が上昇する可能性があるためです。

手動調査に関するよくある質問

Q1. マニュアルでの調査のため、アカウントがレビュー対象である、または送信が一時停止になるという通知を受信した場合、どうすればよいですか?

SES の調査担当者が、お客様の送信について重大な問題が発生したことを確認しました。代表的な問題を以下に示します (ただし、これらに限定されません)。

- お客様の送信が、AWS の適正利用規約 (AUP) に違反している。
- お客様の E メールが相手にとって迷惑メールになっているようである。
- お客様のコンテンツは、フィッシング (シミュレートされたフィッシングを含む) に関連しています。
- お客様のコンテンツは、SES でサポートしていないそれ以外のユースケースに関連しています。

問題が解決可能であるとみなされた場合、お客様のアカウントは一定時間レビュー対象になります。アカウントのレビューが行われている間に、E メール送信状況を変更して、問題を修正する必要があります。

問題が解決不能とみなされた、または問題が非常に深刻である場合は、アカウントをまずレビュー対象にするのではなく、E メール送信機能を一時停止する場合があります。

Q2. どのような問題により、E メール送信においてマニュアルのレビューを実行することになりますか?

アカウントのマニュアルでのレビューを開始する原因となる問題は複数あります。以下の理由がありますが、これらに限定されるものではありません。

- 受信者が SES に連絡し、お客様のアカウントから送信された E メールについて苦情を申し出た。
- E メール送信のパターンに異常な変更が検出された。
- 当社のスパムフィルタで一般的な未承諾コンテンツまたは、質の低いコンテンツの E メールの特性が検出された。

お客様のアカウントがレビュー対象になる、または、アカウントの E メール送信機能が一時停止される場合、通知が送信されます。ほとんどの場合、この通知には、問題に関する情報、および実行すべき次のステップに関する情報が含まれています。

Q3. 「未承諾」 E メールとは何ですか?

未承認 E メールは、受信者が受信を明示的にリクエストしなかった E メールです。これには、受信者がある種類のメール (通知など) にサインアップしたにもかかわらず、異なる種類のメール (広告など) が送信された場合が含まれます。

お客様のアカウントがレビュー対象になる、または、アカウントの E メール送信機能が一時停止される場合、通知が送信されます。「未承諾メールの問題により、これらの内いずれかのアクションが実行されます」という通知を受信した場合、AWSコンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージには、次に示す情報を含めてください。

- お客様が送信したすべてのメッセージは受取人が受信を明示的にリクエストしたものですか? また、それらのメッセージは AWS の適正利用規約に準拠していますか?
- ユーザーがお客様またはウェブサイトと具体的にやり取りする、またはユーザーが E メールを要求するのとは異なる方法で E メールアドレスを取得したかどうか。メーリングリストを入手した方法について説明する必要があります。
- 受信登録や受信登録の解除に関するプロセスは、どのように機能しているか。オプトインやオプトアウトのリンクを含める必要があります。

Q4. 手動レビューのため、アカウントがレビュー対象である、または送信が一時停止になるという通知を受信した場合、どうすればよいですか?

問題の原因を特定し、修正します。問題解決のために変更した後に、AWSコンソールにサインインし、サポートセンターに移動します。代理でオープンしたケースに返信してください。メッセージでは、問題を解決するために行ったステップの詳細情報を提供し、これらのステップでどのように今後問題の再発を防ぐことができるかを説明します。お客様が加えた変更により問題が適切に対処されたと見なされた場合は、アカウントのレビュー期間をキャンセルします。

Q5. どのような種類の問題が 「修正可能」と見なされますか?

一般的に、適切な送信手続きの履歴がある場合、または大部分の送信を継続しながら問題のある送信を削除するステップがある場合を、修正可能な状況と考えています。たとえば、3 つの異なる種類の E メールを送信しており、そのうち 1 種類だけが問題のあるメールであるとき、場合によっては、問題のある送信を停止するだけで、残りの送信を継続することができます。

Q6. 問題の原因がわからない場合、どうなりますか?

AWSコンソールにサインインし、サポートセンターに移動します。お客様に代わって当社がオープンしたケースに返信し、問題の原因となったメールのサンプルをリクエストします。

DNS ブラックホールリスト (DNSBL) に関するよくある質問

ドメインネームシステムベースのブラックホールリスト(DNSBLs) Realtime List(RBL)、拒否リスト,ブロックリスト、またはブラックリストと呼ばれます。これは、不要な E メールを送信した疑いのある IP アドレスを E メールプロバイダーに知らせるためのものです。

異なる DNSBL は、E メールの配信性能にさまざまな影響を与えます。このトピックでは、DNSBL が Amazon SES からの E メールの配信に与える影響と、 DNSBL から Amazon SES IP アドレスを削除するためのポリシーについて説明します。

> **Note**
>
> このトピックは、E メールプロバイダーが着信メッセージをブロックするために使用する DNSBL に関するものです。以前にバウンスされた E メールアドレスを所有する受取人に対する E メール送信を Amazon SES でブロックする方法については、「Amazon SES グローバルサプレッションリスト」を参照してください。

Q1. DNSBL は E メール配信にどのような影響を与えますか?

異なる DNSBL は、メッセージの正常な配信に異なる影響を与えます。主要な E メールプロバイダー (Gmail、Hotmail、AOL、Yahoo など) は、Spamhaus が提供する DNSBL など、評価の高いごく少数のブラックリストを認識しているようです。当社の経験によれば、他の DNSBL の影響は少ない傾向がありますが、一部のメールシステムでは特定の DNSBL を他の DNSBL より重視しています。

また、多くの E メールプロバイダーは独自の内部拒否リストを持っています。E メールプロバイダーは、このようなリストを厳重に保護し、一般と共有することはほとんどありません。このようなリストに IP アドレスが追加されると、そのプロバイダーを使用する受信者に E メールを送信する能力に大きな影響を与える可能性があります。

Q2. IP アドレスは、どのようにしてDNSBLに追加されるのですか?

IP アドレスは、いくつかの経路で DNSBL に追加されます。IP アドレスは、スパムトラップに E メールを送信すると、 DNSBL に追加されます。スパムトラップとは、人間のユーザーに属さない E メールアドレスです。スパムトラップは、迷惑メールを収集しスパム発信者を特定するためだけに存在します。いくつかの DNSBL は、個別のユーザーからの IP アドレスの送信を受け入れます。IP アドレス範囲全体の送信を受け入れる DNSBL さえあります。他の DNSBL は E メール管理者の協力によって維持され、管理者が自分のシステムを不正使用していると思われる IP アドレスを含めることができます。

Q3. Amazon SES は、IP アドレスが DNSBL に追加されるのをどのようにして防ぐのですか?

当社のシステムは、不正使用の兆候を探します。IP アドレスが DNSBL に追加される可能性のある送信パターンやその他の特性を検出した場合は、送信者に通知を送信します。状況が深刻な場合、または通知を送信した後に送信者が問題を修正しない場合、送信者が問題を解決するまで E メールを送信する機能が一時停止されます。このように送信ポリシーを適用すると、Amazon の IP アドレスが DNSBL に載せられる可能性が低くなります。

Q4. Amazon SES は IP アドレスを DNSBL から削除できますか?

Amazon では、Amazon SES サービス全体の配信に影響を与える可能性のあるブラックリストや、Gmail、Yahoo、AOL、Hotmail などの主要な E メールプロバイダーを使用する受信者に電子メールを送信する機能に影響を与える可能性のある DNSBL を積極的にモニタリングしま

す。Spamhaus が提供する DNSBL はこのカテゴリに分類されます。これらのいずれかの条件を満たすリストに当社の IP アドレスの 1 つが表示された場合、そのアドレスが DNSBL からできるだけ早く削除されるように迅速に対応します。

Amazon SES サービス全体の配信に影響を与える可能性の低い DNSBL や、主要な E メールプロバイダーへの配信に大きな影響を与えない DNSBL はモニタリングしません。SORBS と UCEPROTECT が提供する DNSBL は、このカテゴリに分類されます。これらのリストを運営するベンダーはリストへの追加やリストからの削除に関して特定の方法に従っているため、これらのリストから当社の IP アドレスを削除することはできません。

Q5. 送信側 IP アドレスが Spamhaus 以外の DNSBL に追加されているという理由で、E メールプロバイダーから E メールを拒否されています。どうすればよいですか?

まず、メッセージがブロックされている原因が確かに DNSBL であるかを確認します。送信側 IP アドレスが DNSBL に追加されていることが E メールの拒否の原因である場合は、通常、次の例のように DNSBL プロバイダー名が記載されたバウンス通知が届きます。

```
554 5.7.1 Service unavailable; Client host [192.0.2.0] blocked using DNSBLName;
 See: http://www.example.com/query/ip/192.0.2.0
```

バウンス通知を受信しても上の例に示すようなメッセージとは異なる場合、E メールプロバイダーがメッセージを拒否した理由は DNSBL 追加とは関係ないとほぼ断定できます。

送信側 IP アドレスが DNSBL に追加されているために E メールプロバイダーから E メールをブロックされたことが確認できる場合、いくつかの対処方法があります。

- メッセージを拒否したドメインのポストマスターに連絡し、スパムフィルタリングポリシーからの例外をリクエストします。一部のポストマスターにはサポートプロセスがあり、このプロセスを説明するポストマスターページを公開していることがあります。連絡しようとしているドメインがポストマスターサポートポリシーを公開していない場合は、postmaster@*example.com* に E メールを送信してポストマスターに連絡できる場合があります。ここで、*example.com* は問題となっているドメインです。ドメインは、ポストマスターのメールボックスを持つことが「RFC 5321」で要求されています。

  ポストマスターに連絡するときは、受信したバウンスコード、送信しようとしている E メールのヘッダー、DNSBL が E メールの配信に及ぼす影響の測定値、E メールが誤って DNSBL されていると信じる理由の情報を説明します。正当な E メールを送信していることを示すためにポストマ

スターに提供できる情報が多くなればなるほど、ポストマスターは例外を作成する可能性が高くなります。

- E メールプロバイダーが対応しない場合、またはポリシーを変更する意思がない場合は、専用 IP アドレスの使用を検討します。専用 IP アドレスはお客様専用のアドレスです。適切な送信プラクティスを実装することで、高いエンゲージメント率を維持し、バウンス、苦情、およびスパムトラップのヒット数を低く抑えることができます。適切な送信プラクティスは、アドレスが DNSBL に追加されるのを阻止できます。

Q6. Gmail、Yahoo、Hotmail などの主要なプロバイダーに送信した E メールがスパムフォルダに追加されます。これは、送信側 IP アドレスが DNSBL に追加されていることが原因ですか?

おそらくそうではありません。大きな影響を与える DNSBL (Spamhaus のいずれかの DNSBL など) に IP アドレスが含まれている場合、主要な E メールプロバイダーは、この IP アドレスをスパムメールフォルダに送信せずに、この IP アドレスからの E メールを完全に拒否します。

通常、主要な E メールプロバイダーは、E メールを拒否せずに受け入れる場合、ユーザーエンゲージメントを考慮して E メールを受信トレイに入れるか、スパムフォルダに入れるかを判断します。ユーザーエンゲージメントとは、ユーザーが以前に受信したメッセージを操作した方法を指します。

メッセージがユーザーの受信トレイに届く確率を高めるには、以下のすべてのベストプラクティスを実行してください。

- E メールアドレスのリストは、絶対にレンタルまたは購入しません。リストのレンタルや購入は AWS利用ポリシー (AUP) 違反であり、状況の如何を問わず、Amazon SES では許可されません。
- E メールの受信を明示的に希望したユーザーにのみ E メールを送信します。世界中の多くの国や管轄区域では、E メールの受信に明示的に同意していない受取人に E メールを送信することは違法です。
- 過去 30〜90 日間に送信したメッセージを開いていないユーザーやメッセージ内のリンクをクリックしていないユーザーには、E メールの送信を停止します。このステップは、エンゲージメント率を高く保つために役立ちます。これにより、今後送信するメッセージが受取人の受信トレイに到着する可能性が高くなります。
- 送信する各メッセージの設計要素と記述スタイルを統一し、ユーザーがメッセージの送信元を簡単に識別できるようにします。
- SPF や DKIM などの E メール認証メカニズムを使用します。

- ユーザーがウェブフォームを使用してコンテンツにサブスクライブする場合、E メールをユーザーに送信し、今後の E メールの受信をユーザーが希望することを確認します。ユーザーが E メールの受信を希望することを確認するまでは、追加の E メールを送信しません。このプロセスは、確定オプトインまたはダブルオプトインとして知られています。
- ユーザーによるサブスクリプション解除手続きを容易にし、サブスクリプション解除リクエストを即座に了承します。
- リンクを含む E メールを送信する場合は、リンクを Spamhaus Domain Block List (DBL) に対してテストして確認します。リンクをテストするには、Spamhaus ウェブサイトの Domain Lookup Tool を使用します。

これらのプラクティスを実装することで、送信者の評価を向上させることができます。これにより、送信する E メールが受信者の受信トレイに届く可能性が高まります。これらのプラクティスを実装することで、アカウントのバウンス率や苦情率を低く抑えることができ、E メールがスパムトラップに送信されるリスクを低減できます。

Amazon SES E メール送信メトリクスに関するよくある質問

Amazon SES は、E メール送信に関するいくつかのメトリクスを収集します。これらのメトリクスを使用して、E メールプログラムの効果分析およびバウンス率や苦情率などの重要な統計のモニタリングができます。

このセクションでは、E メール送信メトリクスに関する以下のトピックについてよくある質問を示します。

- 一般的な質問
- オープンの追跡
- クリックの追跡

> **Note**
>
> イベント追跡は、受信者のメールサービスプロバイダー (ESP) と、Amazon SES の制御の範囲外であるプライバシー設定をどのように構成したかに依存します。トラッキングイベントの数は、次のような条件下で歪む (不正確なカウントを返す) 可能性があります。
>
> - E メール受信者は、プライバシーを保護する E メールサービスプロバイダー (ESP) を使用しています。

- E メール受信者は、ESP にデータを共有する権限を明示的に与えていません。
- E メール受信者の ESP は画像やリンクをキャッシュします。SES は最初のオープンのみカウントし、後続のオープン数をカウントすることはできません。

一般的な質問

Q1. E メールの配信後、いつまで Amazon SES はオープンとクリックのメトリクスを収集しますか?

Amazon SES は各 E メールの送信後 60 日間、オープンとクリックのメトリクスを収集します。

Q2. ユーザーが特定の E メールを複数回開くか、特定の E メール内のリンクを複数回クリックした場合、これらの各イベントが別個に追跡されるのですか?

受信者が E メールを複数回開いた場合、Amazon SES によって各オープンが一意のオープンイベントとしてカウントされます。同様に、受信者が同じリンクを複数回クリックすると、Amazon SES によって各クリックは一意なクリックイベントとしてカウントされます。ただし、上記の注釈欄に記載したシナリオによって、これらのカウントが歪む可能性があります。

Q3. オープンとクリックのメトリクスは集約されますか? または受取人レベルで測定できますか?

オープンとクリックは、受取人レベルで追跡されます。オープンとクリックの追跡により、E メールを開いた受信者や E メール内のリンクをクリックした受信者を判断できます。

Q4. Amazon SES API を使用して開封とクリックのメトリクスを取得できますか?

Amazon SES API は、開封とクリックのメトリクスを取得するメソッドを提供していません。ただし、CloudWatch API を使用して Amazon SES のオープンとクリックのメトリクスを取得できます。たとえば、AWS CLI を使用して、以下のコマンドを発行することで、 CloudWatch API を使用してクリックメトリクスを取得できます。

```
aws cloudwatch get-metric-statistics --namespace AWS/SES --metric-name Click \
  --statistics Sum --period 86400 --start-time 2017-01-01T00:00:00Z \
  --end-time 2017-12-31T23:59:59Z
```

上に示したコマンドは、2017 年の各日のクリックイベントの総数を取得します。開封メトリクスを取得するには、`metric-name` パラメータの値を `Open` に変更します。また、`start-time` および

end-time パラメータを変更して分析期間を変更したり、または、period パラメータを変更してよりきめ細かい分析をしたりできます。

オープンの追跡

Q1. オープンの追跡はどのように行われますか?

Amazon SES を通じて送信される E メールごとに、1 x 1 ピクセルの透明 GIF イメージが挿入されます。E メールには、このイメージファイルへの固有のリファレンスが含まれており、このイメージがダウンロードされると、どのメッセージを誰が開いたかを SES で正確に判断できます。

デフォルトでは、このピクセルは E メールの下部に挿入されます。ただし、E メールプロバイダーのアプリケーションによっては、特定のサイズを超えると E メールのプレビューが切り捨てられ、メッセージの残りの部分を確認するためのリンクが表示される場合があります。このシナリオでは、SES ピクセルの追跡イメージはロードされず、追跡しようとしているオープン率は破棄されます。これを回避するには、必要に応じて E メールの本文に {{ses:openTracker}} プレースホルダーを挿入して、E メールの先頭または他のいずれかの場所にピクセルを配置します。プレースホルダーを含むメッセージを SES が受信すると、そのメッセージはオープンの追跡ピクセルイメージに置き換えられます。

> **Important**
>
> {{ses:openTracker}} プレースホルダーを 1 つだけ追加します。複数のプレースホルダーを追加すると、400 BadRequestException エラーコードが返されます。

この追跡用のピクセルを追加しても、E メールの外観は変わりません。

Q2. オープンの追跡はデフォルトで有効になりますか?

オープンの追跡は、デフォルトですべての Amazon SES ユーザーに利用可能です。オープンの追跡を使用するには、以下の操作を行う必要があります。

1. 設定セットを作成します。
2. 設定セットに、イベント送信先を作成する。
3. 送信先にオープンイベント通知を発行するようにイベント送信先を設定する。
4. オープンの追跡対象である E メールごとにステップ 1 で作成した設定セットを指定する。

設定セットのイベント送信先を通じてオープンの追跡を有効にする方法の詳細については、「the section called “イベント送信先の作成”」を参照してください。ピクセルプレースホルダーは、SMTP メール内で、書式設定およびテンプレート化された raw E メールのような方法で使用できます。

イベント発行を使用して E メール送信をモニタリングする 方法の詳細情報。

Q3. オープンの追跡用ピクセルを特定の E メールから除外できますか?

オープンの追跡用ピクセルを E メールから除外するには 2 つの方法があります。最初の方法では、設定セットを指定しないで E メールを送信します。別の方法として、オープンイベントに関するデータを発行するように設定されていない設定セットを指定できます。

Q4. プレーンテキストメールのオープンは追跡されますか?

オープンの追跡の対象となるのは HTML メールのみです。オープンの追跡にはイメージの挿入が必要であるため、ユーザーがテキスト専用 (HTML 以外) の E メールクライアントで E メールを開くと、オープンメトリクスを収集することはできません。

クリックの追跡

Q1. クリックの追跡はどのように行われますか?

クリックを追跡するには、Amazon SES で E メールの本文の各リンクを変更します。受信者がリンクを開くと、オープンは Amazon SES サーバーに送信され、即座に送信先アドレスに転送されます。オープンの追跡と同様に、各転送先リンクは一意です。これにより、リンクをクリックした受信者、クリックした時間、リンク元の E メールを Amazon SES で判断できます。

> **Important**
>
> 単一のメッセージを複数の宛先に送信した場合、各受取人によって同じクリック追跡リンクが保存されます。個々の受取人のクリックアクティビティを追跡するには、1 回の送信操作につき 1 人の受取人にメールを送信します。

Q2. クリックの追跡を無効化できますか?

E メールの HTML 本文のアンカータグに `ses:no-track` 属性を追加することで、個々のリンクのクリック追跡を無効にできます。たとえば、AWS ホームページにリンクする場合、通常のアンカーリンクは次のようになります。

```
<a href="https://aws.amazon.com">Amazon Web Services</a>
```

そのリンクのクリック追跡を無効にするには、以下のように変更します。

```
<a ses:no-track href="aws.amazon.com">Amazon Web Services</a>
```

ses:no-track はスタンダードの HTML 属性ではないため、受信者の受信トレイに届く E メールのバージョンから Amazon SES によって自動的に削除されます。

特定の設定セットを使用して送信するすべてのメッセージに対してクリック追跡を無効にすることもできます。クリック追跡を無効にするには、設定セットのイベント送信先を変更して、クリックイベントがキャプチャされないようにします。

設定セットのイベント送信先を通じてクリックの追跡を有効/無効にする方法の詳細については、「the section called “イベント送信先の作成”」を参照してください。

イベント発行を使用して E メール送信をモニタリングする 方法の詳細情報。

Q3. 各 E メールで追跡できるリンクの数はいくつでしょうか?

クリック追跡システムでは、最大 250 のリンクを追跡できます。

Q4. プレーンテキストの E メールのリンクに対するクリックメトリクスは収集されますか?

HTML E メールのクリックのみを追跡できます。

Q5. リンクに一意の識別子をタグ付けできますか?

ses:tags 属性を使用すると、メール内のリンクにキーと値のペアとしてタグを追加できます。追加できるタグの数に制限はありません。この属性を使用する場合は、CSS インラインプロパティを渡す場合と同じ形式を使用してキーと値を指定します。この形式では、キーの後にコロン (:)、その後に続けて値を入力します。複数のキー値ペアを渡す必要がある場合は、各ペアをセミコロン (;) で区切ります。

たとえば、リンクにタグとして product:book, genre:fiction, subgenre:scifi, type:newrelease を追加するとします。この場合、結果のリンクは次のようになります。

```
<a ses:tags="product:book;genre:fiction;subgenre:scifi;type:newrelease;"
```

```
    href="http://www.amazon.com/…/">New Releases in Science Fiction</a>
```

これらのタグはイベント発行先に渡され、ユーザーがクリックした特定のリンクに対する追加の分析を実行できます。

> **Note**
>
> リンクタグには、0〜9 の数字、A〜Z の文字 (大文字と小文字)、ハイフン (-)、およびアンダースコア (_) を使用できます。

Q6. 追跡されるリンクは、HTTP または HTTPS プロトコルを使用しますか。

リンクの追跡には E メールの元のリンクと同じプロトコルを使用します。

たとえば、E メールに `https://www.amazon.com` へのリンクが含まれる場合、リンクは HTTPS プロトコルを使用する追跡リンクに置き換えられます。E メールに `http://www.example.com` へのリンクが含まれる場合、リンクは HTTP プロトコルを使用する追跡リンクに置き換えられます。E メールに前述のリンクの両方が含まれる場合、HTTPS リンクは HTTPS プロトコルを使用する追跡リンクに、HTTP リンクは HTTP プロトコルを使用する追跡リンクに置き換えられます。

Q7. 追跡されている E メール内のリンクはありません。理由

Amazon SES では、E メール内のリンクに適切にエンコードされた URL が含まれています。具体的には、リンク内の URL は、RFC 3986 に準拠している必要があります。E メールのリンクが適切にエンコードされていない場合は、受信者は E メールにリンクを表示できますが、Amazon SES はそのリンクのクリックイベントを追跡しません。

不適切なエンコードに関連する問題は、通常、クエリ文字列が含まれている URL で発生します。例えば、E メール内のリンクの URL のクエリ文字列にエンコードされていないスペース文字が含まれている場合 (次の例の 「John」と「Doe」の間のスペースなど http://www.example.com/path/to/page?name=John Doe)、 Amazon SES はリンクを追跡しません。ただし URL でエンコードされた空白文字 (次の例の「%20」など http://www.example.com/path/to/page?name=John%20Doe) が代わりに使用されている場合、Amazon SES は想定どおりに追跡します。

クイック検索インデックス

次のインデックスは、ハウツーまたはコンセプトの 2 つの検索方法を提供することで、Amazon SES で情報をスピーディに見つけられるように作成されています。ハウツーは何かをする「ハウツー」を説明し、コンセプトは全体像を説明しています。

フィードバックをお寄せください

右上端の [Feedback] (フィードバック) ボタンを押して、ご意見をお寄せください。

- このインデックスは役に立ちましたか?
- このインデックスに追加してほしいハウツーまたはコンセプトはありますか?
- もっと別の分類にすべきと思われるものはありましたか?

SES ハウツーとコンセプトのリンク

How-tos

SES のハウツーリンクはアルファベット順にリストアップされ、対応するセクションに移動して、選択したアクションを実行する「ハウツー」を表示します。

- 方法を学ぶ..。
  - カスタム MAIL FROM ドメインの設定の一部として SPF レコードを追加する
  - IP プールを割り当てる
  - E メール受信でスパムをブロックする
  - オープン/クリックのカスタムドメインを設定する
  - SNS 通知を設定する
  - SMTP エンドポイントに接続する
  - 設定セットを作成する
  - ドメイン ID を作成する
  - E メールアドレス ID を作成する
  - イベント送信先を作成する
  - IP アドレスフィルタを作成する
  - 専用 IP を有効にするためのマネージド IP プールを作成する (マネージド)

- 受信ルールを作成する
- CloudWatch を使用して評価アラームを作成する
- カスタムポリシーを使用して送信承認ポリシーを作成する
- Policy Generator を使用して送信承認ポリシーを作成する
- 専用 IP アドレス用の標準専用 IP プールを作成する (標準)
- ID を削除する
- 個人データを削除する
- ID を編集する
- E メールのフィードバック転送を有効にする
- 評価メトリクスをエクスポートする
- サンドボックス外に移動する
- SES の開始方法
- Virtual Deliverability Manager の使用を開始する
- E メールを受信する権限を付与する
- スループットを向上させる
- 送信クォータを引き上げる
- 既存の E メールサーバーと統合する
- API コールをログする
- 設定セットを管理する
- Easy DKIM と BYODKIM を管理する
- 送信とレピュテーションの指標を監視する
- 送信統計情報を監視する
- 使用統計情報を監視する
- 送信クォータを確認する
- ID の DKIM レコードを取得する
- SMTP 認証情報を取得する
- 設定セットレベルの抑制を使用してアカウントレベルのサプレッションリストを上書きする
- E メールアドレス ID に引き継がれた DKIM 署名を上書きする
- E メール送信を一時停止する
- MX レコードを公開する

- AWSリソースの不正使用をレポートする
- 専用 IP アドレスをリクエストする
- テクニカルサポートをリクエストする
- Virtual Deliverability Manager のアドバイザーを使用して、配信性能と評価の問題を解決する
- CloudWatch からイベントデータを取得する
- Kinesis Data Firehose からイベントデータを取得する
- SNS からイベントデータを取得する
- AWS SDK を使用して E メールを送信する
- プログラムで E メールを送信する
- SES API を使用して E メールを送信する
- SMTP を使用して E メールを送信する
- CLI または SES API を使用して添付ファイル付きの raw E メールを送信する
- メールボックスシミュレーターを使用してテスト E メールを送信する
- BYODKIM (自分の DKIM を使用する) を設定する
- DMARC ポリシーを設定する
- Easy DKIM を設定する
- E メール受信を設定する
- イベント発行を設定する
- MAIL FROM ドメインを設定する
- 送信承認を設定する (ID 所有者のタスク)
- 送信承認を設定する (代理送信者のタスク)
- E メールの送信時に設定セットを指定する
- SMTP インターフェイスへの接続のテスト
- バウンス率と苦情率を追跡する
- 継承された DKIM 署名プロパティについて理解する
- 評価メトリクスを使用する
- ソフトウェアパッケージを使用し、E メールを送信する
- サブスクリプション管理を使用する
- テンプレートを使用して E メールを送信する

- アカウントレベルのサプレッションリストを使用する

- ドメイン ID を検証する
- E メールアドレス ID を検証する
- ID を表示する
- Virtual Deliverability Manager ダッシュボードを使用して、アカウントの配信性能メトリクスを概要レベルおよび詳細レベルで表示する
- 専用 IP の SNDS メトリクスを表示する
- 専用 IP アドレスをウォームアップする

Concepts

SES コンセプトのリンクがアルファベット順にリストアップされ、選択したコンセプトを説明する対応する章とセクションに移動します。

- 次に関する情報を確認できます。
  - AWS リソースの不正使用、レポートする
  - アカウントダッシュボード
  - アカウントレベルのサプレッションリスト
  - E メール受信のアクションオプション
  - ヘッダー追加アクション
  - サポート対象外の添付ファイルのタイプ
  - バウンス応答アクションを返す
  - BYODKIM (自分の DKIM を使用する)
  - BYOIP (自分の IP を使用する)
  - コード例
  - コンプライアンス検証
  - 設定セットレベルの抑制の使用
  - 設定セット
  - コンテンツのエンコーディング
  - クロスアカウント通知のレガシーサポート
  - カスタムの MAIL FROM ドメイン
  - データ保護
  - 専用 IP アドレス

- 専用 IP アドレス (マネージド)
- 専用 IP アドレス (標準)
- DKIM での E メールの認証
- DMARC (ドメインベースのメッセージ認証、レポート、および適合性)
- DKIM による DMARC への準拠
- SPF による DMARC への準拠
- Easy DKIM
- E メールのフィードバック転送先
- E メール受信認証
- E メール受信の概念
- E メール受信コンソールウォークスルー
- E メール受信マルウェアスキャン
- E メール受信許可
- E メール受信のユースケース
- E メール受信制限
- E メール送信認証方法
- エンドポイント
- イベント通知
- E メールでのイベント通知
- SNS でのイベント通知
- イベントの発行
- FAQ (よくある質問)
- グローバルサプレッションリスト
- サポートされているヘッダーフィールド
- ID の管理
- ID およびアクセス管理
- インフラストラクチャセキュリティ
- Amazon WorkMail アクションとの統合
- IP アドレスフィルタを使用した IP ベースの制御
- Lambda 関数のアクションを呼び出す

- リスト管理
- リストとサブスクリプション
- ログ記録とモニタリング
- マルウェアの検出
- 手動での DKIM 署名
- イベント発行を使用して E メール送信をモニタリングする
- 送信者評価をモニタリングする
- 送信アクティビティのモニタリング
- クォータ
- 受信ルール
- 受信ルールを使用した受信者ベースの制御
- リージョン
- 評価メトリクス
- 評価メトリクスのメッセージ
- 耐障害性
- S3 バケットアクションへの配信
- サンドボックス - 脱出
- セキュリティ
- サポートされるセキュリティプロトコル
- 送信承認
- 送信承認ポリシーの分析
- 送信承認ポリシーの例
- 送信承認プロセス
- 専用 IP の SNDS メトリクス
- SNS 通知コンテンツ
- SNS 通知の例
- SNS トピックアクションへの公開
- SPF (センダーポリシーフレームワーク)
- ルールセットアクションの停止

- サブスクリプションの管理

- サポート、テクニカルをリクエストする
- カスタム E メール検証用テンプレート
- トラブルシューティング
- 検証済み ID
- Virtual Deliverability Manager
- VPC エンドポイント

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。