aws

开发人员指南

Amazon Simple Email Service

Amazon Simple Email Service: 开发人员指南

Table of Contents

什么是 Amazon SES？

Amazon Simple Email Service（SES）是一个易于使用且经济高效的电子邮件平台，有便于您通过该平台，使用您自己的电子邮件地址和域来发送或接收电子邮件。

举个例子：您可以发出诸如特惠促销之类的营销电子邮件，或下单确认通知等交易相关的电子邮件，还可以发出新闻简讯等其它类型的通信。您在使用 Amazon SES 接收邮件时，还可以开发出各种实用的软件功能，例如：电子邮件自动回复程序、电子邮件取消订阅系统，以及根据所传入的电子邮件生成客服工单的应用程序。

有关与 Amazon SES 相关主题的更多信息，请参阅 AWS 消息收发和目标博客。

优势

对企业来说，构建大规模的电子邮件解决方案通常是一个复杂且成本高昂的挑战。您必须解决电子邮件服务器管理、网络配置和 IP 地址信誉等基础设施难题。除此之外，许多第三方电子邮件解决方案还会要求与您签署合同，而且价格还需要协商，到了使用初期，您还需要支付大量的前期费用。Amazon SES 就为您铺平了道路，因为 Amazon.com 在这方面已经积累了多年经验，并且为服务自身的庞大客户群构建了完善的电子邮件基础设施，而这些都能有利于您获益良多。

相关服务

Amazon SES 与其他 AWS 产品无缝集成。譬如，您可以：

- 为任何应用程序添加发送电子邮件的功能。
- 您可以使用AWS 软件开发工具包、使用 Amazon EC2 S ES S MTP 接口或直接调用 Amaz on S ES API 从亚马逊发送电子邮件。
- 使用 AWS Elastic Beanstalk 创建具备电子邮件功能的应用程序，如使用 Amazon SES 向客户发送新闻通讯的程序。
- 设置 Amazon Simple Notification Service（Amazon SNS），在您的电子邮件被退回、产生投诉或已成功送达收件人的邮件服务器时通知您。在使用 Amazon SES 接收电子邮件时，您的电子邮件内容可以发布到 Amazon SNS 主题。
- 使用 AWS Management Console 来设置 Easy DKIM，这是一种对您的电子邮件进行身份验证的方法。尽管您可以通过任何 DNS 提供商来使用 Easy DKIM，但使用 Route 53 管理域的话，设置起来格外简单。

- 使用 AWS Identity and Access Management （ IAM ）控制用户对您的电子邮件发送的访问权限。
- 将您收到的电子邮件存储在 Amazon Simple Storage Service（ Amazon S3 ）中。
- 触发 AWS Lambda 函数，对您收到的电子邮件执行操作。
- 使用 AWS Key Management Service （ AWS KMS ）选择性地加密在 Amazon S3 存储桶中收到的邮件。
- 使用 AWS CloudTrail 记录您使用控制台或 Amazon SES API 进行的 Amazon SES API 调用。
- 将您的电子邮件发送事件发布到亚马逊 CloudWatch或亚马逊 Data Firehose。如果您将电子邮件发送事件发布到 Firehose，则可以在亚马逊 Redshi ft、亚马逊服务或亚马逊 S OpenSearch 3 中访问这些事件。

定价

有了 Amazon SES，您就可以按照所发送和所接收到的电子邮件数量付费。有关详情，请参阅 Amazon SES 定价。

区域和 Amazon SES

SES 已 AWS 区域 在全球多个地区推出。在每个区域中， AWS 维护多个可用区。这些可用区的物理位置是相互隔离的，但可通过私有、低延迟、高吞吐量和高度冗余的网络连接联合在一起。这些可用区使我们能够提供极高水平的可用性和冗余，同时最大程度地减少延迟。

有关所有 SES 区域端点的列表，请参阅《AWS 一般参考》中的 Amazon Simple Email Service 端点和配额。要详细了解每个区域中可用的可用区数量，请参阅 AWS 全球基础设施。

本部分包含您计划在多个 AWS 区域中使用 SES 时需要了解的信息。它讨论了以下主题：

- SES 区域和端点
- 沙盒移除和发送限制提高
- 电子邮件地址和域的验证
- Easy DKIM
- 账户级黑名单
- 反馈通知
- SMTP 凭证

- 发送授权
- 用于自定义 MAIL FROM 域的反馈端点
- 电子邮件接收
- 设置 (MX) 记录

有关的一般信息 AWS 区域，请参阅《AWS 一般参考》中的AWS 服务端点。

SES 区域和端点

当您使用 SES 发送电子邮件时，将会连接到一个为 SES API 或 SMTP 接口提供端点的 URL。《AWS 一般参考》包含您通过 SES 发送和接收电子邮件所用的端点的完整列表。有关更多信息，请参阅 Amazon Simple Email Servic AWS 一般参考 e 终端节点和配额，具体部分如下所示：

- API 端点 — 当您通过 SES 发送电子邮件时，您可以使用下表中 URLs 列出的内容向 SES API 发出 HTTPS 请求。
- SMTP 端点-使用 SMTP 接口时，您可以使用下表中 URLs 列出的来发送电子邮件。
- 电子邮件接收终端节点-如果您已将 SES 配置为接收发送到您的域的电子邮件，则在域的 DNS 设置中设置邮件交换器 (MX) 记录时，可以使用此表中 URLs 列出的入站 SMTP 终端节点。

> **Note**
>
> 入站 SMTP URLs 不是 IMAP 服务器地址。换而言之，您不能通过它们使用 Outlook 等应用程序接收电子邮件。有关为传入电子邮件提供 IMAP 服务器的服务，请参阅 Amazon WorkMail。

沙盒移除和发送限制提高

您账户的沙盒状态在不同 AWS 区域中可能不同。换而言之，如果您的账户已从美国西部（俄勒冈州）区域的沙盒中删除，但它可能仍在美国东部（弗吉尼亚州北部）区域的沙盒中，除非您也在该区域将其从沙盒中删除。

根据 AWS 区域，发送限制也可能不同。例如，如果您的账户在欧洲地区（爱尔兰）中可以每秒发送 10 封邮件，在其他区域中可以发送邮件的数量也许会更多或更少。

当您提交将您的账户从沙盒中移除的请求时，或者当您提交请求以提高账户的发送配额时，请确保选择您的请求应用到的所有 AWS 区域 。您可在一个支持中心案例中提交多个请求。

电子邮件地址和域的验证

在使用 SES 发送电子邮件之前，您必须验证自己拥有计划从中发送电子邮件的电子邮件地址或域。电子邮件地址和域的验证状态在不同 AWS 区域中也会不同。例如，如果您在美国西部（俄勒冈州）区域中验证了域，在美国东部（弗吉尼亚州北部）区域中再次验证该区域之前，您无法使用该域来发送电子邮件。有关验证电子邮件地址和域的更多信息，请参阅Amazon SES 中已验证的身份。

Easy DKIM

你必须为每个要使用 Easy DKIM AWS 区域 的地方执行 Easy DKIM 设置过程。也就是说，在每个区域中，您必须使用 SES 控制台或 SES API 来生成 CNAME 记录。接下来，您必须将所有 CNAME 记录添加到您域的 DNS 配置中。有关设置 Easy DKIM 的更多信息，请参阅Amazon SES 中的 Easy DKIM。

并非所有人都 AWS 区域 使用默认的 SES DKIM 域名，`dkim.amazonses.com`要查看您的地区是否使用特定区域的 DKIM 域，请查看中的 DKIM 域名表。AWS 一般参考

账户级黑名单

您的 SES 账户级别禁止列表 AWS 账户 仅适用于当前的。 AWS 区域您可以使用 Amazon SES API v2 或控制台手动在账户级黑名单中逐个或批量添加或删除地址。有关使用账户级黑名单的更多信息，请参阅 使用 Amazon SES 账户级黑名单。

反馈通知

在多个 AWS 区域中设置反馈通知时，需要注意以下两点：

- 已验证的身份设置（例如，您通过电子邮件还是 SNS 接收反馈）仅适用于您执行设置的区域。例如，如果您在美国西部（俄勒冈州）和美国东部（弗吉尼亚州北部）区域验证 user@example.com，并且您希望通过 SNS 通知接收被退回的电子邮件，则必须使用 SES API 或 SES 控制台为两个区域的 user@example.com 设置 SNS 反馈通知。
- 您用于反馈转发的 SNS 主题必须位于使用 SES 的同一个区域中。

有关通过反馈通知监控发送活动的更多信息，请参阅为 Amazon SES 设置事件通知。

SMTP 凭证

您通过 SES SMTP 接口发送电子邮件所用的凭证在各个 AWS 区域中是唯一的。如果您使用 SES SMTP 接口在多个区域中发送电子邮件，则必须为每个区域生成一组 SMTP 凭证。

> **Note**
>
> 如果您在 2019 年 1 月 10 日之前创建了 SMTP 凭证，则您的 SMTP 凭证是使用旧版本的签名创建的 AWS 。出于安全考虑，您应删除在此日期之前创建的凭证，并将其替换为较新的凭证。您可以使用 IAM 控制台删除较早的凭证。

用于自定义 MAIL FROM 域的反馈端点

如果您使用自定义 MAIL FROM 域，SES 会要求发布 MX 记录，以便您的域可以接收电子邮件提供商向您发送的退回邮件和投诉通知。您可以将相同的自定义 MAIL FROM 域用于不同的已验证身份，AWS 区域 因为退回通知和投诉通知会发送到特定地区的反馈端点。

配置自定义 MAIL FROM 域时，SES 会自动为自定义 MAIL FROM 配置所在的区域指定正确的反馈端点。此端点在 MX 记录的值字段中提供，供您发布（添加）到域的 DNS 配置。

自定义 MAIL FROM 设置过程在 使用自定义 MAIL FROM 域中介绍。作为参考，SES 用于不同内容的反馈端 AWS 区域 点列在中的反馈端点表中 AWS 一般参考。

发送授权

委托发件人只能从验证身份所有者身份 AWS 区域 的地方发送电子邮件。给委托发件人授予权限的发送授权策略必须附加到该区域中的身份。有关发送授权的更多信息，请参阅使用 Amazon SES 的发送授权。

电子邮件接收

除 Amazon S3 存储桶外，您用于通过 SES 接收电子邮件的所有 AWS 资源都必须与 SES 终端节点 AWS 区域 相同。例如，如果您在美国西部（俄勒冈州）区域使用 SES，那么您使用的任何 SNS 主题、KMS 键和 Lambda 函数也必须在美国西部（俄勒冈州）区域。同样，要在同一个区域中通过 SES 接收电子邮件，您必须在该区域中创建一个有效的接收规则集。有关电子邮件接收的概念和设置过程，请参阅使用 Amazon SES 接收电子邮件。

中的电子邮件接收端点表 AWS 一般参考 列出了 SES 支持电子邮件接收的所有终端节点 AWS 区域 的电子邮件接收端点。

Amazon SES 中的服务配额

以下各节列出并介绍了适用于 Amazon SES 资源和操作的配额。一些配额可以提升，而另一些配额不能提升。要确定您是否可以请求提升配额，请参阅 Adjustable（可调整）列。

> **Note**
>
> SES 配额适用于您在中 AWS 区域 使用的每个配额 AWS 账户。

电子邮件发送限额

以下配额适用于通过 SES 发送电子邮件。

发送配额

配额取决于收件人的数量，而不是邮件的数量。

资源	默认配额	可调整
每 24 小时周期可发送的电子邮件数	如果您的账户在沙盒中，则每 24 小时周期最多可以发送 200 封电子邮件。 如果您的账户在沙盒之外，此数量将根据您的具体使用情形而异。	是
每秒可发送的电子邮件数（发送速率）	如果您的账户在沙盒中，则可以每秒发送 1 封电子邮件。 如果您的账户在沙盒之外，此速率将根据您的具体使用情形而异。	是

邮件配额

资源	默认配额	可调整
使用 SES v1 API - 最大邮件大小（包括附件）	每封邮件 10 MB (使用 base64 编码之后)。	否（对于邮件大小超过 10 MB 的工作负载，请考虑迁移到 SES v2 API。）

资源	默认配额	可调整
使用 SES v2 API 或 SMTP - 最大邮件大小（包括附件）	每封邮件 40 MB（使用 base64 编码之后）。	否

> **Note**
>
> 大于 10 MB 的邮件受带宽限制的影响，并且根据您的发送速率，您可能会被限制至 40 MB/秒。例如，您可以按每秒 1 封邮件的速率发送 40 MB 的邮件，也可以每秒发送两封 20 MB 邮件。

发送人和接收人配额

资源	默认配额	可调整
每封邮件的最大收件人数	每封邮件 50 个收件人。 **Note** 收件人是任何“To”、“CC”或“BCC”地址。	收件人限制不可调整。
可以验证的最大身份数	每个 10,000 个 AWS 区域身份 **Note** 身份是您用于通过 SES 发送电子邮件的域或电子邮件地址。	请联系您的 AWS 客户经理讨论您的用例。
专用 IP 池的最大数量（包括托管式和标准 IP 池）	50	不可以

与事件发布相关的配额

资源	默认配额	可调整
最大配置集数	10000	否
配置集名称的最大长度	配置集名称最多可包含 64 个字母数字字符。它们还可包含连字符 (-) 和下划线 (_)。名称不能包含空格、重音字符或任何其他特殊字符。	否
每个配置集的最大事件目标数	10	否
每个 CloudWatch 事件目标的最大维度数	10	否

电子邮件模板配额

资源	默认配额	可调整
每个模板中的最大电子邮件模板数量 AWS 区域	20000	否
最大模板大小	500 KB	否
每个模板中的最大替换值数量	无限制	不适用
每个模板化电子邮件的最大收件人数	50 个目标。目标是指“收件人”、“抄送”或“密件抄送”行中的任何电子邮件地址。 **Note** 您可以在单次 API 调用中联系的目标数可能受	否

资源	默认配额	可调整
	您的账户的最大发送速率限制。	

电子邮件接收配额

下表列出了与通过 SES 接收电子邮件相关联的配额。

资源	默认配额	可调整
每个接收规则集的最大规则数	200	否
每个接收规则的最大操作数	10	否
每个接收规则的最大收件人数	500	否
每个接收规则集的最大数量 AWS 账户	40	否
每个 IP 地址过滤器的最大数量 AWS 账户	100	否
Amazon S3 存储桶中可存储的最大电子邮件大小（包括标头）	40 MB	否
使用 Amazon SNS 通知可发布的最大电子邮件大小（包括标头）	150 KB	否
使用 Amazon SNS 通知可发布的最大电子邮件标头大小	10 KB	否
使用 AWS Lambda 函数可发布的最大电子邮件标头大小	50 KB	否

Mail Manager 配额

下表列出了与 Mail Manager 相关联的配额。

资源	默认配额	可调整
最大打开入口端点数	10	否
最大授权入口端点数	50	不可以
每封邮件的最大收件人数	100	否
最大电子邮件大小（包括标头）	40 MB	否
最大流量策略声明数	20	否
最大流量策略声明条件数	10	否
每个区域的最大流量策略数	100	否
最大 SMTP 中继数	100	否
每个区域的地址列表的最大数量	100	否
每个地址列表的最大地址数	100000	否
最大规则集数	40	否
每个消息的最大规则执行数	40	否
每个规则最大条件数	10	否
每个规则的最大操作数	10	否
每个规则集的最大中继或发送操作数	10	否
最大有效存档数	10	否

资源	默认配额	可调整
存档搜索结果的最大数量	1000	否
导出的档案搜索结果的最大数量	250,000	否
最大并行运行搜索请求数	1	否
最大并行运行导出请求数	1	否
每周存档的最大保留更改数	1	否

常规配额

下表列出了适用于通过 SES 发送和接收电子邮件的配额。

SES API 发送配额

资源	默认配额	可调整
您可以调用 Amazon SES API 操作的速率	所有操作（除了 `SendEmail` 、`SendRawEmail` 和 `SendTemplatedEmail` ）均限制为每秒 1 次请求。	否
MIME 部分	500	否

Amazon SES 凭证的类型

要与 Amazon Simple Email Service（Amazon SES）进行交互，您将使用安全凭证来验证您的身份以及您是否有权与 Amazon SES 进行交互。凭证有很多类型，您使用的凭证取决于您要执行的操作。例如，当您使用 Amazon SES API 来发送电子邮件时将使用 AWS 访问密钥，当您使用 Amazon SES SMTP 接口来发送电子邮件时将使用 SMTP 凭证。

下表列出了您可能会用于 Amazon SES 的凭证的类型，具体取决于您要执行的操作。

如果要访问...	使用这些凭证	凭证包含的内容	如何获取凭证
Amazon SES API （您可以直接访问 Amazon SES API，也可以通过 AWS 软件开发工具包 AWS Command Line Interface、或间接访问 AWS Tools for Windows PowerShell。）	AWS 访问密钥	访问密钥 ID 和秘密访问密钥	请参阅 https://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#access-keys-and-secret-access-keys 中的AWS 一般参考访问密钥。 **Note** 为了获得最佳安全实践，请使用 AWS Identity and Access Management (IAM) 用户访问密钥代替 AWS 账户 访问密钥。您的 AWS 账户 证书授予您对所有 AWS 资源的完全访问权限，因此您应将其存储在安全的地方，改为使用 IAM 用户证书进行 day-to-day 交互 AWS。有关更多信息，请参阅《AWS 一般参考》中的根账户凭证与 IAM 用户凭证。
Amazon SES SMTP 接口	SMTP 凭证	用户名和密码	请参阅 获取 Amazon SES SMTP 凭证。 **Note** 尽管您的 Amazon SES SMTP 凭证不同于您的 AWS 访问密

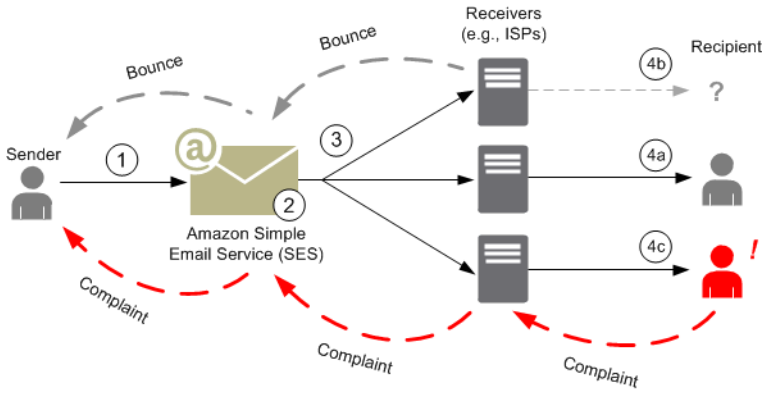
如果要访问...	使用这些凭证	凭证包含的内容	如何获取凭证
			钥和 IAM 用户访问密钥，但 Amazon SES SMTP 凭证实际上是一种 IAM 凭证。IAM 用户可以创建 Amazon SES SMTP 证书，但根账户所有者必须确保 IAM 用户的策略授予他们访问以下 IAM 操作的权限：“iam:”、“iam:ListUsers”、“iam:CreateUser”、“iam:CreateAccessKey” 和 “iam:PutUserPolicy”。

如果要访问...	使用这些凭证	凭证包含的内容	如何获取凭证
Amazon SES 控制台	IAM 用户名和密码 或 电子邮件地址和密码	IAM 用户名和密码 或 电子邮件地址和密码	请参阅 https://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#iam-user-name-and-password 的 IAM 用户名和密码和AWS 一般参考电子邮件地址和密码。 **Note** 出于安全最佳实践，请使用 IAM 用户名和密码，而不使用电子邮件地址和密码。电子邮件地址和密码组合仅供您使用 AWS 账户，因此您应将其存储在安全的地方，而不是使用它们进行 day-to-day交互 AWS。有关更多信息，请参阅《AWS 一般参考》中的根账户凭证与 IAM 用户凭证。

有关不同类型 AWS 安全证书（SMTP 证书除外，它们仅用于 Amazon SES）的更多信息，请参阅中的AWSAWS 一般参考安全证书。

Amazon SES 中电子邮件发送的工作原理

本主题介绍使用 SES 发送电子邮件时发生的情况以及在发送电子邮件后可能发生的各种结果。下图是发送过程的高度概述：



1. 充当电子邮件发件人的客户端应用程序向 SES 提出向一个或多个收件人发送电子邮件的请求。
2. 如果请求有效，SES 将接受电子邮件。
3. SES 通过 Internet 将邮件发送给收件人。一旦邮件传递到 SES，它通常会立即被发送，第一次送达尝试通常发生在几毫秒内。
4. 此时，会存在不同可能性。例如：
    a. ISP 成功地将邮件送达至收件人的收件箱。
    b. 收件人的电子邮件地址不存在，因此 ISP 将退信通知发送到 SES。然后，SES 将该通知转发给发件人。
    c. 收件人收到邮件，但将它视为垃圾邮件并向 ISP 提出投诉。已与 SES 一起设置反馈循环的 ISP 会将该投诉发送到 SES，后者再将投诉转发给发件人。

以下部分回顾了在发件人向 SES 发送电子邮件请求之后以及在 SES 向收件人发送电子邮件之后可能出现的各种结果。

在发件人向 SES 发送电子邮件请求之后

当发件人向 SES 提出发送电子邮件的请求时，调用可能会成功或失败。以下部分介绍每种情况下会发生的情况。

成功的发送请求

如果发送到 SES 的请求成功，SES 将向发件人返回成功响应。此邮件包含一个邮件 ID，这是一个唯一地标识请求的字符串。您可以使用该邮件 ID 来标识发送的邮件或跟踪发送时遇到的问题（您必须在标识符与 SES 在接受电子邮件时传回给您的 SES 邮件 ID 之间存储您自己的映射关系）。然后，SES 基于请求参数汇编电子邮件，扫描邮件是否存在可疑的内容和病毒，然后通过 Internet 使用简单邮件

传输协议（SMTP）发送该邮件。您的邮件通常会立即被发送；其中第一次送达尝试通常发生在几毫秒内。

> **Note**
>
> 如果 SES 接受发件人的请求，然后确定该邮件包含病毒，SES 会停止处理邮件，并且不会尝试将其传输到收件人的邮件服务器。

失败的发送请求

如果发件人向 SES 发送的电子邮件发送请求失败，SES 会向发件人发送一个出错响应并丢弃该电子邮件。请求失败可能会有多种原因。例如，请求的格式可能不正确，或者电子邮件地址可能未由发件人进行验证。

您可用来确定请求是否失败的方法取决于您调用 SES 的方式。以下是如何返回错误和异常的示例：

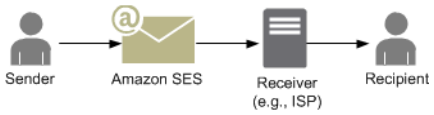
- 如果您通过查询 (HTTPS) API (`SendEmail` 或 `SendRawEmail`) 调用 SES，则这些操作将返回错误。有关更多信息，请参阅 Amazon Simple Email Service API 参考。
- 如果您使用的是使用异常的编程语言的 AWS SDK，则对 SES 的调用将抛出MessageRejectedException。(异常的名称可能因软件开发工具包而略有不同。)
- 如果您使用的是 SMTP 接口，则发件人会收到 SMTP 响应代码，但传递错误的方式取决于发件人的客户端。某些客户端可能显示错误代码；而其他客户端则不显示。

有关使用 SES 发送电子邮件时可能发生的错误的信息，请参阅 Amazon SES 电子邮件发送错误。

在 Amazon SES 发送电子邮件之后

如果发件人向 SES 发送的请求成功，SES 将发送电子邮件且会产生以下任一结果：

- 成功送达且收件人不反对接受电子邮件 – ISP 接受电子邮件，然后 ISP 将电子邮件送达至收件人。成功的送达如下图中所示。

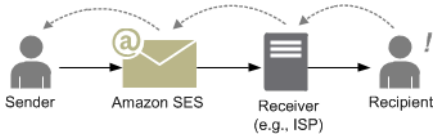


- 硬退信 – 电子邮件由于某种持续存在的情形而被 ISP 拒绝，或者由于电子邮件地址位于 SES 黑名单上而被 SES 拒绝。如果某个电子邮件地址近期对任何 SES 客户造成了“硬退信”，则该邮件地址将被列入 SES 黑名单。在 ISP 中出现硬退信的原因可能是收件人地址无效。ISP 会将硬退信通知发送回

SES，然后 SES 会通过电子邮件或 Amazon Simple Notification Service（Amazon SNS）通知发件人，具体取决于发件人的设置。SES 以同样方式通知发件人出现黑名单退信。来自 ISP 的硬退信的路径如下图所示。

- 软退信 – ISP 可能因某种暂时状况（例如，ISP 太忙而无法处理请求或收件人的邮箱已满）而无法向收件人送达电子邮件。如果域不存在，也可能发生软退信。ISP 将软退信通知发送回 SES，或者，如果出现不存在的域，则 SES 无法找到该域的电子邮件服务器。在任一情况下，SES 都会在一段较长时间内重新尝试发送电子邮件。如果 SES 无法在该时间内送达电子邮件，则它会通过电子邮件或 Amazon SNS 向发件人发送退信通知。如果 SES 可以在重试期间将电子邮件送达至收件人，则表示送达成功。软退信如下图所示。在这种情况下，SES 会重试发送电子邮件，而且 ISP 最终能够将邮件送达至收件人。

- 投诉 – ISP 接受电子邮件并将它送达至收件人，但收件人将该电子邮件视为垃圾邮件并在其电子邮件客户端中单击了类似“标记为垃圾邮件”的按钮。如果 SES 与 ISP 一起设置了反馈循环，则投诉通知会发送到 SES，然后 SES 将投诉通知转发给发件人。大多数都 ISPs 没有提供提交投诉的收件人的电子邮件地址，因此 SES 的投诉通知根据原始邮件的收件人和 SES 收到投诉的 ISP 向发件人提供了一份可能已发送投诉的收件人名单。投诉的路径如下图所示。



- 自动响应 – ISP 接受电子邮件，然后 ISP 将邮件送达至收件人。然后，ISP 会向 SES 发送自动响应，例如 out-of-the-office (OOTO) 消息。然后 SES 将自动响应通知转发给发件人。自动响应如下图所示。

请确保您的支持 SES 的程序不会重试发送生成自动响应的消息。

> **Tip**
>
> 您可以使用 SES 邮箱模拟器测试成功送达、退信、投诉、OOTO 或地址位于黑名单上时发生的情况。有关更多信息，请参阅 手动使用邮箱模拟器。

Amazon SES 中的电子邮件格式

当客户端向 Amazon SES 提出请求时，Amazon SES 构建符合 Internet 邮件格式规范（RFC 5322）的电子邮件。一封电子邮件包含标头、正文和信封，如下所述。

- 标头 – 包含路由说明和有关邮件的信息。例如，发件人的地址、收件人的地址、主题和日期。标头类似于普通信件顶部的信息，但前者可能包含许多其他类型的信息，如邮件的格式。
- 正文—包含邮件本身的文本。
- 信封 – 包含 SMTP 会话期间电子邮件客户端与邮件服务器之间通信的实际路由信息。此电子邮件信封信息类似于普通信件信封上的信息。电子邮件信封的路由信息通常与电子邮件标头中的路由信息相同，但也不总是相同。例如，当您发送密件抄送 (BCC) 时，实际收件人地址 (源自信封) 与收件人的电子邮件客户端中显示的“收件人”地址 (源自标头) 不相同。

以下是电子邮件的简单示例。标头后跟一个空白行，然后是电子邮件的正文。信封未显示，因为它是在 SMTP 会话期间在客户端与邮件服务器之间通信的，而不是电子邮件本身的一部分。

```
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
```

以下部分回顾电子邮件标头和正文并确定您使用 Amazon SES 时需要提供的信息。

电子邮件标头

每封电子邮件都有一个标头。标头的每一行包含一个字段，后跟冒号，然后跟字段正文。当您在电子邮件客户端中阅读电子邮件时，电子邮件客户端通常会显示以下标头字段的值：

- To—邮件收件人的电子邮件地址。
- CC—邮件的抄送收件人的电子邮件地址。
- From—发送电子邮件的电子邮件地址。
- Subject—邮件主题的摘要。
- Date—发送电子邮件的时间和日期。

有许多其他的提供路由信息和描述邮件内容的标头字段。电子邮件客户端通常不会向用户显示这些字段。有关 Amazon SES 接受的标头字段的完整列表，请参阅Amazon SES 标头字段。当您使用 Amazon SES 时，您特别需要了解“From”、“Reply-To”和“Return-Path”标头字段之间的差异。如前所述，“From”地址是邮件发件人的电子邮件地址，而“Reply-To”和“Return-Path”则如下所示：

- Reply-To – 回复将发送到的电子邮件地址。默认情况下，回复将发送到原始发件人的电子邮件地址。
- Return-Path – 退回邮件和投诉应发送到的电子邮件地址。“Return-Path”有时称为“envelope from”、“envelope sender”或“MAIL FROM”。

> **Note**
>
> 当您使用 Amazon SES 时，我们建议您始终设置“Return-Path”参数，以便您可以在发生退回邮件时知道情况并采取纠正措施。

要轻松地将退回的邮件与其目标收件人进行匹配，您可以使用可变信封退回路径 (VERP)。利用 VERP，您为每个收件人设置不同的“Return-Path”，因此，如果有退回邮件，您会自动知道退回邮件的收件人，而不必打开退回邮件并进行分析。

电子邮件正文

电子邮件正文包含邮件的文本。正文可采用以下格式进行发送：

- HTML—如果收件人的电子邮件客户端可解释 HTML，则正文可以包含格式化的文本和超链接
- 纯文本—如果收件人的电子邮件客户端是基于文本的，则正文不得包含任何不可打印的字符。
- HTML 和纯文本—当您使用这两种格式在一封邮件中发送相同内容时，收件人的电子邮件客户端会基于其功能决定显示哪种格式。

如果您要向大量收件人发送一封电子邮件，合理的做法是同时采用 HTML 和文本进行发送。有些收件人拥有支持 HTML 的电子邮件客户端，因此他们可以单击邮件中的嵌式超链接。使用基于文本的电子邮件客户端的收件人需要您提供信息 URLs ，以便他们可以使用 Web 浏览器复制和打开。

您需要向 Amazon SES 提供的电子邮件信息

当使用 Amazon SES 发送电子邮件时，您需要提供的电子邮件信息取决于您调用 Amazon SES 的方式。您可以提供最少量的信息并让 Amazon SES 为您处理所有格式设置。或者，如果您希望执行某些更高级的操作，如发送附件，则可以自行提供原始邮件。下面几节回顾使用 Amazon SES API、Amazon SES SMTP 接口或者 Amazon SES 控制台来发送电子邮件时需要提供的内容。

Amazon SES API

如果您直接调用 Amazon SES API，则可以调用 `SendEmail` 或 `SendRawEmail` API。您需要提供的信息量取决于您调用的 API。

- `SendEmail` API 要求您仅提供源地址、目标地址、邮件主题和邮件正文。您可以选择提供“Reply-To”地址。当您调用此 API 时，Amazon SES 会自动汇编一封格式正确的分为多个部分的多用途 Internet 邮件扩展（MIME）电子邮件，并针对电子邮件客户端软件的显示进行优化。有关更多信息，请参阅 使用 Amazon SES API 发送格式化的电子邮件。
- `SendRawEmail` API 为您提供了灵活性，允许您通过指定标头、MIME 部分和内容类型，设置自己的原始电子邮件的格式并发送这些邮件。`SendRawEmail` 通常由高级用户使用。您需要提供按照 Internet 邮件格式规范 (RFC 5322) 中的要求指定的邮件正文和所有标头字段。有关更多信息，请参阅 使用 Amazon SES API v2 发送原始电子邮件。

如果您使用 AWS 软件开发工具包调用 Amazon SES API，则需要将上面列出的信息提供给相应的函数（例如，`SendEmail`和 Java `SendRawEmail` 的函数）。

有关使用 Amazon SES API 来发送电子邮件的更多信息，请参阅使用 Amazon SES API 发送电子邮件。

Amazon SES SMTP 接口

当您通过 SMTP 接口访问 Amazon SES 时，您的 SMTP 客户端应用程序会汇编此邮件，因此您需要提供的信息取决于您所使用的应用程序。客户端与服务器之间的 SMTP 交换至少需要一个源地址、一个目标地址和邮件数据。

有关使用 Amazon SES SMTP 接口来发送电子邮件的更多信息，请参阅使用 Amazon SES SMTP 接口发送电子邮件。

Amazon SES 控制台

当您使用 Amazon SES 控制台来发送电子邮件时，您需要提供的信息量取决于您是选择发送已设置格式的电子邮件还是发送原始电子邮件。

- 要发送已设置格式的电子邮件，您需要提供源地址、目标地址、邮件主题和邮件正文。Amazon SES 会自动汇编一封格式正确的分为多个部分的 MIME 电子邮件，并针对电子邮件客户端软件的显示进行优化。您还可以指定 reply-to 和 return path 字段。
- 要发送原始电子邮件，您需要提供源地址、目标地址和邮件内容，其中必须包含按照 Internet 邮件格式规范 (RFC 5322) 中的要求指定的邮件正文和所有标头字段。

了解 Amazon SES 中的电子邮件送达率

您希望收件人阅读您的电子邮件，认为邮件有价值，而不是将其标记为垃圾邮件。换言之，您希望最大程度地提高电子邮件送达率 – 到达收件人收件箱的电子邮件的百分比。本主题将回顾您在使用 Amazon SES 时应熟悉的电子邮件送达率概念。

为了最大程度地提高电子邮件送达率，您需要了解电子邮件送达问题，主动采取预防问题发生的措施，随时了解您发送的电子邮件的状态，然后改进电子邮件发送程序 (如有必要)，以进一步提高邮件成功送达的可能性。以下部分回顾这些步骤背后的概念以及 Amazon SES 如何帮助您完成该过程。

了解电子邮件送达问题

在大多数情况下，您的邮件都会成功送达至预期的收件人。但是，在某些情况下，邮件送达可能会失败，或者某收件人可能不希望接收您发送的邮件。退回邮件、投诉和黑名单与这些送达问题相关，这些内容将在以下部分进行介绍。

退回邮件

如果您的收件人的接收方（例如，电子邮件提供商）未能将您的邮件送达至收件人，则接收方会将邮件退回到 Amazon SES。然后，Amazon SES 会通过电子邮件或者 Amazon Simple Notification Service（Amazon SNS）通知您退回的邮件，具体取决于您设置系统的方式。有关更多信息，请参阅为 Amazon SES 设置事件通知。

有查无此人的邮件 和软退回邮件，如下所示：

- 查无此人的邮件 – 一种持久性的电子邮件传送失败。例如，邮箱不存在。Amazon SES 不会重试传送查无此人的邮件，但 DNS 查找故障除外。我们强烈建议您不要尝试重复传送至属于查无此人的邮件的电子邮件地址。

- 软退回邮件 – 一种临时电子邮件传送失败。例如，邮箱已满、存在太多连接（也称为节流）或者连接超时。Amazon SES 会多次重试传送软退回邮件。如果电子邮件仍无法传送，则 Amazon SES 停止重试。

Amazon SES 会通知您将不再重试的查无此人的邮件和软退回邮件。但是，只有查无此人的邮件会计入您使用 Amazon SES 控制台或 `GetSendStatistics` API 检索到的退回邮件率和退回邮件指标。

退回邮件也可以是同步 或异步 的。同步退回邮件发生在发件人和接收方的电子邮件服务器正在通信时。异步退回邮件发生在接收方最初接受电子邮件进行传送但后来未能将邮件送达至收件人时。

投诉

大多数电子邮件客户端程序会提供一个带有“标记为垃圾邮件”标签的按钮或类似按钮，用于将邮件移至垃圾邮件文件夹并将它转发给电子邮件提供商。此外，大多数电子邮件提供商会维护一个滥用地址 (例如，abuse@example.net)，用户可以将不需要的电子邮件转发到此地址并请求电子邮件提供商采取措施阻止它们。在这两种情况下，收件人会提出投诉。如果电子邮件提供商认为您是垃圾邮件发送者，而且 Amazon SES 与电子邮件提供商一起设置了反馈循环，则电子邮件提供商会将投诉发送回 Amazon SES。当 Amazon SES 收到此类投诉时，会通过电子邮件或使用 Amazon SNS 通知将投诉转发给您，具体取决于系统的设置方式。有关更多信息，请参阅 为 Amazon SES 设置事件通知。我们建议您不要尝试重复传送至产生投诉的电子邮件地址。

全局黑名单

Amazon SES 全局黑名单由 SES 拥有和管理，用于保护 SES 共享 IP 池中地址的声誉，其中包含最近导致任何 SES 客户硬退回的收件人电子邮件地址。如果尝试通过 SES 向黑名单中的地址发送电子邮件，您可以成功调用 SES，但 SES 会将该邮件视为“查无此人的邮件”，而不会尝试将其发送出去。与“查无此人的邮件”类似，黑名单退回邮件也会计入发送配额和退回邮件率。电子邮件地址可在黑名单上保留最多 14 天。如果您确定要发送到的电子邮件地址是有效地址，则可以通过确保该地址未列出在您的账户级别黑名单中来覆盖全局黑名单，并且 SES 仍将尝试发送，但如果它被退回，则退回会影响您自己的声誉，而其他人不会收到退回邮件，因为如果他们未使用自己的账户级别黑名单，则无法发送到该电子邮件地址。要了解有关账户级别黑名单详情，请参阅 使用 Amazon SES 账户级黑名单。

积极主动

Internet 上的电子邮件存在的最大问题之一是未经请求的批量电子邮件（垃圾邮件）。电子邮件提供商采取了大量措施来防止客户收到垃圾邮件。Amazon SES 还采取措施来降低电子邮件提供商将您的电子邮件视为垃圾邮件的可能性。Amazon SES 将使用验证、身份验证、发送配额和内容筛选。Amazon SES 还维护对电子邮件提供商的可信声誉，要求您发送高质量电子邮件。Amazon SES 会自动为您执

行其中一些操作（例如内容筛选）；在其他情况下，它会提供工具（例如身份验证）或者为您提供正确的指导（发送配额）。以下部分提供有关每个概念的更多信息。

验证

遗憾的是，垃圾邮件发送者可能会伪造电子邮件标头并仿冒原始电子邮件地址，使电子邮件看起来好像源自另一个来源。为了维护电子邮件提供商与 Amazon SES 之间的信任关系，Amazon SES 需要确保发件人与其声称的身份相符。因此，您需要验证您通过 Amazon SES 发送电子邮件时使用的所有电子邮件地址，以保护您的发送身份。您可以使用 Amazon SES 控制台或者使用 Amazon SES API 来验证电子邮件地址。您也可以验证整个域。有关更多信息，请参阅 创建电子邮件地址身份 和 创建域身份。

如果您的账户仍然处于 Amazon SES 沙盒中，那么您还需要验证除了 Amazon SES 邮箱模拟器提供的地址以外的所有收件人地址。有关移出沙盒的信息，请参阅请求生产访问权限（从 Amazon SES 沙盒中移出）。有关邮箱模拟器的更多信息，请参阅手动使用邮箱模拟器。

身份验证

身份验证 是您可向电子邮件提供商指示您与所声称的身份相符的另一种方法。当您对电子邮件进行身份验证时，您要提供证据，证明您是账户的所有者，且您的电子邮件在传输过程未被篡改。在某些情况下，电子邮件提供商会拒绝转发未经身份验证的电子邮件。Amazon SES 支持两种身份验证方法：发件人策略框架 (SPF) 和 DomainKeys 识别邮件 (DKIM)。有关更多信息，请参阅 在 Amazon SES 中配置身份。

发送配额

如果电子邮件提供商检测到您的电子邮件的数量或发送速率出现突发的意外高峰，则电子邮件提供商可能会怀疑您是垃圾邮件发送者并拦截您的电子邮件。因此，每个 Amazon SES 账户都有一组发送配额。这些配额限制您可以在 24 小时内发送的电子邮件数量，以及您每秒可以发送的电子邮件数量。这些发送配额有助于在电子邮件提供商中维护您的可信度。

大多数情况下，如果您是全新用户，Amazon SES 让您可以每天发送少量的电子邮件。如果您发送的电子邮件对于电子邮件提供商是可接受的，我们会自动增加此配额。随着时间推移，您的发送配额将稳步提高，以便更快地发送更大数量的电子邮件。您还可以创建 SES 提高发送限制案例以请求额外增加配额。

有关发送配额以及如何提高配额的更多信息，请参阅管理您的 Amazon SES 发送限制。

内容筛选

许多电子邮件提供商使用内容筛选来确定传入电子邮件是否为垃圾邮件。内容筛选器查找可疑的内容并拦截符合垃圾邮件特征的电子邮件。Amazon SES 也使用内容筛选器。当您的应用程序向 Amazon

SES 发送请求时，Amazon SES 会代表您汇编电子邮件，然后扫描邮件标头和正文，以确定它们是否包含电子邮件提供商可能视为垃圾邮件的内容。如果您的邮件似乎被 Amazon SES 使用的内容筛选器视为垃圾邮件，则您在 Amazon SES 的声誉将受到负面影响。

Amazon SES 还会扫描所有邮件，检查是否有病毒。如果某个邮件包含病毒，那么 Amazon SES 不会尝试将该邮件发送到收件人的邮件服务器。

信誉

就电子邮件发送而言，声誉至关重要，这是一种表明 IP 地址、电子邮件地址或发送域不是垃圾邮件来源的信心衡量指标。Amazon SES 维护对电子邮件提供商的良好声誉，以便 ISP 将您的电子邮件送达至您的收件人收件箱。同样，您也需要维护对 Amazon SES 的可信声誉。您可以通过发送高质量内容来在 Amazon SES 中建立声誉。当您发送高质量内容时，您的声誉就会随着时间推移变得越来越可信，而且 Amazon SES 会提高您的发送配额。过多的退回邮件和投诉会对您的声誉产生负面影响，并可能导致 Amazon SES 降低您账户的发送配额或终止您的 Amazon SES 账户。

帮助维护您的声誉的一种方法是在您测试系统时使用邮箱模拟器，而不是发送到您自己创建的电子邮件地址。发送到邮箱模拟器的电子邮件不会计入您的退回邮件和投诉指标。有关邮箱模拟器的更多信息，请参阅手动使用邮箱模拟器。

高质量电子邮件

高质量电子邮件是收件人认为有价值并想接收的电子邮件。对不同的收件人而言，有价值的事物各不相同，它们可能是报价、订单确认函、收据、新闻通讯等等。最终，您的送达率取决于您发送的电子邮件的质量，因为电子邮件提供商会拦截他们认为质量较低的电子邮件。

随时了解

无论是您的邮件送达失败、您的收件人对您的电子邮件提出投诉，还是 Amazon SES 将电子邮件成功送达至收件人的邮件服务器，Amazon SES 都会通过提供通知并让您可以轻松监控使用情况统计数据来跟踪问题。

通知

当电子邮件退回时，电子邮件提供商会通知 Amazon SES，而 Amazon SES 会通知您。Amazon SES 会通知您 Amazon SES 将不再重试的查无此人的邮件和软退回邮件。许多电子邮件提供商还会转发投诉，而且 Amazon SES 会与主流电子邮件提供商一起设置投诉反馈循环，因此您不必设置。Amazon SES 可以通过以下两种方式通知您退回邮件、投诉和成功送达：您可以将账户设置为通过 Amazon SNS 接收通知，也可以通过电子邮件接收通知（仅限退回邮件和投诉）。有关更多信息，请参阅 为 Amazon SES 设置事件通知。

使用情况统计数据

Amazon SES 提供使用情况统计数据，以便您可以查看失败的邮件送达以确定并解决根本原因。您可以通过使用 Amazon SES 控制台或者调用 Amazon SES API 查看您的使用情况统计数据。您可以查看您现有的送达邮件数量、退回邮件数量、投诉数量以及感染病毒的被拒绝电子邮件的数量，还可以查看您的发送配额以确保您处于配额范围内。

改进电子邮件发送程序

如果您收到大量退回邮件和投诉，则需要重新评估您的电子邮件发送策略。请记住，过多的退信、投诉和试图发送低质量电子邮件的行为构成滥用行为，并使您 AWS 账户 面临被解雇的风险。最终，您需要确保使用 Amazon SES 发送高质量电子邮件并将电子邮件仅发送到希望接收邮件的收件人。

送t-least-once货

Amazon SES 会在多台服务器上存储邮件的副本，以实现冗余和高可用性。在极少数情况下，当您接收或删除消息时，存储消息副本的某台服务器可能不可用。

如果出现这种情况，则该不可用服务器上的消息副本将不会被删除，并且您在接收消息时可能会再次获得该消息副本。将应用程序设计为幂等 应用程序（多次处理同一消息时，它们不应受到不利影响）。

使用 Amazon SES 发送电子邮件的最佳实践

您管理客户电子邮件通信的方式称作您的电子邮件程序。有多种因素可能会导致您的电子邮件程序成功或失败；这些因素最初看起来混乱而神秘。但是，通过了解电子邮件的传输方式，并按照特定的最佳实践进行操作，您可以提高电子邮件成功到达客户收件箱的几率。

主题

电子邮件程序成功指标

有几种指标可帮助衡量您的电子邮件程序的成功。

本节包含有关以下指标的信息：

- 邮件质量

退信

当电子邮件无法传输给目标收件人时，会出现退信。有两种类型的退信：硬退信和软退信。当电子邮件由于邮件地址不存在等持久性问题而无法送达时，会出现硬退信。当有临时问题阻止电子邮件送达时，会出现软退信。出现软退信的情况有，收件人的收件箱已满，或者收件服务器暂时不可用。Amazon SES 处理软退信的方式是在特定的时间段内尝试重新传输软退回的电子邮件。

务必在您的电子邮件程序中监控硬退信的数量，并从您的收件人列表中删除硬退回的电子邮件地址。当电子邮件接收方检测到较高的硬退回率时，它们将假设您不了解您的收件人。如此一来，硬退回率高可能对电子邮件的送达率造成负面影响。

以下指导原则有助于避免退回和改善您的发件人信誉：

- 尽量保持您的硬退回率低于 5%。电子邮件程序中的硬退件次数越少，您的邮件就越有可能 ISPs 被视为合法和有价值。这个比率应该被视为一个合理且可以实现的目标，但并不是所有 ISPs人的普遍规则。
- 请勿租用或购买电子邮件列表。这些列表可能包含大量无效的地址，这可能会导致硬退回率显著增加。此外，这些列表可能包含垃圾邮件陷阱 - 专门用于捕获非法发件人的电子邮件地址。如果您的邮件落入垃圾邮件陷阱，您的送达率和发件人信誉会不可挽回地受损。
- 持续更新您的邮件列表。如果您很长一段时间未向您的收件人发送电子邮件，请尽可能通过一些其他方式 (如网站登录活动或购买历史记录) 来验证客户的状态。
- 如果您没有办法验证客户的状态，请考虑发送赢回电子邮件。典型的“赢回”电子邮件会提到您很久没有收到客户的消息了，并鼓励客户确认他们仍希望接收您的电子邮件。在发送“赢回”电子邮件后，从您的列表中清除所有未响应的收件人。

当您收到退信时，通过观察以下规则适当地进行回应至关重要：

- 如果某个电子邮件地址硬退信，立即从您的列表中删除该地址。请勿尝试重新发送邮件到硬退信的地址。反复的硬退信会叠加，并最终损害您对收件人 ISP 的声誉。
- 请确保您用于接收退信通知的地址能够接收电子邮件。有关设置退信和投诉通知的更多信息，请参阅为 Amazon SES 设置事件通知。
- 如果您的入站电子邮件来自 ISP 而不是您的内部服务器，大量涌入的退信通知可能会进入您的垃圾邮件文件夹，或被彻底删除。理想情况下，您不应使用托管的电子邮件地址来接收退信。但是，如果必须这样做，您必须经常检查垃圾邮件文件夹，且不可将退信标记为垃圾邮件。在 Amazon SES 中，您可以指定要将退信通知发送到的地址。

- 通常，退信会提供拒收邮件的邮箱地址。但是，如果您需要更精细的数据来将收件人地址与特定的电子邮件营销活动相对应，请包括 X 标头，以便您可以通过它的值追溯到您的内部跟踪系统。有关更多信息，请参阅 Amazon SES 标头字段。

投诉

当电子邮件收件人在基于网页的电子邮件客户端中点击“标记为垃圾邮件”或等效的按钮时，就会出现投诉。如果您累积了大量的此类投诉，ISP 会假定您发送垃圾邮件。这对您的送达率和发件人信誉有负面影响。有些（但不是全部）ISPs 会在举报投诉时通知您；这就是所谓的反馈循环。Amazon SES 会自动将来自提供反馈回路 ISPs 的投诉转发给您。

以下指导原则有助于避免投诉和改善您的发件人信誉：

- 尽量保持您的投诉率低于 0.1%。电子邮件程序中的投诉越少，您的邮件就越有可能 ISPs 被视为合法和有价值。这个比率应该被视为一个合理且可以实现的目标，但并不是所有 ISPs人的普遍规则。
- 如果客户投诉营销电子邮件，您应立即停止向该客户发送营销电子邮件。但是，如果您的电子邮件程序还包括其他类型的电子邮件 (如通知或事务性电子邮件)，继续向发出投诉的收件人发送这些类型的邮件也许是可以接受的。
- 与硬退回一样，如果您很长时间未向某个列表发送电子邮件了，请确保您的收件人了解他们为什么收到您的邮件。建议您发送“欢迎”邮件，提醒他们您的身份以及您为何联系他们。

当您收到投诉时，通过观察以下规则适当地进行回应至关重要：

- 请确保您用于接收投诉通知的地址能够接收电子邮件。有关设置退信和投诉通知的更多信息，请参阅为 Amazon SES 设置事件通知。
- 确保您的投诉通知不会被您的 ISP 或邮件系统标记为垃圾邮件。
- 投诉通知通常包含电子邮件正文；这不同于退回通知，后者只包括电子邮件标头。但是，在投诉通知中，发起投诉的个人的电子邮件地址会被删除。使用自定义 X 标头或在电子邮件正文中嵌入特殊标识符，以便您能区别发起投诉的电子邮件地址。此方法可让您更轻松地识别投诉的地址，以便您可以从您的收件人列表中删除这些地址。

邮件质量

电子邮件接收方使用内容筛选器来检测邮件中的特定属性，以识别您的邮件是否合法。这些内容筛选器会自动审核邮件的内容，以识别常见的不受欢迎的邮件特征乃至恶意邮件。Amazon SES 使用内容筛选技术，协助在邮件发送之前检测和拦截包含恶意软件的邮件。

如果您的电子邮件接收方的内容筛选器认定您的邮件包含垃圾邮件或恶意电子邮件的特性，您的邮件很可能被标记出来，并从收件人的收件箱中移出。

在设计电子邮件时请记住以下几点：

- 现代内容筛选器非常智能，且不断调整和更改。它们不依赖于预定义的规则集。第三方服务（例如ReturnPath或 Litmus）可以帮助识别电子邮件中可能触发内容过滤器的内容。
- 如果您的电子邮件包含链接，请查看基于 DNS URLs 的黑洞名单 (DNSBLs) 中的链接，例如在 Uribl.com 和 S ur bl.org 上找到的链接。
- 避免使用短地址。恶意发件人可能使用短地址来隐藏链接的实际目标。当 ISPs 注意到链接缩短服务（即使是最负盛名的服务）被用于邪恶目的的时，他们可能会完全拒绝访问这些服务。如果您的电子邮件中包含被加入拒绝列表的链接缩短服务链接，它将不会送达客户的收件箱，而您的电子邮件营销活动的成功率也随之受到影响。
- 测试电子邮件中的每个链接，确保其指向预期的页面。
- 请确保您的网站包括隐私策略和使用条款文档，并且这些文档是最新的。最好在您发送的每封电子邮件中添加到这些文档的链接。提供指向这些文档的链接表明您对客户无所隐瞒，有助于建立信任关系。
- 如果您计划发送高频率内容 (如“每日交易”邮件)，请确保您的电子邮件内容每次都有所不同。在发送高频邮件时，您必须确保这些消息及时且有意义，而不是重复和令人厌烦。

保持良好的发件人声誉

在 Amazon SES 中，发件人信誉是指电子邮件提供商和垃圾邮件筛选器所感知的电子邮件发件人的可信度和可靠性。它可以衡量您的电子邮件被视为合法并成功传送到收件人收件箱的可能性。

以下各节介绍了您必须注意的核心电子邮件发送主体，以确保您的电子邮件通信能够到达目标受众，同时保持良好的发件人声誉。

域和“发件人”地址注意事项

- 请仔细考虑您用于发送电子邮件的地址。“发件人”地址是收件人看到的第一条信息，因此可能会留下持久的第一印象。此外，有些人会 ISPs 将您的声誉与您的 “发件人” 地址联系起来。
- 请考虑为不同类型的通信使用不同的子域。例如，假定您从 example.com 域发送电子邮件，并且您打算同时发送营销邮件和事务性邮件。不要从 example.com 发送所有邮件，而是从像 marketing.example.com 这样的子域发送营销邮件，从像 orders.example.com 这样的子域发送事务性邮件。独有的子域可建立它们自己的信誉。使用子域可降低信誉损坏的风险，例如，当您的营销通信落入垃圾邮件陷阱或触发内容筛选器时。

- 如果您计划发送大量邮件，请不要从 sender@hotmail.com 这样的基于 ISP 的地址发送这些邮件。如果 ISP 注意到有大量邮件来自 sender@hotmail.com，该电子邮件的处理方式将与来自您拥有的出站电子邮件发送域的邮件有所不同。
- 请与您的域注册商沟通，确保您的域的 WHOIS 信息准确无误。保持诚实的 up-to-date WHOIS记录表明您重视透明度，并允许用户快速识别您的域名是否合法。
- 避免使用 no-reply 地址，例如 no-reply@example.com，作为您的“From”地址或“Reply-to”地址。使用 no-reply@ 电子邮件地址向您的收件人明确传达一个信息：您没有为他们提供与您联系的方式，并且您对他们的反馈不感兴趣。

身份验证

- 使用 SPF 和 SenderID 对您的域进行身份验证。这些身份验证方法向您的电子邮件收件人确认，您发送的每封电子邮件都来自实际声明的发件域。
- 使用 DKIM 对您的出站邮件签名。此步骤向收件人确认，在发件人与接收方之间沟通中，内容不曾被更改。
- 您可以测试您针对 SPF 和 DKIM 的身份验证设置，方法是向您拥有的基于 ISP 的电子邮件地址发送一封电子邮件，例如个人 Gmail 或 Hotmail 账户，然后查看邮件的标头。该标头将指出您进行身份验证和签署邮件的尝试是否已成功。

构建和维护列表

- 实施双重确认策略。当用户注册接收您的电子邮件时，向其发送一封包含确认链接的邮件，直到他们单击该链接确认地址之前，不要开始发送其他电子邮件。双重确认策略有助于降低因拼写错误而导致的硬退信数量。
- 在使用基于 Web 的表单收集电子邮件地址时，请在提交之前对这些地址执行最低限度的检验。例如，确保您收集的地址格式正确 (即地址格式为 recipient@example.com)，并且它们指向的域具有有效的 MX 记录。
- 请谨慎允许将用户定义的输入未经检查即传递给 Amazon SES。论坛注册和表单提交存在独特的风险，因为该内容完全是用户生成的，垃圾邮件发送者可以用其自己的内容来填写表单。您有责任确保只发送高质量的电子邮件内容。
- 标准的别名 (例如 postmaster@、abuse@ 或 noc@) 不太可能故意用于注册您的电子邮件。确保您仅将邮件发送给确实想要接收它们的真实用户。对于标准别名应特别注意这条规则，因为这些别名习惯上是为电子邮件监控程序保留的。这些别名可能是恶意添加到您的列表中，以蓄意破坏的形式损害您的声誉。

合规

- 请注意电子邮件收件方所在国家/地区的相关电子邮件营销和反垃圾邮件法律和法规。您有责任确保您发送的电子邮件遵守这些法律。本指南中未涵盖这些法律，因此您需要研究它们。有关法律的列表，请参阅 Wikipedia 中的按国家/地区统计的垃圾邮件法律。
- 务必咨询律师来获取法律建议。

将 Amazon SES 与 AWS 软件开发工具包配合使用

AWS 软件开发套件 (SDKs) 可用于许多流行的编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

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

SDK 文档	代码示例
AWS SDK for Swift	AWS SDK for Swift 代码示例

有关特定于 Amazon SES 的示例，请参阅 使用 Amazon SES 的代码示例 AWS SDKs。

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

开始使用 Amazon Simple Email Service

本章将指导您完成初始设置 Amazon SES 所需的任务，以及帮助您开始使用的教程。

主题

设置 Amazon Simple Email Service

开始使用 Amazon SES 前，必须完成以下任务。

任务

报名参加 AWS

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话或收到短信，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

设置您的 SES 账户

通过验证电子邮件地址和发送域来开始使用 SES，这样，您就可以开始通过 SES 发送电子邮件，并使用 SES 账户设置向导为您的账户请求生产访问权限。

使用 SES 账户设置向导设置您的账户

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 从 SES 控制台主页选择开始使用，向导将引导您完成设置 SES 账户的步骤。

仅当您尚未在 SES 中创建任何身份（电子邮件地址或域）时，才会显示 SES 账户设置向导。

授予编程访问权限（在控制台之外与 SES 交互）

如果用户想在 AWS 外部进行交互，则需要编程访问权限 AWS Management Console。授予编程访问权限的方式取决于正在访问的用户类型 AWS。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时证书签署向 AWS CLI AWS SDKs、或发出的编程请求 AWS APIs。	按照您希望使用的界面的说明进行操作。 • 有关的 AWS CLI，请参阅《AWS Command Line Interface 用户指南》 AWS IAM Identity Center中的 “配置 AWS CLI 要使用”。 • 有关工具和 AWS SDKs AWS APIs，请参阅《工具参考指南》中的 IAM 身份中心身份验证AWSSDKs 和工具参考指南。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用临时证书签署向 AWS CLI AWS SDKs、或发出的编程请求 AWS APIs。	按照 IAM 用户指南中的将临时证书与 AWS 资源配合使用中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI AWS SDKs、或发出的编程请求 AWS APIs。	按照您希望使用的界面的说明进行操作。 • 有关信息 AWS CLI，请参阅用户指南中的使用 IAM 用户证书进行身份验证。AWS Command Line Interface • 有关 AWS SDKs 和工具，请参阅《工具参考指南》AWS SDKs 和《工具参考指南》中的使用长期凭证进行身份验证。 • 有关信息 AWS APIs，请参阅 IAM 用户指南中的管理 IAM 用户的访问密钥。

下载 AWS 软件开发工具包（用于使用 SES APIs）

要调用 SES APIs 而不必处理诸如组装原始 HTTP 请求之类的低级细节，可以使用 S AWS DK。它们 AWS SDKs 提供了封装 SES 和其他 AWS 服务功能的函数和数据类型。要下载 S AWS DK，请转至SDKs。下载 SDK 后，创建共享凭据文件并指定您的 AWS 访问密钥。

从另一个电子邮件发送解决方案迁移到 Amazon SES

本主题概述了如果您想将电子邮件发送解决方案从本地托管的解决方案迁移到 Amazon SES，或者从托管在 Amazon 实例上的解决方案迁移到 Ama EC2 zon SES 时必须采取的步骤。

本节中的主题：

- 第 1 步：验证您的域

第 1 步：验证您的域

您必须先验证您发送电子邮件时计划使用的身份，然后才能使用 Amazon SES 发送电子邮件。在 Amazon SES 中，身份可以是电子邮件地址或整个域。在验证域时，您可以使用 Amazon SES 从该域上的任何地址发送电子邮件。有关验证域的更多信息，请参阅创建域身份。

第 2 步：请求生产环境访问权限

在首次开始使用 Amazon SES 时，您的账户位于沙盒环境中。当您的账户位于沙盒中时，您只能将电子邮件发送到您已验证的地址。此外，每天可以发送的邮件数量和每秒可以发送的邮件数量受到限制。有关请求生产环境访问权限的更多信息，请参阅请求生产访问权限（从 Amazon SES 沙盒中移出）。

第 3 步：配置域身份验证系统

可以将域配置为使用身份验证系统（例如 DKIM 和 SPF）。从技术上说，此步骤是可选的。但是，通过为您的域设置 DKIM 和/或 SPF，可以提高电子邮件的送达率，并提高客户对您的信任度。有关设置 SPF 的更多信息，请参阅在 Amazon SES 中使用 SPF 对电子邮件进行身份验证。有关设置 DKIM 的更多信息，请参阅在 Amazon SES 中使用 DKIM 对电子邮件进行身份验证。

第 4 步：生成 SMTP 凭证

如果您计划借助使用 SMTP 的应用程序发送电子邮件，则必须生成 SMTP 凭证。SMTP 凭证不同于您的常规 AWS 凭证。这些证书在每个 AWS 地区也是唯一的。有关生成 SMTP 凭证的更多信息，请参阅获取 Amazon SES SMTP 凭证。

第 5 步。连接到 SMTP 端点

如果您使用邮件传输代理（例如 postfix 或 sendmail），则必须更新该应用程序的配置才能引用 Amazon SES SMTP 端点。有关 SMTP 端点的完整列表，请参阅连接到 Amazon SES SMTP 端点。请注意，您在上一步中创建的 SMTP 凭证与特定 AWS 区域相关联。您必须连接到您在其中创建了 SMTP 凭证的区域中的 SMTP 端点。

后续步骤

此时，您可以开始使用 Amazon SES 发送电子邮件。不过，您可以执行一些可选步骤。

- 您可以创建配置集，它们是应用于您发送的电子邮件的规则集。例如，您可以使用配置集来指定在以下情况下发送通知的位置：发送电子邮件时、收件人打开邮件或单击邮件中的链接时、电子邮件退回时以及收件人将您的电子邮件标记为垃圾邮件时。有关更多信息，请参阅 在 SES 中使用配置集。
- 当您通过 Amazon SES 发送电子邮件时，请务必监控您的账户的退回邮件和投诉。Amazon SES 包含一个声誉指标控制台页面，用于跟踪您的账户的退回邮件和投诉。有关更多信息，请参阅 使用声誉指标跟踪退回邮件率和投诉率。您还可以创建 CloudWatch警报，在这些费率过高时提醒您。有关创建 CloudWatch 警报的更多信息，请参阅使用创建信誉监控警报 CloudWatch。
- 对于发送大量电子邮件的客户或只希望全权控制其 IP 地址的声誉的客户，他们可以租用专用 IP 地址，但每月需额外付费。有关更多信息，请参阅 Amazon SES 专用 IP 地址。

请求生产访问权限（从 Amazon SES 沙盒中移出）

为了帮助防止出现欺诈和滥用以及保护您作为发件人的声誉，我们将某些限制应用于新的 Amazon SES 账户。

我们将所有新账户放在 Amazon SES 沙盒中。每个 AWS 区域账户的沙盒状态都是唯一的。在您的账户位于沙盒时，您可以使用 Amazon SES 的所有特征。不过，在您的账户位于沙盒时，我们将以下限制应用于您的账户：

- 您只能将邮件发送到已验证的电子邮件地址和域，或者发送到 Amazon SES 邮箱模拟器。
- 您每 24 小时最多可以发送 200 封邮件。
- 您每秒最多可以发送 1 封邮件。
- 对于发送身份验证，您或委托发件人均不能向未经验证的电子邮件地址发送电子邮件。
- 对于账户级抑制，将禁用与抑制列表管理相关的批量操作和 Amazon SES API 调用。

当您的账户从沙盒迁移到生产环境时，您可以将电子邮件发送给任何收件人，而不管收件人的地址或域是否经过验证。但是，您仍须验证用作“发件人”、“源”、“发送者”或“退回路径”地址的所有身份。

完成本节中的过程，以请求将您的账户从沙盒中移除并置于生产环境中。

Tip

- 如果您是新客户并且尚未创建任何身份，则控制台中的 SES 帐户设置向导将启用，以帮助您入门。有关如何访问向导的说明，请参阅设置 SES 帐户。
- 如果您已创建了一个或多个身份，则会看到获取设置页面，而不是账户设置向导。
  - 如果您的身份之一是经过验证的域，您还可以直接从获取设置页面请求生产访问权限。这是因为，在请求生产访问权限之前向 SES 验证您的域是一种最佳实践，有助于更快地获得生产访问权限的批准，使您可以立即开始发送电子邮件。

Note

- 如果您使用 Amazon SES 从亚马逊 EC2 实例发送电子邮件，则可能还需要请求从亚马逊 EC2实例的端口 25 上移除限制。有关更多信息，请参阅如何从我的 EC2 实例中移除端口 25 上的限制？ 在 AWS 知识中心中。

使用 AWS Management Console请求生产访问权限（从沙盒中移除您的账户）

1. 打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航面板中，选择账户控制面板。
3. 在控制台顶部的警告框中（显示“Your Amazon SES account is in the sandbox”），选择右侧的查看获取设置页面，然后选择请求生产访问。
4. 在账户详细信息模式中，选择 Marketing（营销）或 Transactional（事务）中最符合您要发送的大多数邮件的单选按钮。
   - 营销电子邮件-发送给目标潜在客户或客户名单，其中包含营销和促销内容，例如购买、下载信息等。 one-to-many
   - 交易电子邮件-以每个收件人唯一 one-to-one的方式发送，通常由用户操作（例如网站购买、密码重置请求等）触发。
5. 在 Website URL（网站 URL）中，输入您网站的 URL，以帮助我们更好地了解您计划发送的内容类型。
6. 在 Additional contacts (其他联系人)中，请说明您希望在哪里接受关于您的账户的通信。这可以是逗号分隔列表，包含最多 4 个电子邮件地址。

7. 在 Preferred contact language (首选联系语言)中，选择您希望用 English (英语) 还是 Japanese (日语) 接收通信。
8. 在确认中，勾选您同意只向明确要求发送电子邮件的个人发送电子邮件的复选框，并确认您已制定了流程处理退信和投诉通知。
9. 选择提交请求按钮，此时将显示一条横幅，确认您的请求已提交并且当前正在审核中。

在提交您的账户详细信息以供审核之后，您将无法在审核完成之前编辑您的详细信息。 AWS 支持 团队会在 24 小时内对您的请求做出初步回应。

为了防止我们的系统被用于发送未经请求或恶意的内容，我们必须仔细审查每个请求。如果我们能做到这一点，我们将在 24 小时内准予您的请求。但是，如果我们需要从您那里获得其他信息，则可能需要更长的时间来解决您的请求。

或者，您也可以使用提交生产访问权限的请求 AWS CLI。当您想要请求大量身份的 AWS CLI 生产访问权限时，或者想要自动设置 Amazon SES 时，使用提交请求会很有帮助。

请求使用 AWS CLI从 Amazon SES 沙盒中删除您的账户

1. 先决条件：必须安装和配置 AWS CLI。有关更多信息，请参阅 AWS Command Line Interface《用户指南》。
2. 在命令行输入以下命令：

```
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"
```

在上述命令中，执行以下操作：

a. *TRANSACTIONAL*替换为您计划通过 Amazon SES 发送的电子邮件类型。您可指定 TRANSACTIONAL 或 MARKETING。如果有多个值适用，请指定适用于您计划发送的大部分邮件的选项。

b. *https://example.com*替换为您网站的网址。提供该信息将帮助我们更好地了解您打算发送的内容类型。

c. *info@example.com*替换为您想要接收有关您的账户的通信的电子邮件地址。这可以是逗号分隔列表，包含最多 4 个电子邮件地址。

d. *EN*用您的首选语言替换。您可以指定 EN 待代表英语，或置顶 JA 代表日语。

e. *Use-case description*替换为对您的用例的描述。

在提交您的账户详细信息以供审核之后，您将无法在审核完成之前编辑您的详细信息。 AWS 支持 团队会在 24 小时内对您的请求做出初步回应。

为了防止我们的系统被用于发送未经请求或恶意的内容，我们必须仔细审查每个请求。如果我们能做到这一点，我们将在 24 小时内准予您的请求。但是，如果我们需要从您那里获得其他信息，则可能需要更长的时间来解决您的请求。

管理您的 Amazon SES 发送限制

Amazon SES 账户有一组发送配额，用于限制可发送的电子邮件的数量和发送邮件的速率。发送配额对所有 Amazon SES 客户都有益处，因为它们有助于在 Amazon SES 和电子邮件提供商之间维护可信的关系。发送配额将帮助您逐步增加发送活动，并降低电子邮件提供商由于您的电子邮件发送量或速率突然意外猛增而阻止您的电子邮件的可能性。

以下配额适用于通过 Amazon SES 发送电子邮件：

- 发送配额—您在 24 小时之内可发送的最大电子邮件数量。此配额是根据滚动时间段计算的。每当您尝试发送电子邮件时，Amazon SES 会确定您在过去 24 小时内发送的电子邮件数量。只要您在过去 24 小时内发送的电子邮件总数低于您的配额，发送请求就会被接受，并发送您的电子邮件。

  如果发送邮件会超过您账户的每日最大值，则系统将拒绝您对 Amazon SES 的调用。
- 发送速率 – Amazon SES 每秒可以承受的来自您账户的最大电子邮件数量。您可能因为短时间突增而暂时超过此配额，但不会在一段时间内持续。

  > **Note**
  >
  > Amazon SES 接受您的邮件的速率可能低于您账户的最大发送速率。

- 最大邮件大小 (MB)—您可以发送的最大电子邮件大小。这其中包括 MIME 编码后作为电子邮件一部分的任何图像和附件。例如，如果附上 5MB 文件，则 MIME 编码后电子邮件中的附件大小约为 6.85MB（约为原始文件大小的 137%）。

  > **Note**
  >
  > 我们建议您将附件上载到云盘，并包含云驱动器附件的 URL，以减少电子邮件大小并提高送达率。SES 不能保证大型电子邮件最终能够出现在收件人邮箱中，因为不同的邮件服务器具有不同的基于大小的策略。

您的 Amazon SES 发送配额在每个 AWS 地区都是独立的。有关在多个 AWS 区域使用 Amazon SES 的信息，请参阅区域和 Amazon SES。

当您的账户位于 Amazon SES 沙盒中时，您每 24 小时周期只能发送 200 封邮件，最大发送速率为每秒发送一封邮件。在提交从沙盒中删除账户的请求时，您还可以同时请求增加配额。有关使您的账户脱离沙盒的更多信息，请参阅请求生产访问权限（从 Amazon SES 沙盒中移出）。

当您的账户已从沙箱中移除后，您可以随时在 Support Center 中创建新案例，请求增加配额。 AWS 有关更多信息，请参阅 提升您的 Amazon SES 发送配额。

> **Note**
>
> 发送配额基于是收件人数而不是电子邮件数。例如，一封包含 10 个收件人的电子邮件占用 10 份配额。但是，我们不建议您在单次调用 `SendEmail` API 操作时向多个收件人发送电子邮件，因为如果调用失败，将拒绝所有电子邮件。我们建议您为每个收件人调用一次 `SendEmail`。

- 要增加您的发送配额，请参阅提升您的 Amazon SES 发送配额。
- 要使用 Amazon SES 控制台或 Amazon SES API 来监控您的发送配额，请参阅监控您的 Amazon SES 发送配额。
- 有关应用程序在您达到发送配额时收到的错误的信息，请参阅与您 Amazon SES 账户的发送配额相关的错误。

提升您的 Amazon SES 发送配额

根据您当前所在的区域，您的账户具有以下可提升的配额。

资源	默认配额	描述
发送配额	200	此账户在当前 AWS 区域中，您在 24 小时周期内可以发送的电子邮件数量上限。
发送速率	1	此账户在当前 AWS 区域中，Amazon SES 每秒可以接受的电子邮件数量上限。

自动提升发送配额

当您的账户不在沙盒中并且您在发送高质量的生产电子邮件时，我们可能会自动提高您账户的发送配额。通常，我们会在您需要实际提高这些配额之前，自动增加这些配额。

要想获得自动提高速率的资格，必须满足以下所有陈述：

- 发送收件人愿意接收的高质量内容 – 发送收件人需要和希望收到的内容。停止向未打开您电子邮件的客户发送电子邮件。
- 发送真实的生产内容 – 发送测试邮件到虚假电子邮件地址会对您的退回率和投诉率产生负面影响。此外，仅向内部收件人发送邮件会难于确定客户是否希望接收您发送的内容。不过，当您向非内部收件人发送生产邮件时，我们可以准确地评估您的电子邮件发送行为。
- 发送量接近您当前的配额 – 要想获得自动提高配额的资格，您的日电子邮件量应该经常接近但不超过账户的每天最大配额。
- 具有较低的邮件退回率和投诉率 - 尽可能减少您收到的退回邮件和投诉的数量。收到大量退回邮件和投诉可能对您的发送配额有负面影响。

用户请求提升发送配额

如果您当前的发送限额不足以满足需求，并且我们没有自动为您增加限额，您可以申请提高限额：

- 发送限额或发送速率 – 可以通过 AWS 服务限额控制台提交对其中任一项的增加请求。

使用 Service Quotas 控制台请求增加 Amazon SES 发送配额。

1. 打开服务配额控制台。
2. 使用控制台右上角的下拉菜单（您的账号旁边），选择您希望增加的区域。
3. 在导航窗格中，选择 AWS services（AWS 服务）。
4. 选择 Amazon Simple Email Service (SES)。
5. 选择配额，然后按照说明请求增加配额。

> AWS 支持 增加请求类型的团队 SLA
>
> 为了防止我们的系统被用于发送未经请求或恶意的内容，我们必须仔细审查每个请求。如果我们能做到这一点，我们将在下面列出的指定时间内准予您的请求，用于请求的增加类型。但是，如果我们需要从您那里获得其他信息，则可能需要更长的时间来解决您的请求。如果您的使用案例与我们的策略不符，我们可能无法准予您的请求。
>
> - 发送配额或发送率：长达 24 小时。

> **Note**
>
> 虽然 Service Quotas 控制台提供多种不同的语言，但实际支持仅以英语提供。

监控您的 Amazon SES 发送配额

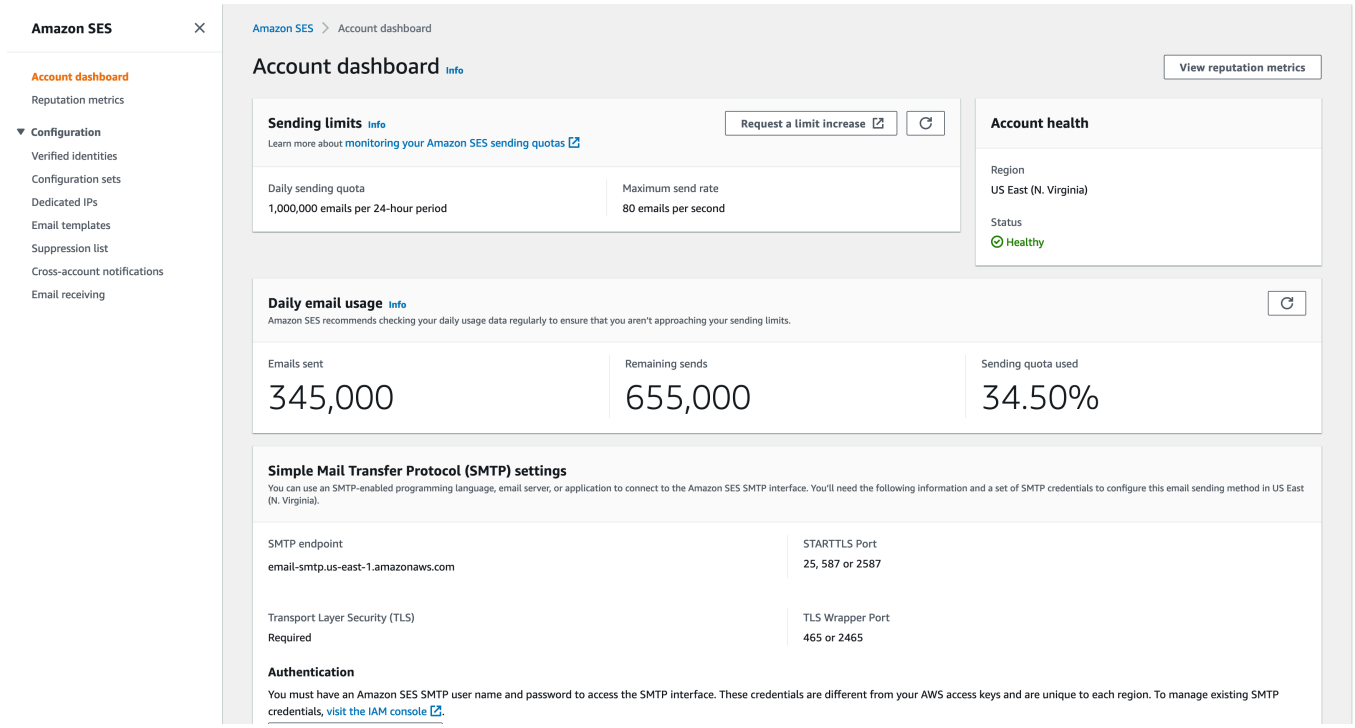
您可以使用 Amazon SES 控制台来监控您的发送配额，也可以使用 Amazon SES API，通过调用查询（HTTPS）接口来直接监控，或者通过 AWS SDK、AWS Command Line Interface 或 AWS Tools for Windows PowerShell 间接监控。

> **Important**
>
> 我们建议您经常检查发送统计数据，以确保不会接近发送配额。如果您快要达到发送配额，请参阅提升您的 Amazon SES 发送配额了解如何提高发送配额的信息。不要等到达到发送配额时再考虑提高。

使用 Amazon SES 控制台来监控您的发送配额

以下过程介绍如何使用 Amazon SES 控制台来查看您的发送配额。

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中，选择 Account dashboard (账户控制面板)。您的发送配额将显示在发送限制下。已发送的电子邮件总数、剩余发送数和已使用的发送限额百分比显示在 Daily email usage（每日电子邮件使用情况）下。



3. 要更新显示，请选择每日电子邮件使用情况框右上角的刷新图标。

使用 Amazon SES API 来监控您的发送配额

Amazon SES API 提供 `GetSendQuota` 操作，该操作返回您的发送配额。调用 `GetSendQuota` 操作时，您会收到以下信息：

- 您在过去 24 小时内发送的电子邮件数量
- 当前 24 小时时段内的发送配额
- 最大发送速率

> **Note**
>
> 有关 `GetSendQuota` 的描述，请参阅 Amazon Simple Email Service API 参考。

与您 Amazon SES 账户的发送配额相关的错误

如果您在达到每日发送配额（24 小时周期内可以发送的电子邮件最大数）或者最大发送速率（每秒可以发送的电子邮件最大数）之后尝试发送电子邮件，Amazon SES 将丢弃电子邮件，不尝试发

送。Amazon SES 还会提供说明问题的错误消息。Amazon SES 生成此错误消息的方式取决于您尝试发送电子邮件的方法。此主题包括您通过 Amazon SES API 和通过 SMTP 接口收到的邮件的信息。

对于在达到最大发送速率时可以使用的方法，请参阅 AWS 消息收发和目标博客中的 How to handle a "Throttling – Maximum sending rate exceeded" error。

达到 Amazon SES API 的发送限制

如果您尝试使用 Amazon SES API（或 AWS 软件开发工具包）发送电子邮件，但已经超过了账户的发送限制，则 API 会产生`ThrottlingException`错误。该错误消息包括以下消息之一：

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

如果遇到限制错误，您应让应用程序等待最多 10 分钟的间隔，然后再重新尝试发送请求。

达到 SMTP 的发送限制

如果您尝试使用 Amazon SES SMTP 接口发送电子邮件，但您已超出账户的发送限额，那么您的 SMTP 客户端可能会显示以下错误之一：

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

不同 SMTP 客户端处理这些错误的方法各不相同。

使用 Amazon SES 设置电子邮件发送

您可以使用 Amazon SES 控制台、Amazon SES 简单邮件传输协议（SMTP）接口或 Amazon SES API，通过 Amazon Simple Email Service（Amazon SES）发送电子邮件。您通常使用控制台发送测试电子邮件和管理发送活动。要发送批量电子邮件，您使用 SMTP 接口或 API。有关 Amazon SES 电子邮件定价的信息，请参阅 SES 定价。

- 如果您要使用支持 SMTP 的软件包、应用程序或编程语言通过 Amazon SES 发送电子邮件，或者将 SES 与您的现有邮件服务器集成，请使用 Amazon SES SMTP 接口。有关更多信息，请参阅 通过 Amazon SES SMTP 接口以编程方式来发送电子邮件。
- 如果您要使用原始 HTTP 请求来调用 Amazon SES，请使用 Amazon SES API。有关更多信息，请参阅 使用 Amazon SES API 发送电子邮件。

> **Important**
>
> 如果您向多个收件人（收件人包括“收件人”、“抄送”和“密件抄送”地址）发送一封电子邮件，而对 Amazon SES 的调用失败，那么整个电子邮件将被拒绝，并且任何收件人都不会收到预期的电子邮件。因此，我们建议您一次向一个收件人发送一封电子邮件。

使用 Amazon SES SMTP 接口发送电子邮件

要通过 Amazon SES 发送生产电子邮件，您可以使用简单邮件传输协议（SMTP）接口或 Amazon SES API。有关 Amazon SES API 的更多信息，请参阅使用 Amazon SES API 发送电子邮件。此部分介绍了 SMTP 接口。

Amazon SES 使用 SMTP（Internet 上最常见的电子邮件协议）发送电子邮件。您可以使用各种支持 SMTP 的编程语言和软件连接到 Amazon SES SMTP 接口，来通过 Amazon SES 发送电子邮件。本节介绍如何获取 Amazon SES SMTP 凭证、如何使用 SMTP 接口发送电子邮件以及如何将多个软件和邮件服务器配置为使用 SES 发送电子邮件。

有关您在通过 Amazon SES 的 SMTP 接口使用 SES 可能遇到的常见问题的解决方法，请参阅Amazon SES SMTP 问题。

通过 SMTP 发送电子邮件的要求

要使用 Amazon SES SMTP 接口发送电子邮件，您需要以下内容：

- SMTP 端点地址。有关 Amazon SES SMTP 端点的列表，请参阅连接到 Amazon SES SMTP 端点。
- SMTP 接口端口号。端口号因连接方法而异。有关更多信息，请参阅 连接到 Amazon SES SMTP 端点。
- SMTP 用户名和密码。SMTP 凭证对每个 AWS 区域唯一。如果计划使用 SMTP 接口在多个 AWS 区域中发送电子邮件，您需获取各个区域的 SMTP 凭证。

> **Important**
>
> 您的 SMTP 凭证与您的 AWS 访问密钥或用于登录 Amazon SES 控制台的凭证不同。有关如何生成 SMTP 凭证的信息，请参阅获取 Amazon SES SMTP 凭证。

- 可使用传输层安全性 (TLS) 进行通信的客户端软件。有关更多信息，请参阅 连接到 Amazon SES SMTP 端点。
- 您已使用 Amazon SES 验证的电子邮件地址。有关更多信息，请参阅 Amazon SES 中已验证的身份。
- 如果您想发送大量电子邮件，请增加发送配额。有关更多信息，请参阅 管理您的 Amazon SES 发送限制。

通过 SMTP 发送电子邮件的方法

您可以通过以下任一方法通过 SMTP 发送电子邮件：

- 要将支持 SMTP 的软件配置为通过 Amazon SES SMTP 接口发送电子邮件，请参阅使用软件包通过 Amazon SES 发送电子邮件。
- 要将应用程序编程为通过 Amazon SES 发送电子邮件，请参阅通过 Amazon SES SMTP 接口以编程方式来发送电子邮件。
- 要将现有电子邮件服务器配置为通过 Amazon SES 发送所有传出邮件，请参阅将 Amazon SES 与您的现有电子邮件服务器集成。
- 要使用命令行与 Amazon SES SMTP 接口进行交互（这对测试很有用），请参阅使用命令行来测试与 Amazon SES SMTP 接口的连接。

有关 SMTP 响应代码的列表，请参阅由 Amazon SES 返回的 SMTP 响应代码。

要提供的电子邮件信息

当您通过 SMTP 接口访问 Amazon SES 时，您的 SMTP 客户端应用程序会汇编此邮件，因此您需要提供的信息取决于您所使用的应用程序。客户端与服务器之间的 SMTP 交换至少需要以下信息：

- 源地址
- 目标地址
- 邮件数据

如果您使用的是 SMTP 接口，并且已启用反馈转发功能，则您的退信、投诉和送达通知将发送到“MAIL FROM”地址。不会使用您指定的任何“Reply-To”地址。

获取 Amazon SES SMTP 凭证

您需要 Amazon SES SMTP 凭证才能访问 SES SMTP 接口。

您用于通过 SES SMTP 接口发送电子邮件的凭证在每个 AWS 区域都是唯一的。如果您使用 SES SMTP 接口在多个区域中发送电子邮件，那么您必须为计划使用的每个区域生成一组 SMTP 凭证。

您的 SMTP 密码与您的 AWS 私有访问密钥不同。有关凭证的更多信息，请参阅Amazon SES 凭证的类型。

> **Note**
>
> 有关当前可用的 SMTP 端点的列表，请参阅中的 SMTP 终端节点。AWS 一般参考

使用 SES 控制台获取 SES SMTP 凭证

要求

IAM 用户可以创建 SES SMTP 凭证，但该用户的策略必须授予这些凭证使用 IAM 自身的权限，因为 SES SMTP 凭证是通过使用 IAM 创建的。您的 IAM 策略必须允许您执行以下 IAM 操作：`iam:ListUsers`、`iam:CreateUser`、`iam:CreateAccessKey` 和 `iam:PutUserPolicy`。如果您尝试使用控制台创建 SES SMTP 证书，但您的 IAM 用户没有这些权限，则会看到一条错误消息，指出您的账户 “无权执行 iam:” ListUsers。

> **Important**
>
> 上面提到的 IAM 操作具有权限管理访问级别，这是最高 IAM 级别，因为它授予在服务中授予或修改资源权限的权限。因此，为了提高 AWS 账户的安全性，强烈建议您限制或定期监控这些包含权限管理访问级别分类的策略。

创建 SMTP 凭证

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧的导航窗格中选择 SMTP settings（SMTP 设置）- 这将打开 Simple Mail Transfer Protocol (SMTP) settings [简单邮件传输协议（SMTP）设置] 页面。
3. 选择右上角的 Create SMTP Credentials（创建 SMTP 凭证）- IAM 控制台将打开。
4. （可选）如果您需要查看、编辑或删除已创建的 SMTP 用户，请选择右下角的 Manage my existing SMTP credentials（管理我现有的 SMTP 凭证）-IAM 控制台将打开。按照以下步骤给出了用于管理 SMTP 凭证的详细信息。
5. 在为 SMTP 创建用户中，在 IAM 用户名字段中键入一个名称。或者，您可以使用此字段中提供的默认值。完成后，选择右下角的创建用户。
6. 选择 SMTP 密码下的显示 - 您的 SMTP 凭证显示在屏幕上。
7. 通过选择下载 .csv 文件以下载这些凭证，或将其复制并存储在安全的位置，因为您在关闭此对话框之后无法查看或保存凭证。
8. 选择返回 SES 控制台。

您可以在 IAM 控制台的 Access management（访问管理）下查看使用此过程创建的 SMTP 凭证列表，并可以选择 Users（用户），然后使用搜索栏查找已分配 SMTP 凭证的所有用户。

您也可以使用 IAM 控制台来删除现有 SMTP 用户。要了解有关删除用户的更多信息，请参阅《IAM 入门指南》中的管理 IAM 用户。

如果您要更改 SMTP 密码，请在 IAM 控制台中删除现有 SMTP 用户。然后，要生成一组新 SMTP 凭证，请完成前面的过程。

通过转换现有 AWS 凭证获取 SES SMTP 凭证

如果您有使用 IAM 接口设置的用户，则可以从其证书中派生该用户的 SES SMTP AWS 证书。

> **Important**
>
> 不要使用临时 AWS 证书来获取 SMTP 证书。SES SMTP 接口不支持从临时安全凭证生成的 SMTP 凭证。

使 IAM 用户能够使用 SES SMTP 接口发送电子邮件

1. 按照以下步骤，使用本节中提供的算法，从用户 AWS 凭证中派生出用户的 SMTP 凭据。

   由于您是从 AWS 凭据开始的，因此 SMTP 用户名与 AWS 访问密钥 ID 相同，因此您只需要生成 SMTP 密码即可。
2. 登录 AWS Management Console 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
3. 在访问管理下，选择策略，然后选择创建策略。
4. 在策略编辑器中，选择 JSON，然后删除编辑器中的所有示例代码。
5. 将下面的策略粘贴到编辑器中。

   JSON

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

6. 选择下一步，然后在策略名称字段中输入 AmazonSesSendingAccess，接着选择创建策略。
7. 在访问管理下，选择用户组，然后选择创建组。
8. 在用户组名称字段中，输入 AWSSESSendingGroupDoNotRename。
9. 从将用户添加到组表中选择 SMTP 用户，即可将其添加到组中。
10. 附加之前创建的 AmazonSesSendingAccess 策略，方法是从附加权限策略表中选择该策略，然后选择创建用户组。

有关将 SES 与 IAM 搭配使用的更多信息，请参阅Amazon SES 中的 Identity and Access Management。

> **Note**
>
> 尽管您可以为任何 IAM 用户生成 SES SMTP 凭证，但我们建议您在生成 SMTP 凭证时创建单独的 IAM 用户。有关为何说这是为特定目的创建用户的最佳实践的原因的信息，请参阅 IAM 最佳实践。

以下伪代码显示了将私有访问 AWS 密钥转换为 SES SMTP 密码的算法。

```
// 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);
```

一些编程语言包括可用于将 IAM 秘密访问密钥转换为 SMTP 密码的库。本节包含一个代码示例，您可以使用该示例使用 Python 将私有访问 AWS 密钥转换为 SES SMTP 密码。

> **Note**
>
> - 以下示例使用 Python 3.6 中推出的 f-string；如果使用较旧的版本，那么它们无法工作。

- 在以下示例中，SMTP_REGIONS 列表只是一个示例，您的实际区域列表可能会更短或更长，具体取决于您计划向哪些地区发送电子邮件，因为每个区域都需要 SMTP 凭证。 AWS 区域

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()
```

要使用此脚本来获取 SMTP 密码，请将前面的代码保存为 `smtp_credentials_generate.py`。然后，在命令行处，运行以下命令：

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

在上述命令中，执行以下操作：

- *path/to/*替换为保存位置的路径smtp_credentials_generate.py。
- 将 *wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY* 替换为您要转换为 SMTP 密码的秘密访问密钥。
- *us-east-1*替换为您要在其中使用 SMTP 凭证的 AWS 区域。

当此脚本成功运行时，唯一的输出是您的 SMTP 密码。

将 SMTP 用户从现有内联策略迁移到组策略（安全建议）

> **Important**
>
> 如果您在 2024 年 9 月 6 日之前创建了 SES SMTP 凭证，则已将内联策略和标签附加到您的 SMTP 用户。SES 不再使用内联策略，并出于安全建议，鼓励您也这样做。

在将 SMTP 用户从现有内联策略迁移到组策略之前，必须先创建一个具有 SES 权限策略的 IAM 用户组来取代内联策略。如果您已创建了此 IAM 用户组，或者它是从 2024 年 9 月 6 日起为您创建的 SMTP 凭证自动创建的，则可以直接跳到以下过程中的步骤 10。

从现有内联策略迁移到托管组

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在访问管理下，选择策略，然后选择创建策略。
3. 在策略编辑器中，选择 JSON，然后删除编辑器中的所有示例代码。
4. 将下面的策略粘贴到编辑器中。

   JSON

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

```
    ]
}
```

5. 选择下一步，然后在策略名称字段中输入 AmazonSesSendingAccess，接着选择创建策略。
6. 在访问管理下，选择用户组，然后选择创建组。
7. 在用户组名称字段中，输入 AWSSESSendingGroupDoNotRename。
8. 从将用户添加到组表中选择 SMTP 用户，即可将其添加到组中。
9. 附加之前创建的 AmazonSesSendingAccess 策略，方法是从附加权限策略表中选择该策略，然后选择创建用户组。

   现在，您已使用 SES 权限策略创建了 IAM 用户组，您可以按照其余步骤中所述，将 SMTP 用户从其当前的内联策略迁移到该组策略。
10. 在访问管理下，选择用户，然后选择要迁移的 SMTP 用户。
11. 选择组选项卡，然后选择将用户添加到多个组。
12. 选择 AWSSESSendingGroupDoNotRename 组，然后选择将用户添加到组。
13. 选择权限选项卡，确认在策略名称列中列出的两行均显示 AmazonSesSendingAccess，其中一行在通过如下方式附加列中显示为内联，另一行显示为组 *AWSSESSendingGroupDoNotRename*。
14. 仅选择策略名称列包含 AmazonSesSendingAccess 且通过如下方式附加列显示为内联的行，然后选择移除，接着确认移除策略。

    验证通过如下方式附加列中显示为组 *AWSSESSendingGroupDoNotRename* 的行仍然存在。
15. 选择标签选项卡，然后选择管理标签。
16. 选择 “键” 列和 SESConsole“值” 列InvokedBy中包含的行旁边的 “删除”，然后选择 “保存更改”。

> **Important**
>
> 必须将 AmazonSesSendingAccess 策略（无论是作为内联策略还是组策略，或者两者兼有）保持附加到 SMTP 用户，以确保他们的发送不受影响。只有在将组策略附加到用户之后，才能移除内联策略。

连接到 Amazon SES SMTP 端点

要使用 Amazon SES SMTP 接口来发送电子邮件，您将连接到 SMTP 端点。有关 Amazon SES SMTP 端点的完整列表，请参阅《AWS 一般参考》中的 Amazon Simple Email Service 端点和限额。

Amazon SES SMTP 终端节点要求使用传输层安全性（TLS）对所有连接进行加密。（请注意，提及 TLS 时通常使用其前身协议的名称：SSL）。Amazon SES 支持两种建立 TLS 加密连接的机制：STARTTLS 和 TLS Wrapper。查看您的软件文档，以确定它是支持 STARTTLS 或 TLS Wrapper，还是同时支持二者。

默认情况下，亚马逊弹性计算云 (Amazon EC2) 会限制端口 25 上的电子邮件流量。为避免通过 SMTP 端点从发送电子邮件时出现超时 EC2，请提交移除电子邮件发送限制的请求以取消限制。或者，您也可以使用不同的端口发送电子邮件，或者使用 Amazon VPC 端点。

有关 SMTP 连接问题，请参阅SMTP 问题。

STARTTLS

STARTTLS 是一种将未加密的连接升级到加密连接的方式。提供了适用于各种协议的 STARTTLS 版本；SMTP 版本已在 RFC 3207 中定义。

要设置 STARTTLS 连接，SMTP 客户端连接到端口 25、587 或 2587 上的 Amazon SES SMTP 端点，发出 EHLO 命令，然后等待服务器宣布其支持 STARTTLS SMTP 扩展。之后，该客户端发出 STARTTLS 命令，同时启动 TLS 协商。在协商完成后，该客户端将通过新的加密连接发出 EHLO 命令，并且 SMTP 会话将正常继续。

TLS Wrapper

TLS Wrapper (也称为 SMTPS 或握手协议) 是一种在无需事先建立未加密连接的情况下启动加密连接的方式。利用 TLS Wrapper，Amazon SES SMTP 端点不执行 TLS 协商：客户端负责使用 TLS 连接到端点，然后继续对整个对话使用 TLS。虽然 TLS Wrapper 是一项旧协议，但许多客户端仍支持它。

要设置 TLS Wrapper 连接，SMTP 客户端将连接到端口 465 或 2465 上的 Amazon SES SMTP 端点。服务器将提供其证书，客户端将发出 EHLO 命令，并且 SMTP 会话将正常继续。

使用软件包通过 Amazon SES 发送电子邮件

有大量支持通过 SMTP 发送电子邮件的商业开源软件包。下面是一些示例：

- 博客平台
- RSS 聚合器

- 列表管理软件
- 工作流系统

您可以将任何类似的支持 SMTP 的软件配置为通过 Amazon SES SMTP 接口发送电子邮件。有关如何为特定软件包配置 SMTP 的说明，请参阅该软件的相关文档。

以下过程介绍如何在 JIRA（一个常用的问题跟踪解决方案）中设置 Amazon SES 发送。利用此配置，JIRA 可以在软件问题的状态发生改变时通过电子邮件通知用户。

将 JIRA 配置为使用 Amazon SES 发送电子邮件

1. 借助 Web 浏览器，使用管理员凭证登录 JIRA。
2. 在浏览器窗口中，选择 Administration。
3. 在 System 菜单上，选择 Mail。
4. 在 Mail administration 页面上，选择 Mail Servers。
5. 选择 Configure new SMTP mail server。
6. 在 Add SMTP Mail Server 表单上，填写以下字段：

    a. Name – 此服务器的描述性名称。

    b. From address (发件人地址) – 发送电子邮件的地址。您必须先使用 Amazon SES 验证该电子邮件地址，然后才能从该地址发送电子邮件。有关验证的更多信息，请参阅Amazon SES 中已验证的身份。

    c. Email prefix – JIRA 在发送前添加到每个主题行前面的字符串。

    d. Protocol – 选择 SMTP。

    > **Note**
    >
    > 如果您无法使用此设置连接到 Amazon SES，请试用 SECURE_SMTP。

    e. Hostname (主机名称) – 有关 Amazon SES SMTP 端点的列表，请参阅连接到 Amazon SES SMTP 端点。例如，如果您要在美国西部（俄勒冈州）区域中使用 Amazon SES 端点，那么主机名会是 email-smtp.us-west-2.amazonaws.com。

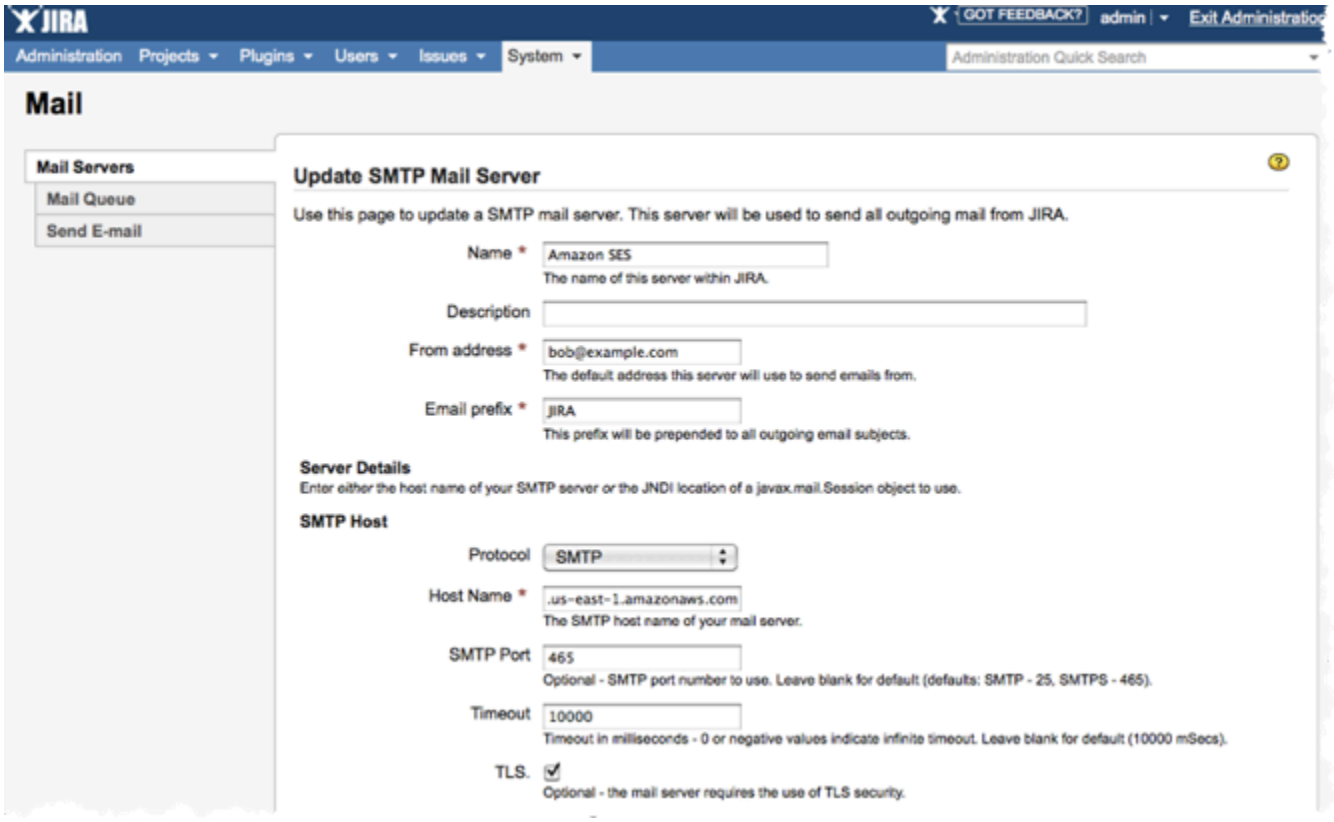
    f. SMTP port – 25、587 或 2587 (使用 STARTTLS 进行连接) 或者 465 或 2465 (使用 TLS Wrapper 进行连接)。

    g. TLS – 选中此复选框。

    h. User name – 您的 SMTP 用户名。

i. Password – 您的 SMTP 密码。

您可以在下图中看到 TLS Wrapper 的设置。



7. 选择测试连接。如果 JIRA 通过 Amazon SES 发送的测试电子邮件成功送达，则表示您的配置已完成。

通过 Amazon SES SMTP 接口以编程方式来发送电子邮件

要使用 Amazon SES SMTP 接口来发送电子邮件，您可以使用支持 SMTP 的编程语言、电子邮件服务器或应用程序。在开启之前，请完成设置 Amazon Simple Email Service中的任务。建议您了解以下信息：

- 您的 Amazon SES SMTP 凭证，用于连接到 Amazon SES SMTP 端点。要获取您的 Amazon SES SMTP 凭证，请参阅获取 Amazon SES SMTP 凭证。

> **Important**
>
> 您的 SMTP 凭证与您的 AWS 凭证不同。有关凭证的更多信息，请参阅Amazon SES 凭证的类型。

- SMTP 端点地址。有关 Amazon SES SMTP 端点的列表，请参阅连接到 Amazon SES SMTP 端点。
- Amazon SES SMTP 接口端口号，取决于连接方式。有关更多信息，请参阅 连接到 Amazon SES SMTP 端点。

代码示例

您可以使用支持 SMTP 的编程语言来访问 Amazon SES SMTP 接口。您可以提供 Amazon SES SMTP 主机名和端口号以及您的 SMTP 凭证，然后使用编程语言的一般 SMTP 函数来发送电子邮件。

默认情况下，亚马逊弹性计算云 (Amazon EC2) 限制通过端口 25 的电子邮件流量。为了避免通过 Amazon 的 SMTP 终端节点发送电子邮件时出现超时 EC2，您可以请求取消这些限制。有关更多信息，请参阅如何从我的 Amazon EC2 实例或 AWS Lambda 函数中移除对端口 25 的限制？ 在 AWS 知识中心中。

本节中的 Java 和 PHP 代码示例使用端口 587 来避免此问题。

> **Note**
>
> 在这些教程中，您将向自己发送电子邮件，以便检查是否收到了该电子邮件。如需进一步试验或进行负载测试，请使用 Amazon SES 邮箱模拟器。您发送到邮箱模拟器的电子邮件不会计入您的发送配额或您的退信率和投诉率。有关更多信息，请参阅 手动使用邮箱模拟器。

选择一种编程语言以查看该语言的示例：

> **Warning**
>
> Amazon SES 不建议使用静态凭证。请参阅 AWS Secrets Manager，了解如何通过从源代码中移除硬编码的凭证来改善安全状况。本教程仅用于在非生产环境中测试 Amazon SES SMTP 接口。

Java

此示例使用 Eclipse IDE 和 JavaMail API 使用 SMTP 接口通过 Amazon SES 发送电子邮件。

执行以下步骤之前，请完成 设置 Amazon Simple Email Service 中的任务。

将 Amazon SES SMTP 接口与 Java 结合使用来发送电子邮件

1. 在 Web 浏览器中，转到该JavaMail GitHub 页面。在 “资产” 下，选择 javax.mail.jar 以下载最新版本的。 JavaMail

   > **Important**
   >
   > 本教程需要 JavaMail 版本 1.5 或更高版本。这些程序已使用 JavaMail 版本 1.6.1 进行了测试。

2. 在网络浏览器中，进入雅加达激活 GitHub 页面，在 “激JavaBeans 活框架 1.2.1 最终版本” 下，下载 jakarta.activation.jar
3. 通过执行以下步骤在 Eclipse 中创建项目：

   a. 启动 Eclipse。
   b. 在 Eclipse 中，依次选择 File、New 和 Java Project。
   c. 在 Create a Java Project 对话框中，键入项目名称，然后选择 Next。
   d. 在 Java Settings 对话框中，选择 Libraries 选项卡。
   e. 选择 Classpath，然后使用添加外部按钮添加两个外部 jar 文件 j avax.mail.jar 和 jakarta.activation.jar。 JARs
   f. 选择 “添加外部” JARs。
   g. 浏览到您下载的文件夹 JavaMail。选择文件 `javax.mail.jar`，然后选择 Open。
   h. 在 Java Settings 对话框中，选择 Finish。

4. 在 Eclipse 中的 Package Explorer 窗口中，展开您的项目。
5. 在您的项目下，右键单击 src 目录，选择 New，然后选择 Class。
6. 在 New Java Class 对话框中的 Name 字段中，键入 `AmazonSESSample`，然后选择 Finish。
7. 将 Amazon SESSample .java 的全部内容替换为以下代码：

```
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. 在 Amazon SESSample .java 中，用您自己的值替换以下电子邮件地址：

> **Important**
>
> 电子邮件地址区分大小写。请确保此处的地址与经验证的地址完全相同。

- *sender@example.com*— 替换为您的 “发件人” 电子邮件地址。运行此程序之前，您必须验证该地址。有关更多信息，请参阅 Amazon SES 中已验证的身份。
- *recipient@example.com*— 替换为您的 “收件人” 电子邮件地址。如果您的账户仍处于沙盒中，您还必须验证此地址，然后才能使用它。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。

9. 在 Amazon SESSample .java 中，将以下内容替换为你自己的值：

- *smtp_username*— 替换为您的 SMTP 用户名凭证。请注意，您的 SMTP 用户名凭证是一个 20 个字符的字母数字字符串，而不是可识别的名称。
- *smtp_password*— `fetchSMTPPasswordFromSecureStorage` 实现获取密码。

10. （可选）如果您想在以 AWS 区域 外的地方使用 Amazon SES SMTP 终端节点*email-smtp.us-west-2.amazonaws.com*，请HOST将变量的值更改为您要使用的终端节点。有关已推出 Amazon SES 的区域的列表，请参阅《AWS 一般参考》中的 Amazon Simple Email Service（Amazon SES）。

11. （可选）如果要在发送此电子邮件时使用配置集，请将变量 *ConfigSet* 的值更改为配置集的名称。有关配置集的更多信息，请参阅在 SES 中使用配置集。
12. 保存亚马逊 SESSample .java。
13. 要构建项目，请选择 Project，然后选择 Build Project。(如果禁用此选项，则您可能已启用自动构建。)
14. 要开始程序和发送电子邮件，请选择 Run，然后再次选择 Run。
15. 检查输出。如果已成功发送电子邮件，则控制台会显示“Email sent!”。否则，将显示一条错误消息。
16. 登录收件人地址的电子邮件客户端。您将看到已发送的电子邮件。

PHP

此示例使用该 PHPMailer 类通过 SMTP 接口通过 Amazon SES 发送电子邮件。

执行以下步骤之前，必须完成 设置 Amazon Simple Email Service 中的任务。除了设置 Amazon SES，您还必须完成以下先决条件才能使用 PHP 发送电子邮件：

先决条件：

- 安装 PHP – 访问 http://php.net/downloads.php 可获得 PHP。安装 PHP 后，在环境变量中添加 PHP 的路径，这样就能通过任何命令提示符运行 PHP。
- 安装 Composer 依赖项管理器 — 安装 Composer 依赖项管理器后，您可以下载并安装该 PHPMailer 类及其依赖项。要安装 Composer，请按照 pos https://getcomer.org/downlo ad 上的安装说明进行操作。
- 安装该 PHPMailer 类 — 安装 Composer 后，运行以下命令进行安装 PHPMailer：

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

在前面的命令中，*path/to/*替换为安装 Composer 的路径。

将 Amazon SES SMTP 接口与 PHP 结合使用来发送电子邮件

1. 创建一个名为 amazon-ses-smtp-sample.php 的文件。使用文本编辑器打开文件并粘贴以下代码：

```
<?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. 在 amazon-ses-smtp-sample.php 中，用你自己的值替换以下内容：

- *sender@example.com*— 替换为您已通过 Amazon SES 验证的电子邮件地址。有关更多信息，请参阅 已验证的身份。Amazon SES 中的电子邮件地址区分大小写。请确保您输入的地址与经验证的地址完全相同。
- *recipient@example.com*— 替换为收件人的地址。如果您的账户仍处于沙盒中，您还必须验证此地址，然后才能使用它。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。请确保您输入的地址与经验证的地址完全相同。
- *smtp_username*— 替换为您从 Amazon SES 控制台的 SMTP 设置页面获得的 SMTP 用户名证书。这与您的 访问密钥 ID 不同 AWS 。请注意，您的 SMTP 用户名凭证是一个 20 个字符的字母数字字符串，而不是可识别的名称。
- *smtp_password*— `fetchSMTPPasswordFromSecureStorage` 实现获取密码。
- （可选）*ConfigSet*— 如果您想在发送此电子邮件时使用配置集，请将此值替换为配置集的名称。有关配置集的更多信息，请参阅在 SES 中使用配置集。
- （可选）*email-smtp.us-west-2.amazonaws.com*— 如果您想在美国西部（俄勒冈）以外的地区使用 Amazon SES SMTP 终端节点，请将其替换为您要使用的区域中的 Amazon SES SMTP 终端节点。有关可用 Amazon SES 的 SMTP 终端节点 URLs 列表，请参阅中的 AWS 区域 亚马逊简单电子邮件服务 (Amazon SES) Service。AWS 一般参考

3. 保存 amazon-ses-smtp-sample.php。
4. 要运行该程序，请在与 amazon-ses-smtp-sample.php 相同的目录中打开命令提示符，然后键入php amazon-ses-smtp-sample.php。
5. 检查输出。如果已成功发送电子邮件，则控制台会显示“Email sent!”。否则，将显示一条错误消息。
6. 登录收件人地址的电子邮件客户端。您将看到已发送的电子邮件。

将 Amazon SES 与您的现有电子邮件服务器集成

如果您目前管理自己的电子邮件服务器，那么您可以使用 Amazon SES SMTP 端点将所有传出电子邮件发送到 Amazon SES。您无需修改现有电子邮件客户端和应用程序；转换到 Amazon SES 的过程对它们来说是透明的。

一些邮件传输代理 (MTAs) 支持通过 SMTP 中继发送电子邮件。本节提供一般指导，说明如何配置一些常 MTAs 用的电子邮件以使用 Amazon SES SMTP 接口发送电子邮件。

Amazon SES SMTP 端点要求使用传输层安全性（TLS）对所有连接进行加密。

主题

- 将 Amazon SES 与 Postfix 集成
- 将 Amazon SES 与 Sendmail 集成
- 将 Amazon SES 与 Microsoft Windows Server IIS SMTP 集成

将 Amazon SES 与 Postfix 集成

Postfix 是广泛使用的 Sendmail 邮件传输代理 (MTA) 的替代品。有关 Postfix 的更多信息，请转到 http://www.postfix.org。本主题中的步骤适用于 Linux、macOS 或 Unix。

> **Note**
>
> Postfix 是第三方应用程序，不由 Amazon Web Services 开发或支持。本节中的步骤仅供参考，如有变更，恕不另行通知。

先决条件

完成此部分中的过程之前，必须先执行以下任务：

- 卸载 Sendmail 应用程序（如果它已安装在您的系统中）。根据您使用的不同操作系统，完成此步骤的过程会有差异。

  > **Important**
  >
  > 以下对 sendmail 的引用指的是 Postfix 命令 `sendmail`，不要与 Sendmail 应用程序混淆。

- 安装 Postfix。根据您使用的不同操作系统，完成此步骤的过程会有差异。
- 安装 SASL 身份验证软件包。根据您使用的不同操作系统，完成此步骤的过程会有差异。例如，如果您使用 RedHat基于系统的系统，则应安装该`cyrus-sasl-plain`软件包。如果您使用基于 Debian 或 Ubuntu 的系统，则应安装 `libsasl2-modules` 软件包。
- 验证您用来发送电子邮件的电子邮件地址或域。有关更多信息，请参阅 创建电子邮件地址身份。
- 如果您的账户仍处于沙盒中，您只能将电子邮件发送到已验证的电子邮件地址。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。

配置 Postfix

完成以下步骤来配置您的邮件服务器，以使用 Postfix 通过 Amazon SES 发送电子邮件。

配置 Postfix

1. 在命令行处，键入以下命令：

```
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**
>
> 如果您在美国西部（俄勒冈）以外的 AWS 地区使用 Amazon SES，请将前面的命令替换*email-smtp.us-west-2.amazonaws.com*为相应区域的 SMTP 终端节点。有关更多信息，请参阅 the section called “区域”。

2. 在文本编辑器中，打开 `/etc/postfix/master.cf`文件。搜索以下条目：

```
-o smtp_fallback_relay=
```

如果找到此条目，请在行首放置 #（井号）字符将其注释掉。保存并关闭文件。

如果此条目不存在，请继续下一个步骤。

3. 在文本编辑器中，打开 `/etc/postfix/sasl_passwd`文件。如果该文件尚不存在，请创建它。
4. 将以下行添加到 `/etc/postfix/sasl_passwd`：

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

> **Note**
>
> 将*SMTPUSERNAME*和*SMTPPASSWORD*替换为您的 SMTP 登录凭据。您的 SMTP 登录凭证与您的 AWS 访问密钥 ID 和秘密访问密钥不同。有关凭证的更多信息，请参阅the section called “获取 SMTP 凭证”。

如果您在美国西部（俄勒冈）以外的 AWS 地区使用 Amazon SES，请使用相应区域的 SMTP 终端节点替换上述示例*email-smtp.us-west-2.amazonaws.com*中的终端节点。有关更多信息，请参阅 the section called “区域”。

是否保存并关闭 `sasl_passwd`。

5. 在命令提示符处，键入以下命令以创建一个包含您的 SMTP 凭证的 Hashmap 数据库文件：

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

6. （可选）您在之前的步骤中创建的 `/etc/postfix/sasl_passwd` 和 `/etc/postfix/sasl_passwd.db` 文件未加密。由于这些文件包含您的 SMTP 凭证，我们建议您修改这些文件的所有权和权限以限制对它们的访问。要限制对这些文件的访问，请执行以下操作：

   a. 在命令提示符处，键入以下命令以更改文件的所有权：

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

   b. 在命令提示符处，键入以下命令以更改文件的权限，以便仅根用户可读取和写入它们：

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

7. 告诉 Postfix 在何处查找 CA 证书（需要使用它来验证 Amazon SES 服务器证书）。您在此步骤中使用的命令因所用操作系统而异。

   - 如果您使用 Amazon Linux、Red Hat Enterprise Linux 或相关分发，请键入以下命令：

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

   - 如果您使用 Ubuntu 或相关分发，请键入以下命令：

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

   - 如果您使用 macOS，则可通过您的系统密钥链生成证书。要生成证书，请在命令行处键入以下命令：

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

   生成证书之后，请键入以下命令：

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

8. 键入以下命令以启动 Postfix 服务器（或在服务器已在运行的情况下，重新加载配置设置）：

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

9. 在命令行键入以下命令并在每行后按 Enter 以发送测试电子邮件。*sender@example.com*替换为您的“发件人”电子邮件地址。From (发件人) 地址必须经过验证才能用于 Amazon SES。*recipient@example.com*替换为目标地址。如果您的账户仍处于沙盒中，则还必须验证收件人地址。最后，邮件最后一行只能包含一个句点 (.)，不能再有其他内容。

```
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. 检查与收件人地址关联的邮箱。如果未收到电子邮件，请检查您的垃圾邮件文件夹。如果您仍然无法找到电子邮件，请检查用于发送电子邮件的系统上的邮件日志（通常位于 `/var/log/maillog`）以了解更多信息。

高级使用示例

此示例演示如何发送使用配置集的电子邮件，以及如何使用 MIME 多部分编码发送纯文本和 HTML 版本的消息以及附件。它还包含一个链接标签，可用于为单击事件分类。电子邮件的内容将在一个外部文件中指定，因此您不必在 Postfix 会话中手动键入命令。

使用 Postfix 发送多部分 MIME 电子邮件

1. 在文本编辑器中，创建一个名为 `mime-email.txt` 的新文件。
2. 在文本文件中，粘贴以下内容，并将红色的值替换为适用于您的账户的值：

```
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--
```

保存并关闭文件。

3. 在命令行处，键入以下命令。*sender@example.com*替换为您的电子邮件地址，然后*recipient@example.com*替换为收件人的电子邮件地址。

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

如果该命令成功运行，它将退出并且不提供任何输出。

4. 查看收件箱中是否有该电子邮件。如果该邮件未送达，请查看系统的邮件日志。

将 Amazon SES 与 Sendmail 集成

Sendmail 最初于 20 世纪 80 年代初发布，此后一直在不断改进。这是一个灵活且可配置的邮件传输代理 (MTA) 并拥有大型的用户社群。Sendmail 于 2013 年被 Proofpoint 收购，但 Proofpoint 继续提供开源版本的 Sendmail。您可以从 Proofpoint 网站或通过大多数 Linux 发行版的程序包管理器下载 Sendmail 的开源版本。

本节中的操作步骤说明如何配置 Sendmail 以通过 Amazon SES 发送电子邮件。已在运行 Ubuntu 18.04.2 LTS 的服务器上测试此过程。

> **Note**
>
> Sendmail 是第三方应用程序，不由 Amazon Web Services 开发或支持。本节中的步骤仅供参考，如有变更，恕不另行通知。

先决条件

在完成本节中的过程之前，您应完成以下步骤：

- 在您的服务器上安装 Sendmail 程序包。

  > **Note**
  >
  > 根据您使用的操作系统发行版，可能还需要安装以下程序包：`sendmail-cf`、`m4` 和 `cyrus-sasl-plain`。

- 验证要用作“From (发件人)”地址的身份。有关更多信息，请参阅 创建电子邮件地址身份。

如果您的账户仍在 Amazon SES 沙盒中，那么您还必须验证您将电子邮件发送到的地址。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。

如果您使用 Amazon SES 从亚马逊 EC2 实例发送电子邮件，则还应完成以下步骤：

- 您可能需要为您的 Amazon EC2 实例分配弹性 IP 地址，以便接收电子邮件的提供商接受您的电子邮件。有关更多信息，请参阅亚马逊 EC2 用户指南中的亚马逊 EC2 弹性 IP 地址。
- 默认情况下，亚马逊弹性计算云 (Amazon EC2) 限制通过端口 25 的电子邮件流量。为了避免通过 Amazon 的 SMTP 终端节点发送电子邮件时出现超时 EC2，您可以请求取消这些限制。有关更多信息，请参阅如何从我的 Amazon EC2 实例或 AWS Lambda 函数中移除对端口 25 的限制？ 在 AWS 知识中心中。

  或者，您可以修改本节中的操作步骤，以使用端口 587 而不是端口 25。

配置 Sendmail

完成本节中的步骤，通过使用 Amazon SES 配置 Sendmail 来发送电子邮件。

> **Important**
>
> 本节中的步骤假设您想在美国西部（俄勒冈）使用 Amazon SES AWS 区域。如果您要使用其他区域，请将此过程中的所有 email-smtp.us-west-2.amazonaws.com 实例替换为所需区域的 SMTP 端点。有关可用 Amazon SES 的 SMTP 终端节点 URLs 列表，请参阅中的 AWS 区域 亚马逊简单电子邮件服务 (Amazon SES) Service。AWS 一般参考

配置 Sendmail

1. 在文件编辑器中，打开文件 `/etc/mail/authinfo`。如果该文件不存在，请创建它。

   在/中添加以下行etc/mail/authinfo：

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

   在前面的示例中，进行以下更改：

- *email-smtp.us-west-2.amazonaws.com*替换为您要使用的 Amazon SES SMTP 终端节点。
- *smtpUsername*替换为您的 Amazon SES SMTP 用户名。
- *smtpPassword*替换为您的 Amazon SES SMTP 密码。

> **Note**
>
> 您的 SMTP 登录凭据不同于您的 AWS 访问密钥 ID 和私有访问密钥。有关获取您的 SMTP 登录凭证的更多信息，请参阅获取 Amazon SES SMTP 凭证。

完成后，保存 authinfo。

2. 在命令行中，输入以下命令可生成 /etc/mail/authinfo.db 文件：

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

3. 在命令行中，键入以下命令可添加对中继到 Amazon SES SMTP 端点的支持。

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

在前面的命令中，*email-smtp.us-west-2.amazonaws.com*替换为您要使用的 Amazon SES SMTP 终端节点的地址。

4. 在命令行中，键入以下命令以重新生成/etc/mail/access.db：

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

5. 在命令行中，键入以下命令可创建 sendmail.cf 和 sendmail.mc 文件的备份：

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

6. 将以下几行添加到/etc/mail/sendmail.mc 文件中，然后再进行任何MAILER()定义。

```
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
```

在上述文本中，执行以下操作：

- *email-smtp.us-west-2.amazonaws.com*替换为您要使用的 Amazon SES SMTP 终端节点。
- *example.com*替换为您要用于发送电子邮件的域名。

完成后，保存该文件。

> **Note**
>
> 默认情况下，Amazon EC2 限制通过端口 25 进行通信。如果您在亚马逊 EC2 实例中使用 Sendmail，则应完成取消电子邮件发送限制的请求。

7. 在命令行中，键入以下命令可将 sendmail.cf 设为可写：

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

8. 在命令行中，键入以下命令可重新生成 sendmail.cf：

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

> **Note**
>
> 如果您遇到“Command not found (找不到命令)”和“No such file or directory (无此类文件或目录)”等错误，请确保您的系统上安装了 m4 和 sendmail-cf 程序包。

9. 在命令行中，键入以下命令可将 sendmail.cf 的权限重置为只读：

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

10. 在命令行中，键入以下命令可重新启动 Sendmail：

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

如果上述方法不起作用，请尝试以下操作，具体取决于 Linux 或 Sendmail 的版本：

```
sudo su service sendmail restart
```

11. 完成以下步骤可发送测试电子邮件：

    a. 在命令行输入以下命令。

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

    *sender@example.com*替换为您的“发件人”电子邮件地址。*recipient@example.com*替换为收件人地址。完成后，按 Enter。

    b. 输入以下消息内容。在每行的结尾处按 Enter。

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

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

    在输入完电子邮件内容后，按 Ctrl+D 发送电子邮件。

12. 检查收件人的电子邮件客户端是否收到这封电子邮件。如果您无法找到电子邮件，请检查垃圾邮件文件夹。如果您仍无法找到电子邮件，请查看邮件服务器上的 Sendmail 日志。日志通常位于/var/log/mail.log 或/var/log/maillog。

将 Amazon SES 与 Microsoft Windows Server IIS SMTP 集成

您可以将 Microsoft Windows Server 的 IIS SMTP 服务器配置为通过 Amazon SES 发送电子邮件。这些说明是在亚马逊 EC2 实例上使用微软 Windows Server 2022 编写的。你可以在微软 Windows Server 2016 上使用相同的配置。

> **Note**
>
> Windows Server 是第三方应用程序，不由 Amazon Web Services 开发或支持。本节中的步骤仅供参考，如有变更，恕不另行通知。

将 Microsoft Windows Server IIS SMTP 服务器与 Amazon SES 集成

1. 首先，按照以下说明设置微软 Windows Server 2022。

   a. 从亚马逊 EC2 管理控制台启动新的微软 Windows Server 2022 Base 亚马逊 EC2实例。

   b. 按照 Amazon EC2 Windows 实例入门中的说明使用远程桌面连接到该实例并登录该实例。

   c. 启动服务器管理器控制面板。

   d. 安装 Web Server 角色。请确保包含 IIS 6 Management Compatibility tools (Web Server 复选框下的一个选项)。

   e. 安装 SMTP Server 特征。

2. 接下来，使用以下说明配置 IIS SMTP 服务。

   a. 返回到服务器管理器控制面板。

   b. 从 Tools 菜单中，选择 Internet Information Services (IIS) 6.0 Manager。

   c. 右键单击 SMTP Virtual Server #1，然后选择 Properties。

   d. 在 Access 选项卡的 Relay Restrictions 下，选择 Relay。

   e. 在 Relay Restrictions 对话框中，选择 Add。

   f. 在 Single Computer 下，输入 127.0.0.1 作为 IP 地址。您现在已授予此服务器访问权限，以通过 IIS SMTP 服务将电子邮件中继到 Amazon SES。

      在此过程中，我们假定电子邮件是在此服务器上生成的。如果生成电子邮件的应用程序在其他服务器上运行，则您必须在 IIS SMTP 中为该服务器授予中继访问权限。

      > **Note**
      >
      > 要将 SMTP 中继扩展到私有子网，对于 Relay Restriction，请使用 Single Computer 127.0.0.1 和 Group of Computers 172.1.1.0 – 255.255.255.0 (在网络掩码部分中)。对于 Connection，请使用 Single Computer 127.0.0.1 和 Group of Computers 172.1.1.0 – 255.255.255.0 (在网络掩码部分中)。

3. 最后，使用以下说明将服务器配置为通过 Amazon SES 发送电子邮件。

   a. 返回到 SMTP Virtual Server #1 Properties 对话框，然后选择 Delivery 选项卡。

   b. 在 Delivery 选项卡上，选择 Outbound Security。

   c. 选择 Basic Authentication（基本身份验证），然后输入您的 Amazon SES SMTP 凭证。您可以使用获取 Amazon SES SMTP 凭证中的过程从 Amazon SES 控制台获取这些凭证。

> **Important**
>
> 您的 SMTP 凭证与您的 AWS 访问密钥 ID 和私有访问密钥不同。请勿尝试使用您的 AWS 凭据通过 SMTP 端点对自己进行身份验证。有关凭证的更多信息，请参阅Amazon SES 凭证的类型。

d. 确保已选中 TLS encryption。
e. 返回到 Delivery 选项卡。
f. 选择 Outbound Connections。
g. 在 Outbound Connections 对话框中，确保端口为 25 或 587。
h. 选择 Advanced (高级)。
i. 对于 Smart host (智能主机) 名称，输入您将使用的 Amazon SES 端点（例如，email-smtp.us-west-2.amazonaws.com）。有关可用 Amazon SES URLs AWS 区域 的终端节点列表，请参阅中的AWS 一般参考亚马逊简单电子邮件服务 (Amazon SES) Service。
j. 返回到服务器管理器控制面板。
k. 在服务器管理器控制面板上，右键单击 SMTP Virtual Server #1，然后重新启动该服务以选择新配置。
l. 通过此服务器发送电子邮件。您可以检查邮件标头，以确认邮件已通过 Amazon SES 送达。

使用命令行来测试与 Amazon SES SMTP 接口的连接

您可以从命令行使用本节中介绍的方法来测试与 Amazon SES SMTP 端点的连接、验证 SMTP 凭证以及解决连接问题。这些过程将使用大多数常见操作系统附带的工具和库。

有关解决 SMTP 连接问题的其他信息，请参阅 Amazon SES SMTP 问题。

先决条件

在连接到 Amazon SES SMTP 接口时，您必须提供一组 SMTP 凭证。这些 SMTP 凭证不同于您的标准 AWS 凭证。这两种类型的凭证不可互换。有关获取您的 SMTP 凭证的更多信息，请参阅the section called “获取 SMTP 凭证”。

测试到 Amazon SES SMTP 接口的连接

您可以使用命令行来测试您到 Amazon SES SMTP 接口的连接，而无需验证或发送任何消息。此过程可用于基本连接问题的故障排除。如果测试连接失败，请参阅SMTP 问题。

本节包括使用 OpenSSL（大多数 Linux、macOS 和 Unix 发行版中都包含此功能，也适用于 Windows）和中的 `Test-NetConnection` cmdlet PowerShell（包含在最新版本的 Windows 中）测试连接的过程。

Linux, macOS, or Unix

可通过两种方式使用 OpenSSL 连接到 Amazon SES SMTP 接口：在端口 587 上使用显式 SSL，或在端口 465 上使用隐式 SSL。

使用显式 SSL 连接到 SMTP 接口

- 在命令行上，输入以下命令来连接到 Amazon SES SMTP 服务器：

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

在前面的命令中，*email-smtp.us-west-2.amazonaws.com*替换为您所在 AWS 地区的 Amazon SES SMTP 终端节点的网址。有关更多信息，请参阅 the section called “区域”。

如果连接成功，则将显示类似于以下内容的输出：

```
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
```

连接会在处于不活动状态大约 10 秒后自动关闭。

或者，您可以使用隐式 SSL 通过端口 465 连接到 SMTP 接口。

使用隐式 SSL 连接到 SMTP 接口

- 在命令行上，输入以下命令来连接到 Amazon SES SMTP 服务器：

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

在前面的命令中，*email-smtp.us-west-2.amazonaws.com*替换为您所在 AWS 地区的 Amazon SES SMTP 终端节点的网址。有关更多信息，请参阅 the section called “区域”。

如果连接成功，则将显示类似于以下内容的输出：

```
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
```

连接会在处于不活动状态大约 10 秒后自动关闭。

PowerShell

你可以使用 T est-NetConnection cmdlet 连接 PowerShell 到 Amazon SES SMTP 服务器。

> **Note**
>
> `Test-NetConnection` cmdlet 可以确定您的计算机是否能连接到 Amazon SES SMTP 端点。但是，它不会测试您的计算机是否能建立与 SMTP 端点的隐式或显式 SSL 连接。要测试 SSL 连接，您可以安装 OpenSSL for Windows 来发送测试电子邮件。

使用 **`Test-NetConnection`** cmdlet 连接到 SMTP 接口

- 在中 PowerShell，输入以下命令连接到 Amazon SES SMTP 服务器：

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

在前面的命令中，*email-smtp.us-west-2.amazonaws.com*替换为您所在 AWS 地区的 Amazon SES SMTP 终端节点的 URL，并*587*替换为端口号。有关 Amazon SES 中的区域端点的更多信息，请参阅the section called “区域”。

如果连接成功，您会看到与以下示例类似的输出：

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

使用命令行通过 Amazon SES SMTP 接口发送电子邮件

您也可以使用命令行通过 Amazon SES SMTP 接口发送邮件。此过程对于测试 SMTP 凭证以及测试特定收件人能否接收您使用 Amazon SES 发送的消息非常有用。

Linux, macOS, or Unix

当电子邮件发件人连接到 SMTP 服务器时，客户端会发出一组标准请求，而服务器会使用标准响应来回复每个请求。这一系列的请求和响应称为 SMTP 对话。当您使用 OpenSSL 连接到 Amazon SES SMTP 服务器时，服务器需要进行 SMTP 对话。

在使用 OpenSSL 连接到 SMTP 接口时，您必须使用 base64 编码对 SMTP 凭证进行编码。此部分包含使用 base64 对凭证进行编码的过程。

从命令行使用 SMTP 接口发送电子邮件

1. 在命令行中输入以下内容，并*email-smtp.us-west-2.amazonaws.com*替换为您的 Amazon SES SMTP 终端节点的 AWS 区域 URL。有关更多信息，请参阅the section called “区域”。：

```
#!/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. 在提示输入每个变量时，输入您的值。
3. • 要通过端口 465 使用隐式 SSL 发送，请使用：

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

如果消息已被 Amazon SES 接受，您将看到与以下示例类似的输出：

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

250 Ok 后面的数字和文本字符串是电子邮件的邮件 ID。

> **Note**
>
> 连接会在处于不活动状态大约 10 秒后自动关闭。

PowerShell

你可以使用 Net.Mail。 SmtpClient使用显式 SSL 通过端口 587 发送电子邮件的类。

> **Note**
>
> Net.Mail.SmtpClient 类已正式被弃用，Microsoft 建议您使用第三方库。此代码仅用于测试目的，而不应用于生产工作负载。

PowerShell 使用显式 SSL 发送电子邮件

1. 在文本编辑器中，创建一个新文件。将以下代码粘贴到该文件中：

```
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
```

完成后，将文件另存为 SendEmail.ps1。

2. 对您在上一步中创建的文件进行以下更改：

   - *sender@example.com*替换为您要从中发送消息的电子邮件地址。
   - *recipient@example.com*替换为您要向其发送消息的电子邮件地址。
   - *email-smtp.us-west-2.amazonaws.com*替换为您所在 AWS 地区的 Amazon SES SMTP 终端节点的网址。有关更多信息，请参阅 区域和 Amazon SES。

3. 在中 PowerShell，输入以下命令：

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

在前面的命令中，*path\to\SendEmail.ps1*替换为您在步骤 1 中创建的文件的路径。

4. 在系统提示时，输入您的 SMTP 用户名和密码。

或者，你可以使用 System.Web.Mail。 SmtpMail使用隐式 SSL 通过端口 465 发送电子邮件的类。

> **Note**
>
> System.Web.Mail.SmtpMail 类已正式被弃用，Microsoft 建议您使用第三方库。此代码仅用于测试目的，而不应用于生产工作负载。

PowerShell 使用隐式 SSL 发送电子邮件

1. 在文本编辑器中，创建一个新文件。将以下代码粘贴到该文件中：

```
[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
```

完成后，将文件另存为 SendEmail.ps1。

2. 对您在上一步中创建的文件进行以下更改：

   - *sender@example.com*替换为您要从中发送消息的电子邮件地址。
   - *recipient@example.com*替换为您要向其发送消息的电子邮件地址。
   - *email-smtp.us-west-2.amazonaws.com*替换为您所在 AWS 地区的 Amazon SES SMTP 终端节点的网址。有关更多信息，请参阅 区域和 Amazon SES。

3. 在中 PowerShell，输入以下命令：

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

   在前面的命令中，*path\to\SendEmail.ps1*替换为您在步骤 1 中创建的文件的路径。

4. 在系统提示时，输入您的 SMTP 用户名和密码。

使用 Amazon SES API 发送电子邮件

要通过 Amazon SES 发送生产电子邮件，您可以使用简单邮件传输协议（SMTP）接口或 Amazon SES API。有关 SMTP 接口的更多信息，请参阅使用 Amazon SES SMTP 接口发送电子邮件。此部分介绍如何使用 API 发送电子邮件。

当您使用 Amazon SES API 发送电子邮件时，您可以指定邮件的内容，而 Amazon SES 会为您汇编 MIME 电子邮件。或者，您可以自行组装电子邮件，以便完全控制邮件的内容。有关 API 的更多信息，请参阅 Amazon Simple Email Service API 参考。有关可用 Amazon SES 的终端节点 URLs 列表，请参阅中的亚马逊简单电子邮件服务终端节点和配额AWS 一般参考。 AWS 区域

您可通过以下方式调用 API：

- 发出直接 HTTPS 请求 – 这是最先进的方法，因为您必须手动处理您的请求的身份验证和签名，然后手动构建请求。有关 Amazon SES API 的信息，请参阅 API v2 参考中的欢迎页面。
- 使用 AWS 软件开发工具包 —AWS SDKs 便于访问多种 AWS 服务，包括 Amazon SES。 APIs 当您使用开发工具包时，它会负责身份验证、请求签名、重试逻辑、错误处理以及其他低级功能，以便您可以专注于构建让客户满意的应用程序。

- 使用命令行接口 – AWS Command Line Interface 是 Amazon SES 的命令行工具。我们还 PowerShell为那些在 PowerShell环境中编写脚本的人提供AWS 工具。

无论您是直接访问 Amazon SES API 还是通过 AWS 软件开发工具包、 AWS Command Line Interface 或 AWS 工具间接访问 Amazon SES API，Amazon SES API 都为您提供了两种不同的发送电子邮件的方式，具体取决于您对电子邮件构成的控制程度： PowerShell

- 已设置格式 – Amazon SES 编写并发送格式正确的电子邮件。您只需提供 From: (发件人:) 和 To: (收件人:) 地址、主题和邮件正文。Amazon SES 将负责完成所有余下工作。有关更多信息，请参阅 使用 Amazon SES API 发送格式化的电子邮件。
- 原始 - 您手动编写和发送电子邮件，并指定您自己的电子邮件标头和 MIME 类型。如果您在设置自己的电子邮件格式方面有经验，则原始接口会为您提供对邮件内容的更多控制。有关更多信息，请参阅 使用 Amazon SES API v2 发送原始电子邮件。

内容

使用 Amazon SES API 发送格式化的电子邮件

您可以使用 AWS Management Console 或通过应用程序直接调用 Amazon SES API 来发送格式化电子邮件，也可以通过软件开发工具包间接调用 Amazon AWS SES API AWS Command Line Interface、或 AWS Tools for Windows PowerShell。

Amazon SES API 提供 `SendEmail` 操作，让您能够编写和发送格式化电子邮件。`SendEmail` 需要发件人地址、收件人地址、邮件主题和邮件正文（文本和/或 HTML）。有关更多信息，请参阅 SendEmail（API 参考）或 SendEmail（API v2 参考）。

> **Note**
>
> 电子邮件地址字符串必须为 7 位 ASCII 字符。如果您希望向或从某个地址的域部分中包含 Unicode 字符的电子邮件地址发送邮件，则必须使用 Punycode 对域进行编码。有关更多信息，请参见 RFC 3492。

有关如何使用各种编程语言编写格式化邮件的示例，请参阅 代码示例。

有关对 `SendEmail` 进行多个调用时如何加快电子邮件发送速度的提示，请参阅增加 Amazon SES 吞吐量。

使用 Amazon SES API v2 发送原始电子邮件

您可以通过内容类型指定为 `raw` 的 Amazon SES API v2 `SendEmail` 操作，使用原始电子邮件格式向收件人发送自定义消息。

关于电子邮件标头字段

简单邮件传输协议 (SMTP) 通过定义邮件信封及其部分参数来指定电子邮件将如何发送，但它本身与邮件内容无关。相反，Internet 邮件格式 (RFC 5322) 定义了如何创建邮件。

根据 Internet 邮件格式规范，每封电子邮件都包含标题和正文。标题包含邮件元数据，正文包含邮件本身。有关电子邮件标题和正文的更多信息，请参阅Amazon SES 中的电子邮件格式。

使用原始电子邮件 MIME 消息构造

SMTP 协议最初设计用于发送仅包含 7 位 ASCII 字符的电子邮件。此规范使得 SMTP 不足以应对非 ASCII 文本编码（如 Unicode）、二进制内容或附件。多用途 Internet 邮件扩展标准 (MIME) 是为了能够使用 SMTP 发送许多其他种类的内容而制定的。

MIME 标准的工作方式是将邮件正文拆分为多个段，然后指定要对每个段执行的操作。例如，电子邮件正文的一个段可能是纯文本，另一个段可能是 HTML。此外，MIME 还允许电子邮件包含一个或多个附件。邮件收件人可以在电子邮件客户端内查看附件，也可以保存附件。

邮件标题和内容由一个空白行分隔。电子邮件的每个段都由一个边界 (表示每个段的开头和末尾的一个字符串) 分隔。

以下示例中的多段邮件包含文本、HTML 部分和附件。附件应放在附件标题的正下方，并且通常按照该示例中所示采用 `base64` 编码。

```
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--
```

邮件的内容类型为 `multipart/mixed`，这表示邮件有很多个段 (在本例中为一个正文和一个附件)，且接收客户端必须单独处理每个段。

还有一个嵌套在正文部分中的段，该段使用 `multipart/alternative` 内容类型。此内容类型表示每个段都包含同一内容的替代版本 (在本例中为一个文本版本和一个 HTML 版本)。如果收件人的电子邮件客户端可以显示 HTML 内容，则它显示邮件正文的 HTML 版本。如果收件人的电子邮件客户端无法显示 HTML 内容，则它显示邮件正文的纯文本版本。

邮件的两个版本还将包含一个附件（在本例中为包含一些客户名称的短文本文件）。

当您将一个 MIME 段嵌套在另一个段中时 (如本示例所示)，嵌套的段必须使用与父段中的 `boundary` 参数不同的 `boundary` 参数。这些边界应该是唯一的字符串。要定义 MIME 段之间的边界，请键入两个连字符 (--) 后跟边界字符串。在 MIME 段的末尾，在边界字符串的开头和末尾处放置两个连字符。

> **Note**
>
> 一封邮件的 MIME 部分不得超过 500 个。

MIME 编码

为了保持与旧系统的兼容性，Amazon SES 遵循 RFC 2821 中定义的 SMTP 7 位 ASCII 限制。如果要发送包含非 ASCII 字符的内容，则必须将这些字符编码为使用 7 位 ASCII 字符的格式。

电子邮件地址

电子邮件地址字符串必须为 7 位 ASCII 字符。如果您希望向或从某个地址的域部分中包含 Unicode 字符的电子邮件地址发送邮件，则必须使用 Punycode 对域进行编码。不允许在电子邮件地址的本地部分 (@ 符号前面的部分) 中使用 Punycode，也不允许在“易记发件人”名称中使用。如果要在 “友好发件人” 名称中使用 Unicode 字符，则必须使用 MIME 编码字语法对 “友好发件人” 名称进行编码，如本节所述。有关 Punycode 的更多信息，请参阅 RFC 3492。

> **Note**
>
> 此规则仅适用于您在邮件信封中指定的电子邮件地址，不适用于邮件标头。使用 Amazon SES API v2 `SendEmail` 操作时，您在 `Source` 和 `Destinations` 参数中指定的地址分别定义信封发件人和收件人。

电子邮件标头

要对邮件标头进行编码，请使用 MIME encoded-word 语法。MIME encoded-word 语法使用以下格式：

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

encoding 的值可以是 Q 或 B。如果编码值为 Q，则 *encoded-text* 的值必须使用 Q 编码。如果编码值为 B，则 *encoded-text* 的值必须使用 base64 编码。

例如，如果您要在电子邮件的主题行中使用字符串“Як ти поживаєш?”，那么您可以使用以下任一编码：

- Q 编码

```
=?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 编码

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

有关 Q 编码的更多信息，请参阅 RFC 2047。有关 base64 编码的更多信息，请参阅 RFC 2045。

消息正文

要对邮件正文进行编码，可以使用 quoted-printable 编码或 base64 编码。然后，使用 `Content-Transfer-Encoding` 标头指示您使用的编码方案。

例如，假设邮件正文包含以下文本：

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

如果选择使用 base64 编码对此文本进行编码，请首先指定以下标头：

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

然后，在电子邮件的正文部分中，包含 base64 编码的文本：

```
4KWn4KWv4KWt4KWoIOCkruClhyDgpLDgpYcg4KSf4KWJ4KSu4KSy4KS/4KSC4KS44KSoIOCkqOCl
```

```
hyDgpKrgpLngpLLgpL4g4KSILeCkruClh+CksiDgpLjgpILgpKbgpYfgpLYg4KSt4KWH4KSc4KS+
IHwg4KSw4KWHIOCkn+ClieCkruCksuCkv+CkguCkuOCkqCDgpKjgpYcg4KS54KWAIOCkuOCksOCl
jeCkteCkquCljeCksOCkpeCkriBAIOCkmuCkv+CkqOCljeCkuSDgpJXgpL4g4KSa4KSv4KSoIOCk
leCkv+Ckr+CkviDgpJTgpLAg4KSH4KSo4KWN4KS54KWAIOCkleCliyDgpIjgpK7gpYfgpLIg4KSV
4KS+IOCkhuCkteCkteCkv+Ckt+CljeCkleCkvuCksOCklSDgpK7gpL7gpKjgpL4g4KSc4KS+4KSk4KS+
IOCkueCliAo=
```

> **Note**
>
> 在某些情况下，您可以在使用 Amazon SES 发送的邮件中使用 8 位 `Content-Transfer-Encoding`。但是，如果 Amazon SES 必须对邮件进行任何更改（例如，当您使用打开和单击跟踪）时，8 位编码的内容在到达收件人的收件箱时可能无法正确显示。因此，您应始终对不是 7 位 ASCII 的内容进行编码。

文件附件

要将文件附加到电子邮件，您必须使用 base64 编码对附件进行编码。附件通常放在专用的 MIME 邮件部分中，其中包括以下标头：

- Content-Type – 附件的文件类型。以下是常见 MIME Content-Type 声明的示例：
  - 纯文本文件 – `Content-Type: text/plain; name="sample.txt"`
  - Microsoft Word 文档 – `Content-Type: application/msword; name="document.docx"`
  - JPG 图像 – `Content-Type: image/jpeg; name="photo.jpeg"`
- Content-Disposition – 指定收件人的电子邮件客户端应如何处理内容。对于附件，此值为 `Content-Disposition: attachment`。
- Content-Transfer-Encoding – 用于对附件进行编码的方案。对于文件附件，此值几乎总是 `base64`。
- 编码的附件 - 您必须对实际附件进行编码，并将其包含在附件标题下方的正文中，如示例所示。

Amazon SES 接受最常见的文件类型。有关 Amazon SES 不接受的文件类型的列表，请参阅SES 不支持的附件类型。

使用 Amazon SES API v2 发送原始电子邮件

Amazon SES API v2 提供 `SendEmail` 操作，当您将内容类型设置为简单、原始或模板化时，该操作使您可以指定的格式编写和发送电子邮件。有关的完整说明，请参阅 `SendEmail`。以下示例将内容类型指定为 `raw`，以使用原始电子邮件格式发送消息。

> **Note**
>
> 有关对 SendEmail 进行多个调用时如何加快电子邮件发送速度的提示，请参阅增加 Amazon SES 吞吐量。

邮件正文必须包含格式正确的原始电子邮件，后者具有适当的标头字段和邮件正文编码。尽管能够在应用程序内手动构建原始邮件，但使用现有邮件库执行此操作轻松得多。

Java

以下代码示例说明如何使用JavaMail库和适用于 Java 的 AWS SDK来撰写和发送原始电子邮件。

```
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 ### Java # AWS SDK // 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

下面的代码示例说明如何使用 Python email.mime 程序包和AWS SDK for Python (Boto) 编写和发送原始电子邮件。

```
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 ()
```

使用模板通过 Amazon SES API 发送个性化电子邮件

在 Amazon SES 中，您可以使用存储的模板或内联模板发送模板化电子邮件。

- 存储的模板-指使用 Amazon SES v2 API 中的`CreateEmailTemplate`操作在 SES 中创建和保存的`Template`资源。模板包含电子邮件的主题和正文，其中包含与书面内容一致的变量（占位符）。调用`SendEmail`或 `SendBulkEmail` v2 API 操作时，会提供存储模板的名称和模板中占位符变量的动态数据。

  存储的模板可以轻松重复使用，并且可以在发送类似类型的电子邮件时为您节省时间和精力。无需从头开始创建每封电子邮件，只需创建基本结构并设计一次，然后只需更新模板中的动态内容即可。
- 内联模板 — 不使用`Template`资源，而是在调用`SendEmail`或 `SendBulkEmail` v2 API 操作时，会提供电子邮件的主题和正文，其中包含与书面内容内联的变量（占位符）以及这些占位符变量的值。

  内联模板无需管理 SES 账户中的模板资源，从而简化了发送批量电子邮件的流程，并允许您将模板内容直接包含在应用程序逻辑中，从而简化了集成过程。它们不计入每个 20,000 个模板的限制。AWS 区域

使用存储的模板时适用以下限制：

- 每个模板中最多可以创建 20,000 个电子邮件模板 AWS 区域。
- 每个模板的大小最多为 500 KB (包括文本和 HTML 部分)。

使用内联模板时适用以下限制：

- 每个输入 JSON 文件的大小可达 1 MB，包括文本和 HTML 部分。

以下内容适用于存储模板和内联模板：

- 可以使用的替换变量的数量没有限制。
- 每次调用`SendBulkEmail`操作时，您最多可以向 50 个目标对象发送电子邮件。该`Destination`对象可以包含`ToAddresses`、`CcAddresses`和中定义的多个收件人`BccAddresses`。

单次调用 v2 API 可以联系的目的地数量可能会受到您账户的最大发送速率的限制。有关更多信息，请参阅 管理您的 Amazon SES 发送限制。

本章包括使用存储模板和内联模板的过程和示例。

> **Note**
>
> 本节中的过程假定您已安装和配置 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

（可选）第 1 部分：设置渲染失败事件通知

如果您发送一封包含无效的个性化内容的电子邮件，那么 Amazon SES 可能接受该邮件，但无法传递它。因此，如果您计划发送个性化电子邮件，则应将 SES 配置为通过 Amazon SNS 发送渲染失败事件通知。当您收到呈现失败事件通知时，您可以验证哪些邮件包含无效的内容、修复问题，然后重新发送邮件。

此部分中的过程可选，不过强烈建议使用。

配置呈现失败事件通知

1. 创建 Amazon SNS 主题。有关操作步骤，请参阅《Amazon Simple Notification Service 开发人员指南》中的创建主题。
2. 订阅 Amazon SNS 主题。例如，如果您希望通过电子邮件接收呈现失败通知，请使用电子邮件端点 (即，您的电子邮件地址) 订阅主题。

   有关操作步骤，请参阅《Amazon Simple Notification Service 开发人员指南》中的订阅主题。
3. 完成the section called “设置 Amazon SNS 目标 ”中的操作步骤来设置您的配置集，以将呈现失败事件发送到您的 Amazon SNS 主题。

（可选）第 2 部分：创建电子邮件模板

如果您打算使用存储的模板，本节将向您展示如何使用 `CreateEmailTemplate`SES v2 API 操作来创建模板。如果要使用内联模板，可以跳过此步骤。

此过程假定您已安装和配置 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

创建模板

1. 在文本编辑器中，创建一个新文件并粘贴以下代码，根据需要对其进行自定义。

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

   此代码包含以下属性：

   - TemplateName— `Template` 资源的名称。当您发送电子邮件时，您将引用此名称。
   - TemplateContent— 包含以下属性的容器：
     - SubjectPart – 电子邮件的主题行。此属性可能包含替换标签。这些标签使用以下格式：`{{tagname}}`。当您发送电子邮件时，您可以为每个目标的 `tagname` 指定一个值。
     - HtmlPart— 电子邮件的 HTML 正文。此属性可能包含替换标签。前面的示例包含两个标签：`{{name}}` 和 `{{favoriteanimal}}`。
     - TextPart— 电子邮件的正文。电子邮件客户端未显示 HTML 内容的收件人将看到此版本的电子邮件。此属性也可能包含替换标签。
2. 自定义前面的示例以满足您的需求，然后将该文件另存为 *`mytemplate.json`*。
3. 在命令行中，键入以下命令以使用 `CreateEmailTemplate`v2 API 操作创建新模板：

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

第 3 部分：发送个性化电子邮件

您可以使用以下两个 SES v2 API 操作使用存储的模板或内联模板发送电子邮件：

- 该`SendEmail`操作对于向单个目标对象发送自定义电子邮件非常有用。v2 API `Destination`对象可以包含ToAddressesCcAddresses、和BccAddresses属性。它们可以任意组合使用，并且可以包含一个或多个将接收相同电子邮件的电子邮件地址。

- 该SendBulkEmail操作对于通过对 v2 API 的单次调用向多个目标对象发送唯一的电子邮件非常有用。

本节提供了如何通过这两个 AWS CLI 发送操作使用发送模板化电子邮件的示例。

向单个目标对象发送模板化电子邮件

您可以使用该SendEmail操作向在单个目标对象中定义的一个或多个收件人发送电子邮件。Destination 对象中的所有收件人都会收到同一电子邮件。

向单个目标对象发送模板化电子邮件

1. 根据您要使用存储的模板还是内联模板，选择要粘贴到文本编辑器中的相应代码示例，并根据需要对其进行自定义。

   Stored template code example

   请注意，您在上一步中创建的模板被引用为TemplateName参数的值。MyTemplate

```
{
    "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"
}
```

   此代码包含以下属性：

   - FromEmailAddress— 发件人的电子邮件地址。

- 目标-包含在ToAddresses、CcAddresses和BccAddresses属性中定义的电子邮件收件人的对象。它们可以任意组合使用，并且可以包含一个或多个将接收相同电子邮件的电子邮件地址。
- TemplateName— 要应用于电子邮件的Template资源名称。
- TemplateData— 包含键值对的转义的 JSON 字符串。这些键对应于存储模板的TemplateContent属性中定义的变量，例如{{name}}。这些值表示替换变量的内容。
- ConfigurationSetName— 发送电子邮件时要使用的配置集的名称。

> **Note**
>
> 我们建议您使用配置为将呈现失败事件发布到 Amazon SNS 的配置集。有关更多信息，请参阅 the section called “（可选）第 1 部分：设置通知”。

Inline template code example

请注意，TemplateContent属性（通常在存储的模板中定义）是与使其成为内联模板的*TemplateData*属性一起内联定义的。

```
{
    "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"
```

```
}
```

此代码包含以下属性：

- FromEmailAddress— 发件人的电子邮件地址。
- 目标-包含在ToAddresses、CcAddresses和BccAddresses属性中定义的电子邮件收件人的对象。它们可以任意组合使用，并且可以包含一个或多个将接收相同电子邮件的电子邮件地址。
- TemplateContent— 包含以下属性的容器：
  - SubjectPart – 电子邮件的主题行。此属性可能包含替换标签。这些标签使用以下格式：`{{tagname}}`。当您发送电子邮件时，您可以为每个目标的 `tagname` 指定一个值。
  - HtmlPart— 电子邮件的 HTML 正文。此属性可能包含替换标签。前面的示例包含两个标签：`{{name}}` 和 `{{favoriteanimal}}`。
  - TextPart— 电子邮件的正文。电子邮件客户端未显示 HTML 内容的收件人将看到此版本的电子邮件。此属性也可能包含替换标签。
- TemplateData— 包含键值对的转义的 JSON 字符串。这些键对应于此文件中`TemplateContent`属性中定义的变量，例如`{{name}}`。这些值表示替换变量的内容。
- ConfigurationSetName— 发送电子邮件时要使用的配置集的名称。

> **Note**
>
> 我们建议您使用配置为将呈现失败事件发布到 Amazon SNS 的配置集。有关更多信息，请参阅 the section called “（可选）第 1 部分：设置通知”。

2. 自定义前面的示例以满足您的需求，然后将该文件另存为 *myemail.json*。
3. 在命令行中，键入以下 v2 API 命令以发送电子邮件：

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

向多个目标对象发送模板化电子邮件

您只需调用 SES v2 API，即可使用该`SendBulkEmail`操作向多个目标对象发送电子邮件。SES 向每个`Destination`对象中的一个或多个收件人发送一封唯一的电子邮件。

向多个目标对象发送模板化电子邮件

1. 根据您要使用存储的模板还是内联模板，选择要粘贴到文本编辑器中的相应代码示例，并根据需要对其进行自定义。

   Stored template code example

   请注意，您在上一步中创建的模板被引用为TemplateName参数的值。MyTemplate

```
{
    "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"
}
```

此代码包含以下属性：

- FromEmailAddress— 发件人的电子邮件地址。
- DefaultContent— 包含TemplateName和对象的 JSON TemplateData 对象。
- TemplateName— 要应用于电子邮件的Template资源名称。
- TemplateData— 包含键值对，如果对象在属性中包含空的 JSON ReplacementEmailContent 对象{}，则将使用这些ReplacementTemplateData键值对。
- BulkEmailEntries— 包含一个或多个Destination对象的数组。

- 目标-包含在ToAddresses、CcAddresses和BccAddresses属性中定义的电子邮件收件人的对象。它们可以任意组合使用，并且可以包含一个或多个将接收相同电子邮件的电子邮件地址。
- ReplacementTemplateData— 包含键值对的转义的 JSON 字符串。例如，这些键对应于模板中的变量{{name}}。值表示用来替换电子邮件中的变量的内容。（如果此处的 JSON 字符串为空（由表示）{}，则将使用DefaultContent对象中TemplateData属性中定义的键值对。）
- ConfigurationSetName— 发送电子邮件时要使用的配置集的名称。

> **Note**
>
> 我们建议您使用配置为将呈现失败事件发布到 Amazon SNS 的配置集。有关更多信息，请参阅 the section called “（可选）第 1 部分：设置通知”。

Inline template code example

请注意，TemplateContent属性（通常在存储的模板中定义）是与使其成为内联模板的*TemplateData*属性一起内联定义的。

```
{
    "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"
}
```

此代码包含以下属性：

- FromEmailAddress— 发件人的电子邮件地址。
- DefaultContent— 包含TemplateContent和对象的 JSON TemplateData 对象。
- TemplateContent— 包含以下属性的容器：
  - SubjectPart – 电子邮件的主题行。此属性可能包含替换标签。这些标签使用以下格式：{{tagname}}。当您发送电子邮件时，您可以为每个目标的 tagname 指定一个值。
  - HtmlPart— 电子邮件的 HTML 正文。此属性可能包含替换标签。前面的示例包含两个标签：{{name}} 和 {{favoriteanimal}}。
  - TextPart— 电子邮件的正文。电子邮件客户端未显示 HTML 内容的收件人将看到此版本的电子邮件。此属性也可能包含替换标签。
- TemplateData— 包含键值对，如果对象在属性中包含空的 JSON ReplacementEmailContent 对象{}，则将使用这些ReplacementTemplateData键值对。
- BulkEmailEntries— 包含一个或多个Destination对象的数组。
- 目标-包含在ToAddresses、CcAddresses和BccAddresses属性中定义的电子邮件收件人的对象。它们可以任意组合使用，并且可以包含一个或多个将接收相同电子邮件的电子邮件地址。
- ReplacementTemplateData— 包含键值对的转义的 JSON 字符串。这些键对应于此文件中TemplateContent属性中定义的变量，例如{{name}}。值表示用来替换电子邮件中的变量的内容。（如果此处的 JSON 字符串为空（由表示）{}，则将使用DefaultContent对象中TemplateData属性中定义的键值对。）
- ConfigurationSetName— 发送电子邮件时要使用的配置集的名称。

> **Note**
>
> 我们建议您使用配置为将呈现失败事件发布到 Amazon SNS 的配置集。有关更多信息，请参阅 the section called “（可选）第 1 部分：设置通知”。

2. 更改上一步骤代码中的值以满足您的需求，然后将该文件另存为 *mybulkemail.json*。
3. 在命令行中，键入以下 v2 API 命令以发送批量电子邮件：

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

高级电子邮件个性化

如果您使用的是存储的模板，也就是说，您已通过使用 SES v2 API 的`CreateEmailTemplate`操作在 Amazon SES 中创建了`Template`资源，则可以利用 Handlebars 系统创建包含高级功能的模板，例如嵌套属性、数组迭代、基本条件语句和内联部分的创建。本部分提供有关这些特征的一些示例。

除本部分介绍的特征以外，Handlebars 还包含许多其他特征。有关更多信息，请参阅handlebarsjs.com 上的 Built-In Helpers。

> **Note**
>
> 为消息呈现 HTML 模板时，SES 不会转义 HTML 内容。这意味着，如果您包括用户输入的数据，例如联系人表单的用户输入数据，则需要在客户端将其转义。

主题

- 解析嵌套属性
- 遍历列表
- 使用基本条件语句
- 创建内联部分

解析嵌套属性

Handlebars 包括对嵌套路径的支持，这让您能够轻松组织复杂的客户数据，然后在电子邮件模板中引用这些数据。

例如，您可以将收件人数据组织到多个常规类别中。在每个类别中，您可以包含详细信息。以下代码示例显示了包含单一收件人的此种结构：

```
{
  "meta":{
    "userId":"51806220607"
```

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

在电子邮件模板中，您可以通过以下方式引用嵌套属性：提供父属性名称，后跟句点 (.)，然后是要包含其值的属性的名称。例如，对于上例中显示的数据结构，要在电子邮件模板中包含每个收件人的名字，请在电子邮件模板中包含以下文本：`Hello {{contact.firstName}}!`

Handlebars 能够解析多层嵌套路径，这意味着您可以灵活地组织模板数据结构。

遍历列表

`each` 帮助程序函数可遍历数组中的项目。以下代码是一个电子邮件模板的示例，此模板使用 `each` 帮助程序函数创建每个收件人兴趣的明细列表。

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

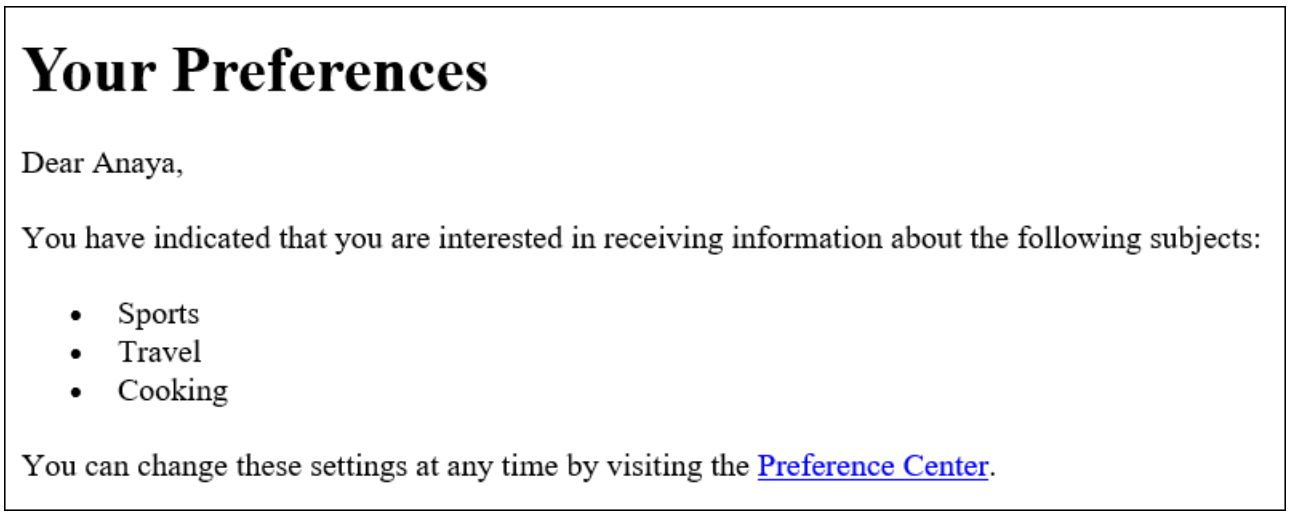
> **Important**
>
> 在前面的代码示例中，HtmlPart 和 TextPart 属性的值包含换行符，以便提高示例的可读性。您的模板的 JSON 文件不能在这些值中包含换行符。如果您将此示例复制并粘贴到自己的 JSON 文件中，请在继续前从 HtmlPart 和 TextPart 部分中删除换行符和多余空格。

创建模板后，您可以使用 SendEmail 或 SendBulkEmail 操作使用此模板向收件人发送电子邮件。只要每个收件人在 Interests 对象中至少有一个值，他们就会收到包含其兴趣明细列表的电子邮件。以下示例显示了可用于使用上述模板向多个收件人发送电子邮件的 JSON 文件：

```
{
  "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\":[]}"
}
```

当您使用 SendBulkEmail 操作向前面示例中列出的收件人发送电子邮件时，他们会收到类似下图所示示例的邮件：



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.

使用基本条件语句

本部分基于前一部分中介绍的示例。前一部分中的示例使用 each 帮助程序遍历兴趣列表。但是，未指定兴趣的收件人会收到包含空列表的电子邮件。通过使用 {{if}} 帮助程序，您可根据模板数据中是否存在特定属性以不同方式设置电子邮件格式。以下代码使用了 {{if}} 帮助程序：如果 Subscription 数组包含任何值，则显示前一部分中的项目符号列表。如果数组为空，则显示其他的文本块。

```
{
  "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**
>
> 在前面的代码示例中，HtmlPart 和 TextPart 属性的值包含换行符，以便提高示例的可读性。您的模板的 JSON 文件不能在这些值中包含换行符。如果您将此示例复制并粘贴到自己的 JSON 文件中，请在继续前从 HtmlPart 和 TextPart 部分中删除换行符和多余空格。

以下示例显示了可用于使用上述模板向多个收件人发送电子邮件的 JSON 文件：

```
{
  "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\":[]}"
}
```

在此示例中，模板数据包含兴趣列表的收件人会收到与前一部分中所示示例相同的电子邮件。模板数据不包含任何兴趣的收件人会收到类似下图所示示例的电子邮件：

创建内联部分

您可以使用内联部分简化包含重复字符串的模板。例如，您可以在模板开头添加以下代码，从而创建一个内联部分，其中包含收件人的名字和姓氏 (如果可用)：

```
{{#* inline \"fullName\"}}{{firstName}}{{#if lastName}} {{lastName}}{{/if}}{{/
inline}}\n
```

> **Note**
>
> 需要使用换行符 (\n) 将 {{inline}} 块与模板内容分开。最终输出中不显示换行符。

创建 fullName 部分后，您可以通过在此部分的名称前加上一个大于号 (>) 并后跟一个空格来将其包含在模板中的任何位置，如下例所示：{{> fullName}}。内联部分不会在电子邮件部分间转移。例如，要在电子邮件的 HTML 和文本版本中使用相同的内联部分，则必须在 HtmlPart 和 TextPart 部分中都定义此部分。

遍历数组时，也可以使用内联部分。您可以使用以下代码创建使用 fullName 内联部分的模板。在此示例中，内联部分应用至收件人姓名和一个其他名称数组：

```
{
  "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**
>
> 在前面的代码示例中，HtmlPart 和 TextPart 属性的值包含换行符，以便提高示例的可读性。您的模板的 JSON 文件不能在这些值中包含换行符。如果您将此示例复制并粘贴到自己的 JSON 文件中，请从这些部分中删除换行符和多余空格。

管理电子邮件模板

除了创建电子邮件模板外，您还可以使用 Amazon SES v2 API 更新或删除现有模板、列出所有现有模板或查看模板内容。

本节包含使用执行 AWS CLI 与 SES 模板相关的任务的过程。

> **Note**
>
> 本节中的过程假定您已安装和配置 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

查看电子邮件模板列表

您可以使用 ListEmailTemplateSES v2 API 操作来查看所有现有电子邮件模板的列表。

查看电子邮件模板列表

- 在命令行输入以下命令：

```
aws sesv2 list-email-templates
```

如果当前区域中您的 SES 账户中存在现有电子邮件模板，则此命令将返回类似于以下示例的响应：

```
{
    "TemplatesMetadata": [
        {
            "Name": "SpecialOffers",
            "CreatedTimestamp": "2020-08-05T16:04:12.640Z"
        },
        {
            "Name": "NewsAndUpdates",
            "CreatedTimestamp": "2019-10-03T20:03:34.574Z"
        }
    ]
}
```

如果您尚未创建任何模板，那么该命令会返回没有任何成员的 TemplatesMetadata 对象。

查看特定电子邮件模板的内容

您可以使用 GetEmailTemplateSES v2 API 操作来查看特定电子邮件模板的内容。

查看电子邮件模板的内容

- 在命令行输入以下命令：

```
aws sesv2 get-email-template --template-name MyTemplate
```

在前面的命令中，*MyTemplate*替换为要查看的模板的名称。

如果您提供的模板名称与您的 SES 账户中存在的模板相匹配，则此命令将返回类似于以下示例的响应：

```
{
    "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>"
```

```
    }
}
```

如果您提供的模板名称与您的 SES 账户中存在的模板不匹配，则该命令将返回NotFoundException错误。

删除电子邮件模板

您可以使用 DeleteEmailTemplateSES v2 API 操作来删除特定的电子邮件模板。

删除电子邮件模板

- 在命令行输入以下命令：

```
aws sesv2 delete-email-template --template-name MyTemplate
```

在前面的命令中，*MyTemplate*替换为要删除的模板的名称。

此命令不提供任何输出。您可以使用GetTemplate操作验证模板是否已删除。

更新电子邮件模板

您可以使用 UpdateEmailTemplateSES v2 API 操作来更新现有的电子邮件模板。例如，如果您要更改电子邮件模板的主题行，或者如果您需要修改邮件本身的正文，那么此操作很有用。

更新电子邮件模板

1. 使用 GetEmailTemplate 命令，在命令行中输入以下命令来检索现有模板：

```
aws sesv2 get-email-template --template-name MyTemplate
```

在前面的命令中，*MyTemplate*替换为要更新的模板的名称。

如果您提供的模板名称与您的 SES 账户中存在的模板相匹配，则此命令将返回类似于以下示例的响应：

```
{
    "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. 在文本编辑器中，创建一个新文件。将上一个命令的输出粘贴到文件中。
3. 根据需要修改模板。您省略的任何行都将从模板中删除。例如，如果您只想更改模板的 SubjectPart，您仍然需要包含 TextPart 和 HtmlPart 属性。

   完成后，将文件另存为 update_template.json。
4. 在命令行输入以下命令：

   ```
   aws sesv2 update-email-template --cli-input-json file://path/to/
   update_template.json
   ```

   在前面的命令中，*path/to/update_template.json*替换为您在上一步中创建的update_template.json文件的路径。

   如果模板更新成功，则此命令不提供任何输出。您可以使用GetEmailTemplate操作来验证模板是否已更新。

   如果您指定的模板不存在，则此命令会返回 TemplateDoesNotExist 错误。如果模板不包含 TextPart 和/或 HtmlPart 属性，则此命令会返回 InvalidParameterValue 错误。

使用软件开发工具包通过 Amazon AWS SES 发送电子邮件

您可以使用 AWS 软件开发工具包通过 Amazon SES 发送电子邮件。 AWS SDKs 适用于多种编程语言。有关更多信息，请参阅用于 Amazon Web Services 的工具。

先决条件

要完成下一节中的任何代码示例，必须完成以下先决条件：

- 如果您尚未执行此操作，请完成设置 Amazon Simple Email Service中的任务。
- 使用 Amazon SES 验证您的电子邮件地址 – 您必须先验证您拥有发件人的电子邮件地址，然后才能使用 Amazon SES 发送电子邮件。如果您的账户仍在 Amazon SES 沙盒中，您还必须验证收件人的电子邮件地址。我们建议您使用 Amazon SES 控制台来验证电子邮件地址。有关更多信息，请参阅创建电子邮件地址身份。

- 获取您的 AWS 证书-您需要访问密钥 ID 和 AWS 私有 AWS 访问密钥才能使用软件开发工具包访问 Amazon SES。您可以使用 中的安全凭证 AWS Management Console页面来查找您的凭证。有关凭证的更多信息，请参阅Amazon SES 凭证的类型。
- 创建共享凭证文件 – 为了使此部分中的示例代码正常运行，您必须创建一个共享凭证文件。有关更多信息，请参阅 创建共享凭证文件，以便在使用软件开发工具包通过 Amazon AWS SES 发送电子邮件时使用。

代码示例

> **Important**
>
> 在以下教程中，您将向自己发送电子邮件，以便检查是否收到了该电子邮件。如需进一步试验或进行负载测试，请使用 Amazon SES 邮箱模拟器。您发送到邮箱模拟器的电子邮件不会计入您的发送配额或您的退信率和投诉率。有关更多信息，请参阅 手动使用邮箱模拟器。

.NET

以下过程介绍如何使用 Visual Studio 和 适用于 .NET 的 AWS SDK通过 Amazon SES 发送电子邮件。

已使用以下组件测试此解决方案：

- Microsoft Visual Studio Community 2017 版本 15.4.0。
- Microsoft .NET Framework 版本 4.6.1。
- AWSSDK.Core 软件包（版本 3.3.19），使用安装。 NuGet
- 的 AWSSDK。 SimpleEmail 软件包（版本 3.3.6.1），使用安装。 NuGet

在开始前，请执行以下任务：

- 安装 Visual Studio —Visual Studio 可在 https://www.visualstudio.com/上

要使用发送电子邮件 适用于 .NET 的 AWS SDK

1. 通过执行以下步骤创建新项目：

   a. 启动 Visual Studio。

b. 在 File 菜单上，依次选择 New 和 Project。

c. 在 New Project 窗口上的左侧面板中，展开 Installed，然后展开 Visual C#。

d. 在右侧面板中，选择 Console App (.NET Framework)。

e. 对于名称，键入 **AmazonSESSample**，然后选择 确定。

2. 通过完成以下步骤，使用 NuGet 将 Amazon SES 软件包包含在您的解决方案中：

a. 在 “解决方案资源管理器” 窗格中，右键单击您的项目，然后选择 “管理 NuGet 包”。

b. 在 NuGet：Amazon SESSample 选项卡上，选择浏览。

c. 在搜索框中，键入 **AWSSDK.SimpleEmail**。

d. 选择AWSSDK。 SimpleEmail软件包，然后选择 “安装”。

e. 在 Preview Changes 窗口中，选择 OK。

3. 在 Program.cs 选项卡上，粘贴以下代码：

```
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 (### .NET # AWS SDK)";
```

```
        // 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 ### .NET # AWS SDK.";

        // The HTML body of the email.
        static readonly string htmlBody = @"<html>
<head></head>
<body>
  <h1>Amazon SES Test (### .NET # SDK)</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/'> ### .NET # AWS SDK</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. 在代码编辑器中，执行下列操作：

   - *sender@example.com*替换为 “发件人:” 电子邮件地址。必须验证此地址。有关更多信息，请参阅 已验证的身份。
   - *recipient@example.com*替换为 “收件人：” 地址。如果您的账户仍处于沙盒中，则还必须验证此地址。
   - *ConfigSet*替换为发送此电子邮件时要使用的配置集的名称。
   - *USWest2*替换为您用于通过 Amazon SES 发送电子邮件的 AWS 区域 终端节点的名称。有关已推出 Amazon SES 的区域的列表，请参阅《AWS 一般参考》中的 Amazon Simple Email Service（Amazon SES）。

完成后，保存 `Program.cs`。

5. 通过完成以下步骤来生成并运行应用程序：

   a. 在 Build 菜单上，选择 Build Solution。

   b. 在 Debug 菜单上，选择 Start Debugging。此时显示一个控制台窗口。

6. 检查控制台的输出。如果已成功发送电子邮件，则控制台会显示“`The email was sent successfully.`”

7. 如果已成功发送电子邮件，请登录收件人地址的电子邮件客户端。您将看到已发送的电子邮件。

Java

以下过程向您展示了如何使用适用于 Java EE 开发人员的 Eclipse IDE、AWS Toolkit for Eclipse如何创建软件开发 AWS 工具包项目和修改 Java 代码以通过 Amazon SES 发送电子邮件。

在开始前，请执行以下任务：

- 安装 Eclipse – 访问 https://www.eclipse.org/downloads 可获得 Eclipse。本教程中的代码使用 Eclipse Neon.3 (版本 4.6.3) 和 Java 运行时环境的 1.8 版本进行了测试。
- 安装 AWS Toolkit for Eclipse —有关向 Eclipse 安装中 AWS Toolkit for Eclipse 添加的说明，请访问 /eclipse。https://aws.amazon.com本教程中的代码已使用 2.3.1 版本的 AWS Toolkit for Eclipse进行了测试。

要使用发送电子邮件 适用于 Java 的 AWS SDK

1. 通过执行以下步骤在 Eclipse 中创建 AWS Java 项目：

   a. 启动 Eclipse。

   b. 在 File 菜单上，选择 New，然后选择 Other。在 New 窗口中，展开 AWS 文件夹，然后选择 AWS Java Project。

   c. 在 “新建 AWS Java 项目” 对话框中，执行以下操作：

      i. 对于 Project name，键入项目的名称。

      ii. 在 “适用于 Java 的 AWS SDK 示例” 下，选择 Amazon 简单电子邮件服务 JavaMail 示例。

iii. 选择完成。

2. 在 Eclipse 中的 Package Explorer 窗格中，展开您的项目。
3. 在您的项目下，展开 src/main/java 文件夹，展开 com.amazon.aws.samples 文件夹，然后双击 AmazonSESSample.java。
4. 将 AmazonSESSample.java 的整个内容替换为以下代码：

```
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 (### Java # AWS SDK)";

  // The HTML body for the email.
  static final String HTMLBODY = "<h1>Amazon SES test (### Java # AWS SDK)</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 ### Java # AWS SDK.";

  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. 在 AmazonSESSample.java 中，将以下内容替换为您自己的值：

> **Important**
>
> 电子邮件地址区分大小写。请确保此处的地址与经验证的地址完全相同。

- `SENDER@EXAMPLE.COM` – 替换为您的 From (发件人) 电子邮件地址。运行此程序之前，您必须验证该地址。有关更多信息，请参阅 Amazon SES 中已验证的身份。
- `RECIPIENT@EXAMPLE.COM` – 替换为您的 To (收件人) 电子邮件地址。如果您的账户仍处于沙盒中，您还必须验证此地址，然后才能使用它。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。
- （可选）**`us-west-2`** - 如果您要在美国西部（俄勒冈州）以外的区域中使用 Amazon SES，请将它替换为您要使用的区域。有关已推出 Amazon SES 的区域的列表，请参阅《AWS 一般参考》中的 Amazon Simple Email Service（Amazon SES）。

6. 保存 `AmazonSESSample.java`。
7. 要构建项目，请选择 Project，然后选择 Build Project。

> **Note**
>
> 如果禁用此选项，则可能启用自动构建；如果是这样，请跳过此步骤。

8. 要开始程序和发送电子邮件，请选择 Run，然后再次选择 Run。
9. 在 Eclipse 中查看控制台窗格的输出。如果已成功发送电子邮件，则控制台会显示“`Email sent!`”，否则将显示一条错误消息。
10. 如果已成功发送电子邮件，请登录收件人地址的电子邮件客户端。您将看到已发送的电子邮件。

PHP

本主题说明如何使用 适用于 PHP 的 AWS SDK 通过 Amazon SES 发送电子邮件。

在开始前，请执行以下任务：

- 安装 PHP – 访问 http://php.net/downloads.php 可获得 PHP。本教程需要 PHP 版本 5.5 或更高版本。安装 PHP 后，在环境变量中添加 PHP 的路径，这样就能通过任何命令提示符运行 PHP。本教程中的代码已使用 PHP 7.2.7 进行测试。
- 安装 适用于 PHP 的 AWS SDK 版本 3-有关下载和安装说明，请参阅适用于 PHP 的 AWS SDK 文档。本教程中的代码已使用版本 3.64.13 的软件开发工具包进行测试。

要通过 Amazon SES 发送电子邮件，请使用 适用于 PHP 的 AWS SDK

1. 在文本编辑器中，创建一个名为 amazon-ses-sample.php 的文件。粘贴以下代码：

```
<?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 (### PHP # AWS SDK)';
$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/">'.
                '### PHP # AWS SDK</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. 在 amazon-ses-sample.php 中，将以下内容替换为您自己的值：

- **path_to_sdk_inclusion**—替换为包含在程序 适用于 PHP 的 AWS SDK 中所需的路径。有关更多信息，请参阅 适用于 PHP 的 AWS SDK 文档。
- **sender@example.com** – 替换为您已使用 Amazon SES 验证过的电子邮件地址。有关更多信息，请参阅 已验证的身份。Amazon SES 中的电子邮件地址区分大小写。请确保您输入的地址与经验证的地址完全相同。
- **recipient1@example.com**，**recipient2@example.com**— 替换为收件人的地址。如果您的账户仍处于沙盒中，则还必须验证收件人地址。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。请确保您输入的地址与经验证的地址完全相同。
- （可选）**ConfigSet** – 如果您要在发送此电子邮件时使用配置集，请将此值替换为配置集的名称。有关配置集的更多信息，请参阅在 SES 中使用配置集。
- （可选）**us-west-2** - 如果您要在美国西部（俄勒冈州）以外的区域中使用 Amazon SES，请将它替换为您要使用的区域。有关已推出 Amazon SES 的区域的列表，请参阅《AWS 一般参考》中的 Amazon Simple Email Service（Amazon SES）。

3. 保存 amazon-ses-sample.php。
4. 要运行程序，请在 amazon-ses-sample.php 所在的同一目录中打开命令提示符，然后键入以下命令：

```
$ php amazon-ses-sample.php
```

5. 检查输出。如果已成功发送电子邮件，则控制台会显示“Email sent!”，否则将显示一条错误消息。

> **Note**
>
> 如果在运行程序时遇到“cURL error 60: SSL certificate problem”错误，请下载最新的 CA 服务包，如适用于 PHP 的 AWS SDK文档中所述。然后，在 amazon-ses-sample.php 中，将以下行添加到 SesClient::factory 数组，将 path_of_certs 替换为您下载的 CA 捆绑的路径，然后重新运行程序。
>
> ```
> 'http' => [
>    'verify' => 'path_of_certs\ca-bundle.crt'
> ]
> ```

6. 登录收件人地址的电子邮件客户端。您将看到已发送的电子邮件。

Ruby

本主题说明如何使用 适用于 Ruby 的 AWS SDK 通过 Amazon SES 发送电子邮件。

在开始前，请执行以下任务：

- 安装 Ruby —Ruby 的网址为 https://www.ruby-lang。 org/en/downloads/。本教程中的代码已使用 Ruby 1.9.3 进行测试。安装 Ruby 后，在环境变量中添加 Ruby 的路径，这样就能通过任何命令提示符运行 Ruby。
- 安装 适用于 Ruby 的 AWS SDK —有关下载和安装说明，请参阅《适用于 Ruby 的 AWS SDK 开发人员指南》 适用于 Ruby 的 AWS SDK中的安装。本教程中的示例代码已使用 2.9.36 版本的适用于 Ruby 的 AWS SDK进行了测试。
- 创建共享凭证文件 – 为了使此部分中的示例代码正常运行，您必须创建一个共享凭证文件。有关更多信息，请参阅 创建共享凭证文件，以便在使用软件开发工具包通过 Amazon AWS SES 发送电子邮件时使用。

要通过 Amazon SES 发送电子邮件，请使用 适用于 Ruby 的 AWS SDK

1. 在文本编辑器中，创建一个名为 `amazon-ses-sample.rb` 的文件。将以下代码粘贴到该文件中：

```
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 (### Ruby # AWS SDK)"
```

```
# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (### Ruby # AWS SDK)</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/">'\
  '### Ruby # AWS SDK</a>.'

# The email body for recipients with non-HTML email clients.
textbody = "This email was sent with Amazon SES using the ### Ruby # AWS SDK."

# 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. 在 amazon-ses-sample.rb 中，将以下内容替换为您自己的值：

   - **sender@example.com** – 替换为您已使用 Amazon SES 验证过的电子邮件地址。有关更多信息，请参阅 已验证的身份。Amazon SES 中的电子邮件地址区分大小写。请确保您输入的地址与经验证的地址完全相同。
   - **recipient@example.com** – 替换为收件人的地址。如果您的账户仍处于沙盒中，您还必须验证此地址，然后才能使用它。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。请确保您输入的地址与经验证的地址完全相同。
   - （可选）**us-west-2** - 如果您要在美国西部（俄勒冈州）以外的区域中使用 Amazon SES，请将它替换为您要使用的区域。有关已推出 Amazon SES 的区域的列表，请参阅《AWS 一般参考》中的 Amazon Simple Email Service（Amazon SES）。

3. 保存 amazon-ses-sample.rb。
4. 要运行程序，请在 amazon-ses-sample.rb 所在的目录中打开命令提示符，然后键入 ruby amazon-ses-sample.rb
5. 检查输出。如果已成功发送电子邮件，则控制台会显示“Email sent!”，否则将显示一条错误消息。
6. 登录收件人地址的电子邮件客户端。您将找到已发送的电子邮件。

Python

本主题说明如何使用 AWS SDK for Python (Boto) 通过 Amazon SES 发送电子邮件。

在开始前，请执行以下任务：

- 使用 Amazon SES 验证您的电子邮件地址 – 您必须先验证您拥有发件人的电子邮件地址，然后才能使用 Amazon SES 发送电子邮件。如果您的账户仍在 Amazon SES 沙盒中，您还必须验证

收件人的电子邮件地址。我们建议您使用 Amazon SES 控制台来验证电子邮件地址。有关更多信息，请参阅 创建电子邮件地址身份。

- 获取您的 AWS 证书-您需要访问密钥 ID 和 AWS 私有 AWS 访问密钥才能使用软件开发工具包访问 Amazon SES。您可以通过 的安全凭证 AWS Management Console页面来查找您的凭证。有关凭证的更多信息，请参阅Amazon SES 凭证的类型。
- 安装 Python —Python 已在 thon.org/downloads/ 上https://www.py线。本教程中的代码已使用 Python 2.7.6 和 Python 3.6.1 进行了测试。安装 Python 后，在环境变量中添加 Python 的路径，这样就能通过任何命令提示符运行 Python。
- 安装 AWS SDK for Python (Boto)—有关下载和安装说明，请参阅AWS SDK for Python (Boto) 文档。本教程中的示例代码已使用 1.4.4 版本的 SDK for Python 进行了测试。

使用 SDK for Python 通过 Amazon SES 发送电子邮件

1. 在文本编辑器中，创建一个名为 `amazon-ses-sample.py` 的文件。将以下代码粘贴到该文件中：

```
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. 在 amazon-ses-sample.py 中，将以下内容替换为您自己的值：

   - **sender@example.com** – 替换为您已使用 Amazon SES 验证过的电子邮件地址。有关更多信息，请参阅 已验证的身份。Amazon SES 中的电子邮件地址区分大小写。请确保您输入的地址与经验证的地址完全相同。
   - **recipient@example.com** – 替换为收件人的地址。如果您的账户仍处于沙盒中，您还必须验证此地址，然后才能使用它。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。请确保您输入的地址与经验证的地址完全相同。
   - （可选）**us-west-2** - 如果您要在美国西部（俄勒冈州）以外的区域中使用 Amazon SES，请将它替换为您要使用的区域。有关已推出 Amazon SES 的区域的列表，请参阅《AWS 一般参考》中的 Amazon Simple Email Service（Amazon SES）。

3. 保存 amazon-ses-sample.py。
4. 要运行程序，请在 amazon-ses-sample.py 所在的目录中打开命令提示符，然后键入 python amazon-ses-sample.py。
5. 检查输出。如果已成功发送电子邮件，则控制台会显示“Email sent!”，否则将显示一条错误消息。
6. 登录收件人地址的电子邮件客户端。您将看到已发送的电子邮件。

创建共享凭证文件，以便在使用软件开发工具包通过 Amazon AWS SES 发送电子邮件时使用

以下过程演示如何在主目录中创建一个共享凭证文件。若要让软件开发工具包示例代码正常运行，您必须创建此文件。

1. 在文本编辑器中，创建一个新文件。在此文件中，粘贴以下代码：

```
[default]
aws_access_key_id = YOUR_AWS_ACCESS_KEY_ID
aws_secret_access_key = YOUR_AWS_SECRET_ACCESS_KEY
```

2. 在您刚刚创建的文本文件中，YOUR_AWS_ACCESS_KEY替换为您唯一的 AWS 访问密钥 ID，然后YOUR_AWS_SECRET_ACCESS_KEY替换为您唯一的私有访问 AWS 密钥。
3. 保存该文件。下表显示了您的操作系统的正确位置和文件名。

如果您使用的是……	将文件另存为…
Windows	`C:\Users\<yourUserName>\.aws\credentials`
Linux、macOS 或 Unix	`~/.aws/credentials`

> **Important**
>
> 请勿在保存凭证文件时包含文件扩展名。

Amazon SES 支持的内容编码

提供以下内容供参考。

Amazon SES 支持以下内容编码：

- `deflate`
- `gzip`
- `identity`

根据 RFC 7231 规范，Amazon SES 还支持以下 Accept-Encoding 标头格式：

- `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 和安全协议

本主题介绍当您连接到 Amazon SES 时以及当 SES 将电子邮件传递给接收方时，您可以使用的安全协议。

电子邮件发件人到 Amazon SES

您用于连接到 Amazon SES 的安全协议取决于您使用的是 Amazon SES API 还是 Amazon SES SMTP 接口，如下所述。

HTTPS

如果您使用的是 Amazon SES API（直接使用或通过 AWS 软件开发工具包），则所有通信都将通过 Amazon SES HTTPS 终端节点通过 TLS 加密。Amazon SES HTTPS 端点支持 TLS 1.2 和 TLS 1.3。

SMTP 接口

如果通过 SMTP 接口访问 Amazon SES，那么您需要使用传输层安全性（TLS）来加密您的连接。请注意，提及 TLS 时通常使用其前身协议的名称：安全套接字层 (SSL)。

Amazon SES 支持两种建立 TLS 加密连接的机制：STARTTLS 和 TLS Wrapper。

- STARTTLS – STARTTLS 是一种将未加密的连接升级到加密连接的方式。提供了适用于各种协议的 STARTTLS 版本；SMTP 版本已在 RFC 3207 中定义。对于 STARTTLS 连接，Amazon SES 支持 TLS 1.2 和 TLS 1.3。
- TLS Wrapper – TLS Wrapper（也称为 SMTPS 或握手协议）是一种在无需先建立未加密连接的情况下启动加密连接的方式。利用 TLS Wrapper，Amazon SES SMTP 端点不执行 TLS 协商：客户端负责使用 TLS 连接到端点，然后继续对整个对话使用 TLS。虽然 TLS Wrapper 是一项旧协议，但许多客户端仍支持它。对于 TLS 包装器连接，Amazon SES 支持 TLS 1.2 和 TLS 1.3。

有关使用这些方法来连接到 Amazon SES SMTP 接口的信息，请参阅连接到 Amazon SES SMTP 端点。

Amazon SES 到接收方

虽然 TLS 1.3 是我们的默认传送方式，但 SES 可以使用早期版本的 TLS 将电子邮件传送到邮件服务器。

默认情况下，Amazon SES 使用操作 TLS。这意味着 Amazon SES 始终尝试与接收邮件服务器建立安全连接。如果 Amazon SES 无法建立安全连接，那么它会发送未加密的邮件。

您可以使用配置集来更改此行为。使用 PutConfigurationSetDeliveryOptionsAPI 操作将配置的`TlsPolicy`属性设置为`Require`。您可以使用 AWS CLI 来实施此更改。

配置 Amazon SES 以要求配置集的 TLS 连接

- 在命令行输入以下命令：

  ```
  aws sesv2 put-configuration-set-delivery-options --configuration-set-
  name MyConfigurationSet --tls-policy REQUIRE
  ```

  在前面的示例中，*MyConfigurationSet*替换为配置集的名称。

  如果您使用此配置集来发送电子邮件，那么只有当 Amazon SES 能够建立安全连接时，它才会将邮件发送到接收电子邮件服务器。如果 Amazon SES 无法与接收电子邮件服务器建立安全连接，那么它会丢弃该邮件。

End-to-end 加密

您可以使用 Amazon SES 发送使用 S/MIME 或 PGP 加密的消息。使用这些协议的消息会被发送人加密。只有拥有解密消息所需私有密钥的收件人才能查看其内容。

Amazon SES 支持以下 MIME 类型，您可以使用这些类型来发送 S/MIME 加密的电子邮件：

- `application/pkcs7-mime`
- `application/pkcs7-signature`
- `application/x-pkcs7-mime`
- `application/x-pkcs7-signature`

Amazon SES 还支持以下 MIME 类型，您可以使用它们来发送通过 PGP 加密的电子邮件：

- application/pgp-encrypted
- application/pgp-keys
- application/pgp-signature

Amazon SES 标头字段

Amazon SES 可接受采用 RFC 822 中所述格式的所有电子邮件标头。

以下字段不能在消息的标头部分中出现多次：

- 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

注意事项

- acceptLanguage 字段为非标准字段。如果可能，您应该使用 Accept-Language 标头。
- 如果您指定 Date 标头，则 Amazon SES 会使用时间戳将其覆盖，该时间戳对应于 SES 接受消息时 UTC 时区中的日期和时间。
- 如果您提供 Message-ID 标头，则 Amazon SES 会用自己的值来覆盖标头。
- 如果您指定 Return-Path 标头，则 Amazon SES 会将退信和投诉通知发送到您指定的地址。但是，收件人收到的消息包含不同的 Return-Path 标头值。
- 如果您使用 Amazon SES API v2 SendEmail 操作，并且指定了简单或模板化内容类型，或者使用 SendBulkEmail 操作，则无法为 SES 设置的标头设置自定义标题内容；因此，不允许将以下标头作为自定义标头：
  - BCC, CC, Content-Disposition, Content-Type, Date, From, Message-ID, MIME-Version, Reply-To, Return-Path, Subject, To

在 SES 中处理电子邮件附件

SES 中的电子邮件附件是您在使用 SES API v2 SendEmail 和SendBulkEmail操作时可以包含在电子邮件中的文件。此功能使您能够通过添加符合 SES 支持的 MIME 类型的文档（例如 PDFs Word 文

件、图像或其他文件类型）来丰富电子邮件内容。您还可以包含直接在电子邮件内容中呈现的内嵌图像，而无需收件人单独下载它们。每封电子邮件可以包含多个附件，邮件总大小上限为 40MB。

> **Note**
>
> `SendEmail`带有Raw内容类型、SMTP 接口的 SES API v2 和 SES API v1 继续通过原始电子邮件 M IME 消息构造来处理附件。

附件在 SES 中的工作原理

在发送带有附件的电子邮件时，在不同的阶段会发生两种不同的编码：

第 1 阶段 — 向 SES 发送数据：

- 当你想向 SES 发送附件时，需要将二进制数据（如 PDF 或图像）转换为可以安全传输的格式。
- 这就是 base64 编码的用武之地——这是必需的，因为你无法在 JSON 请求中发送原始二进制数据。
- 如果你使用的是 AWS SDK，它会自动处理这种编码。
- 如果您使用的是 AWS CLI，则需要在发送附件之前自己对其进行 base64 编码。

第 2 阶段 — SES 创建电子邮件：

- SES 收到您的数据后，需要创建一封包含附件的真实电子邮件。
- 这就是ContentTransferEncoding设置发挥作用的地方。
- SES 将使用您指定的任何编码 ContentTransferEncoding 方法自动格式化最后一封电子邮件中的附件。

可以这样想——它类似于通过邮件发送包裹。首先，您需要将包裹送到邮局（第 1 阶段——需要 Base64 编码），然后邮局会对其进行适当的包装以进行最终交付（第 2 阶段-ContentTransferEncoding）。

附件对象结构

当您通过 SES 发送带有附件的电子邮件时，该服务会自动处理复杂的 MIME 邮件结构。您只需要通过以下 SES API v2 `Attachment`对象结构提供附件内容和元数据：

- FileName（必填）-向收件人显示的文件名（必须包括文件扩展名）。如果未提供，SES 将ContentType从的扩展中派生FileName。
- ContentType（可选）— 符合 IANA 标准的媒体类型标识符。
- ContentDisposition（可选）-指定应如何呈现附件：ATTACHMENT（默认）或INLINE。
- ContentDescription（可选）-内容的简短描述。
- RawContent（必填）-附件的实际内容。
- ContentTransferEncoding（可选）-指定内容的编码类型：SEVEN_BIT（默认），BASE64或QUOTED_PRINTABLE。

所有附加内容都必须编码为 base64，例如：

- 纯文本内容：Text attachment sample content.
- Base64 编码：VGV4dCBhdHRhY2htZW50IHNhbXBsZSBjb250ZW50Lg==

以下示例说明了在使用 SES API v2 指定附件时如何使用附件对象结构，SendEmail以及如何使用 AWS CLI 引用包含附件对象元素的 JSON 文件进行SendBulkEmail操作。

Example — SendEmail 内容简单

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

request-send-email-simple.json

```
{
    "FromEmailAddress": "sender@example.com",
    "Destination": {
        "ToAddresses": [
            "recipient@example.com"
        ]
    },
    "Content": {
        "Simple": {
            "Subject": {
                "Data": "Email with attachment"
            },
            "Body": {
                "Text": {
```

```
                    "Data": "Please see attached document."
                },
                "Html": {
                    "Data": "Please see attached <b>document</b>."
                }
            },
            "Attachments": [
                {
                    "RawContent": "<base64-encoded-content>",
                    "ContentDisposition": "ATTACHMENT",
                    "FileName": "document.pdf",
                    "ContentDescription": "PDF Document Attachment",
                    "ContentTransferEncoding": "BASE64"
                }
            ]
        }
    }
}
```

Example — SendEmail 带有简单内容和内联附件

```
aws sesv2 send-email --cli-input-json file://request-send-email-simple-inline-
attachment.json
```

request-send-email-simple-inline-atchament.json

```
{
    "FromEmailAddress": "sender@example.com",
    "Destination": {
        "ToAddresses": [
            "recipient@example.com"
        ]
    },
    "Content": {
        "Simple": {
            "Subject": {
                "Data": "Email with attachment"
            },
            "Body": {
                "Html": {
                    "Data": "<html><body>Our logo:<br><img src=\"cid:logo123\" alt=
\"Company Logo\"></body></html>"
                }
```

```
            },
            "Attachments": [
                {
                    "RawContent": "<base64-encoded-content>",
                    "ContentDisposition": "INLINE",
                    "FileName": "logo.png",
                    "ContentId": "logo123"
                }
            ]
        }
    }
}
```

Example — SendEmail 包含模板内容

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

request-send-email-template.json

```
{
    "FromEmailAddress": "sender@example.com",
    "Destination": {
        "ToAddresses": [
            "recipient@example.com"
        ]
    },
    "Content": {
        "Template": {
            "TemplateName": "MyTemplate",
            "TemplateData": "{\"name\":\"John\"}",
            "Attachments": [
                {
                    "RawContent": "<base64-encoded-content>",
                    "ContentDisposition": "ATTACHMENT",
                    "FileName": "document.pdf",
                    "ContentDescription": "PDF Document Attachment",
                    "ContentTransferEncoding": "BASE64"
                }
            ]
        }
    }
}
```

Example — 附 SendBulkEmail 带附件内容

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

request-send-bulk-email.json

```
{
    "FromEmailAddress": "sender@example.com",
    "DefaultContent": {
        "Template": {
            "TemplateName": "MyTemplate",
            "TemplateData": "{}",
            "Attachments": [
                {
                    "RawContent": "<base64-encoded-content>",
                    "ContentDisposition": "ATTACHMENT",
                    "FileName": "document.pdf",
                    "ContentDescription": "PDF Document Attachment",
                    "ContentTransferEncoding": "BASE64"
                }
            ]
        }
    },
    "BulkEmailEntries": [
        {
            "Destination": {
                "ToAddresses": [
                    "recipient@example.com"
                ]
            },
            "ReplacementEmailContent": {
                "ReplacementTemplate": {
                    "ReplacementTemplateData": "{\"name\":\"John\"}"
                }
            }
        }
    ]
}
```

最佳实践

- 将邮件总大小（包括附件）控制在 40MB 以下。

- 尽可能让 SES 根据文件扩展名自动检测内容类型。
- 仅当内容类型不属于常见 MIME 类型时，才明确指定内容类型。
- 考虑使用内联图像来提高电子邮件呈现效果。
- SES 支持各种 MIME 类型的附件，但中不支持的附件类型列出的除外。

SES 不支持的附件类型

您可以使用多用途 Internet 邮件扩展（MIME）标准，通过 Amazon SES 发送带附件的电子邮件。Amazon SES 接受所有文件附件类型，带有以下列表中的文件扩展名的附件除外。

.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

有些 ISPs 还有其他限制（例如对存档附件的限制），因此我们建议您在发送正式电子邮件 ISPs 之前测试通过主要方式发送的电子邮件。

使用 Amazon SES 接收电子邮件

除了使用 Amazon SES 管理您的电子邮件发送外，您还可以配置 SES 代表您的一个或多个域接收电子邮件。作为电子邮件接收方，SES 会处理底层邮件接收操作，如与其他邮件服务器通信、扫描垃圾邮件和病毒、阻止来自不可信来源（Spamhaus 或 SES 的阻止列表上的地址）的邮件，以及为所在域内的收件人接收邮件等。

您收到的电子邮件的处理范围由您指定的自定义说明决定。这些说明有以下两种形式：

- 接收规则（基于收件人的控制）可提供对传入电子邮件的最佳控制精细度。收据规则可以进行高级处理，例如将收到的邮件传送到 Amazon S3 存储桶、将其发布到 Amazon SNS 主题、发送到亚马逊 WorkMail，或者在消息发送到特定电子邮件地址时自动发送退回消息等等。
- IP 地址筛选条件（基于 IP 的控制）提供了广泛的控制级别，且易于设置。这些筛选条件允许您显式阻止或允许来自特定 IP 地址或 IP 地址范围的所有邮件。

要开始学习使用接收规则或 IP 地址筛选器进行电子邮件接收、设置和实施，请先通读 电子邮件接收概念和使用案例，了解其工作原理和不同使用方式的概览。接下来，设置电子邮件接收 将指导您完成电子邮件接收设置的先决条件。然后，电子邮件接收控制台演练 将指导您完成用于配置接收规则和 IP 地址筛选条件的向导。

> **Note**
>
> 只有当您的账户位于 SES 支持电子邮件接收 AWS 区域 的地方时，才能使用电子邮件接收功能。中的电子邮件接收端点表 AWS 一般参考 列出了 SES 支持电子邮件接收的所有终端节点。 AWS 区域

本节中的主题：

- Amazon SES 电子邮件接收概念和使用案例
- 设置 Amazon SES 电子邮件接收
- Amazon SES 电子邮件接收控制台演练
- 查看 Amazon SES 电子邮件接收的指标

Amazon SES 电子邮件接收概念和使用案例

当使用 Amazon SES 作为电子邮件接收方时，您需要告诉该服务如何处理您的邮件。主要方法是通过接收规则，使用基于收件人的控制来指定基于收件人执行的一组操作，从而为您提供对电子邮件接收的精细化控制。另一种方法是 IP 地址筛选条件，能够提供广泛的基于 IP 的控制，以根据来源 IP 地址或地址范围阻止或允许邮件。

本节介绍了这两种方法，以及 Amazon SES 如何处理收到的电子邮件的概览，并通过使用案例来帮助您在设置规则和筛选条件时考虑如何接收、筛选和处理电子邮件。

本节中的主题：

使用接收规则进行基于收件人的控制

控制传入邮件的主要方法是指定如何通过您的任何已验证身份（包括域、子域或电子邮件地址）的有序操作列表来处理邮件。请注意，电子邮件地址必须属于您的已验证域身份之一。这些操作在规则集中您创建的接收规则中进行定义和排序。

另一种选项是，您还可以添加收件人条件，以指定仅在传入邮件的收件人地址与条件中指定的收件人身份匹配时才执行操作。例如，如果您拥有 example.com，您可以指定 user@example.com 收到的邮件应退回，并且 example.com 及其子域的所有其他邮件将送达。

否则，如果您不添加任何收件人条件，则操作将应用于所有内容，包括所有电子邮件地址、域和属于已验证域的子域。以下操作可应用于您的接收规则：

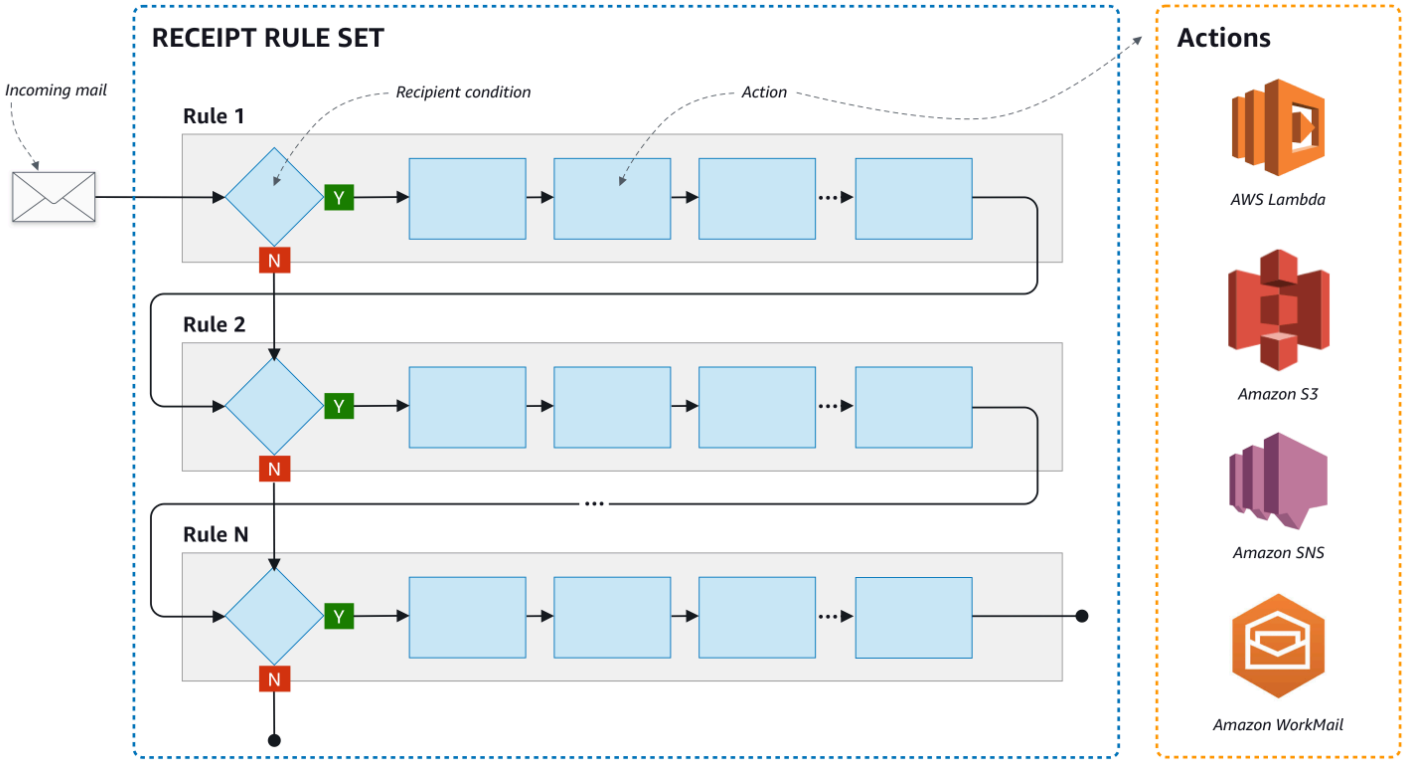
- 添加标头操作 – 向收到的电子邮件添加一个标头。此操作通常仅与其他操作结合使用。
- Return bounce response action（返回退回邮件响应操作）– 通过将退回邮件响应返回给发件人来阻止电子邮件，并通过 Amazon SNS 通知您（可选）。
- 调用 AWS Lambda 函数操作- 通过 Lambda 函数调用您的代码，也可以选择通过 Amazon SNS 通知您。
- 传送到 S3 存储桶操作 – 此操作会将邮件传送到 Amazon S3 存储桶，并通过 Amazon SNS 通知您（可选）。

- 发布到 Amazon SNS 主题操作—将完整的电子邮件发布到 Amazon SNS 主题。

> **Note**
>
> SNS 操作包含 Amazon SNS 通知中电子邮件内容的完整副本。此处提到的其他 Amazon SNS 通知选项只通知您电子邮件的送达情况，它们包含与电子邮件相关的信息，但不包含电子邮件内容本身。

- 停止规则集操作 – 终止对接收规则集的评估，并会通过 Amazon SNS 通知您（可选）。
- 与亚马逊整合 WorkMail 操作- 使用亚马逊处理邮件 WorkMail。您通常不会直接使用此操作，因为设置由Amazon WorkMail 负责。

接收规则将聚集到规则集。如果没有现有的规则集，则必须先创建规则集，然后才能开启创建接收规则。您可以为 AWS 账户定义多个规则集，但任何时候都只有一个规则集处于活动状态。下图显示了接收规则、规则集和操作之间的关系。



使用 IP 地址筛选条件进行基于 IP 的控制

您可以通过设置 IP 地址筛选器控制邮件流。IP 地址筛选器是可选的，供您指定是接受还是阻止来自某个 IP 地址或 IP 地址范围的邮件。IP 地址筛选器可以包括阻止列表（包含您想要阻止传入邮件的 IP 地址）和允许列表（包含您想要始终接收邮件的 IP 地址）。

IP 地址筛选器对阻止垃圾邮件非常有效。Amazon SES 会维护自己的阻止列表，其中包括 Spamhaus 中列出的已知发送垃圾邮件的 IP 地址。但是，您可选择接收来自这些 IP 地址的邮件，方法是将这些地址添加到您的允许列表。由于没有用于显示哪些 IP 地址受到阻止的日志，因此受到阻止的发件人需要通知您。这也是帮助发件人确定其 IP 地址是否在阻止列表（例如 Spamhaus）中的好机会，并建议他们要求从列表中撤除。这样做对您和发件人都有好处，因为您不必为发件人维护 IP 地址筛选器，而且发件人将提高其电子邮件可送达性。

> **Note**
>
> - 无论您的 IP 地址筛选器配置如何，Amazon 都 EC2 将屏蔽端口 25（邮件发送）上的出站流量，除非列入许可名单。有关更多信息，请参阅这篇 AWS re: Post 文章。
> - 如果您只希望接收来自有限已知 IP 地址列表的邮件，则请设置包含 `0.0.0.0/0` 的阻止列表，并设置包含所信任 IP 地址的允许列表。原定设置情况下，此配置会阻止全部 IP 地址，并仅允许来自您显式指定的 IP 地址的邮件。

电子邮件接收过程

当 Amazon SES 为您的域接收电子邮件时，将发生以下事件：

1. Amazon SES 首先查看发件人的 IP 地址。在以下情况下，Amazon SES 不允许邮件通过此阶段：
   - IP 地址位于您的阻止列表中。
   - IP 地址在 Amazon SES 阻止列表中但不在您的允许列表中。
2. Amazon SES 检查活动规则集，以确定是否有任何接收规则包含收件人条件：
   - 如果存在收件人条件，且该条件与任何传入电子邮件的收件人匹配，则 Amazon SES 将接受该电子邮件。否则，如果没有任何匹配项，Amazon SES 将阻止电子邮件。
   - 如果接收规则不包含收件人条件，则 Amazon SES 将接受邮件，该规则的所有操作都将应用于您拥有的所有已验证身份。
3. Amazon SES 对电子邮件进行身份验证，并扫描其内容中是否存在垃圾邮件和恶意软件：

- 根据在 SMTL 事务期间使用的 MAIL FROM 的域下指定的 SPF 策略检查将电子邮件传输到 Amazon SES 的远程主机的 IP 地址。
- 检查电子邮件标题部分中存在的 DKIM 签名。
- 如果启用了内容扫描，则会扫描电子邮件内容中是否存在垃圾邮件和恶意软件。
- 电子邮件身份验证和内容扫描结果将在接收规则评估期间提供给您。

请参阅电子邮件身份验证和恶意软件检测了解更多信息。

4. 对于 Amazon SES 接受的电子邮件，激活的规则集中的所有接收规则都将按照您定义的顺序应用；在每个接收规则中，操作将按照您定义的顺序执行。

Amazon SES 电子邮件接收的使用案例和限制

本节介绍了 Amazon SES 电子邮件接收中的一些一般注意事项和使用案例。常见问题和事实将以问答形式呈现，以帮助确定使用 Amazon SES 代表您拥有的一个或多个已验证域来接收和管理电子邮件是否有益。

区域可用性

在您的区域中，Amazon SES 是否支持接收电子邮件？

Amazon SES 仅支持在某些 AWS 地区接收电子邮件。有关支持电子邮件接收的区域的完整列表，请参阅《 AWS 一般参考》中的 Amazon Simple Email Service 端点和限额。

基于 POP 或 IMAP 的电子邮件客户端

是否可以使用 Microsoft Outlook 接收传入的电子邮件？

Amazon SES 不包括用于接收传入电子邮件的 POP 或 IMAP 服务器。这意味着您不能使用 Microsoft Outlook 等电子邮件客户端来接收传入的电子邮件。如果您需要一种能够使用电子邮件客户端发送和接收电子邮件的解决方案，请考虑使用 Amazon WorkMail。

使用其他 AWS 服务

您是否已设置适当的权限？

如果需要将邮件传送到 S3 存储桶、发布到不属于您的 Amazon SNS 主题、触发 Lambda 函数或使用客户托管式密钥，您需要向 Amazon SES 授予访问这些资源的权限。要授予 Amazon SES 访问权限，您需要 APIs 针对来自控制台的资源或这些 AWS 服务创建策略。有关 授予权限。

电子邮件内容

您希望 Amazon SES 如何传递电子邮件内容？

Amazon SES 可以通过两种方式为您提供电子邮件内容：它可以将电子邮件存储在您指定的 S3 存储桶中，也可以向您发送 Amazon SNS 通知，其中包含电子邮件的副本。Amazon SES 以多用途 Internet 邮件扩展（MIME）格式为您传送原始、未经修改的电子邮件。有关 MIME 格式的更多信息，请参阅 RFC 2045。

您将接收多大的电子邮件？

如果您选择将电子邮件存储在 S3 桶中，最大电子邮件大小（包括标头）为 40 MB。如果您通过 Amazon SNS 通知接收电子邮件，则最大电子邮件大小（包括标头）为 150KB。

您希望如何触发邮件处理？

在邮件送达后，您需要使用自定义的代码来处理它。例如，您的应用程序可能会将 Base-64 编码的电子邮件转换为可显示的格式，并通过电子邮件客户端提供给最终用户。您可以通过几种方法开始此过程：

- 如果您的电子邮件传送到 Amazon S3，您的应用程序可以侦听 S3 操作生成的 Amazon SNS 通知，从通知中提取电子邮件的消息 ID，然后使用该消息 ID 从 Amazon S3 中检索电子邮件。

  或者，您也可以编写 Lambda 函数，将电子邮件的处理方式纳入接收规则中。在这种情况下，接收规则首先将电子邮件写入 Amazon S3，再触发 Lambda 函数。Lambda 操作可以在接收规则内同步或异步执行，具体取决于 Lambda 函数是否需要返回可能影响其他操作执行的结果。我们建议您使用异步执行，除非您的使用案例中绝对需要同步执行。有关的更多信息 AWS Lambda，请参阅《AWS Lambda 开发人员指南》。
- 如果您的电子邮件使用 SNS 操作通过 Amazon SNS 通知传送，您的应用程序可以侦听 Amazon SNS 通知，然后从通知中提取电子邮件。

您是否希望加密电子邮件？

Amazon SES 与 AWS Key Management Service (AWS KMS) 集成，可以选择加密它写入您的 S3 存储桶的邮件。Amazon SES 在将您的邮件写入到 Amazon S3 之前，使用客户端加密对邮件进行加密。这意味着，在从 Amazon S3 取回邮件后，您必须负责对内容进行解密。适用于 Java 的 AWS SDK 和适用于 Ruby 的 AWS SDK 提供能够为您处理解密的客户端。Amazon SES 仅在您选择将电子邮件传送到 S3 存储桶时才加密电子邮件。

不需要的邮件

您希望在电子邮件接收过程中的哪个阶段阻止不需要的邮件？

当发件人尝试向收件人发送电子邮件时，发件人的电子邮件服务器将与收件人的服务器交换一系列命令。此序列称为 SMTP 会话。

您可以在电子邮件接收过程中的两个点阻止传入的电子邮件：SMTP 会话期间和 SMTP 会话之后。使用 IP 地址筛选器 可在 SMTP 会话期间阻止邮件，使用接收规则 可在 SMTP 会话之后阻止电子邮件。

您可以使用 IP 地址筛选器阻止源自特定 IP 地址的电子邮件。使用 IP 地址筛选器阻止不需要的电子邮件的好处是，对于在 SMTP 会话期间被阻止的邮件，我们不会向您收取费用。使用 IP 地址筛选器的缺点是，它们会阻止来自您指定的 IP 地址的电子邮件，而不对邮件的实际内容进行任何分析。有关 IP 地址筛选器的更多信息，请参阅创建 IP 地址筛选条件控制台演练。

您可以使用接收规则根据邮件发送到的地址（或者域或子域）向电子邮件的发件人发送退回邮件通知。使用接收规则的好处是，您可以先对传入邮件执行额外分析，然后再向发件人发送退回邮件通知。例如，只有当邮件未通过 DKIM 身份验证或被识别为垃圾邮件时，您才可以使用 AWS Lambda 发送退回通知。使用接收规则的缺点是，由于在 SMTP 会话后处理接收规则，因此，我们会就您收到的每封邮件向您收取费用。如果您使用 Lambda 分析传入邮件的内容，可能还需支付费用。有关接收规则的更多信息，请参阅创建接收规则控制台演练。有关使用 Lambda 分析传入电子邮件的更多信息，请参阅Lambda 函数示例。

邮件流

您打算如何划分邮件流？

您的域很可能收到不同类别的邮件。例如，域中的某些邮件 (例如，发送给 user@example.com 的电子邮件) 可能面向个人收件箱。其他邮件 (例如，发送给 unsubscribe@example.com 的电子邮件) 可能最好定向到自动运行的系统。您可以使用接收规则来划分传入邮件，使其得到不同处理。有关如何设置接收规则的信息，请参阅创建接收规则。

电子邮件接收身份验证和恶意软件扫描

Amazon SES 对收到的每封电子邮件进行身份验证，并扫描其内容中是否存在垃圾邮件和恶意软件：SES 不会根据电子邮件身份验证或内容扫描结果对收到的电子邮件采取任何操作；但是，这些操作的结果将作为属性提供给您，以便您可以在 SES 接收规则操作（例如 Amazon SNS 通知中使用它们，或作为投递到 Amazon S3 的邮件标题。

电子邮件身份验证

Amazon SES 使用 SPF、DKIM 和 DMARC 对收到的每封电子邮件进行身份验证。作为活动接收规则集中的规则评估的一部分，每种身份验证机制的结果都会在 SES 发送的 Amazon SNS 通知中提供。此外，如果您选择在 Amazon S3 中接收电子邮件副本，则电子邮件身份验证的结果将被捕获到 SES 添加到电子邮件标题部分的 `Authentication-Results` 标题中：

```
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` 标题如 RFC 8601 所述

电子邮件内容中的垃圾邮件和恶意软件扫描

Amazon SES 会根据与电子邮件匹配的接收规则的 ScanEnabled(API) 或垃圾邮件和病毒扫描 (控制台) 属性的值来扫描收到的电子邮件内容中是否存在恶意软件。默认情况下，SES 会扫描收到的电子邮件内容中是否存在恶意软件。要禁用对已收到的符合特定接收规则的电子邮件的内容扫描，如果使用 API，则需要将接收规则的ScanEnabled标志设置为 false，如果使用控制台，则需要清除 “垃圾邮件和病毒扫描” 复选框。如果已启用与电子邮件匹配的接收规则扫描，则内容扫描结果会在 SES 发送的 Amazon SNS 通知中提供，作为活动接收规则集中的规则评估的一部分。此外，如果您选择在 Amazon S3 中接收电子邮件副本，则内容扫描结果将被捕获到 SES 添加到电子邮件标题部分的 `X-SES-Spam-Verdict` 和 `X-SES-Virus-Verdict` 标题中。

```
X-SES-Spam-Verdict: PASS
X-SES-Virus-Verdict: FAIL
```

上述标题的可能值如下所列：

- 垃圾电子邮件
- 病毒

现在，您已经了解了电子邮件接收的概念、工作原理及使用案例，接下来您可以开启 设置电子邮件接收。

设置 Amazon SES 电子邮件接收

本节介绍了开始配置 Amazon SES 以接收邮件之前所需的先决条件。请务必先阅读 电子邮件接收概念和使用案例，以了解 Amazon SES 工作原理的概念，并考虑希望如何接收、筛选和处理电子邮件。

在创建规则集、接收规则和 IP 地址筛选条件配置电子邮件接收之前，必须先完成以下设置先决条件：

- 发布 DNS 记录证明您拥有域，使用 Amazon SES 验证您的域。
- 允许 Amazon SES 发布 MX 记录来接收您域的电子邮件。
- 授予 Amazon SES 访问其他 AWS 资源的权限，以便执行接收规则操作。

创建和验证域身份时，您需要将记录发布到 DNS 设置，以完成验证过程，但仅凭这一点还不足以使用电子邮件接收。针对电子邮件接收，还需要发布 MX 记录以指定自定义邮件发件人域。此记录用于您域的 DNS 设置，以允许 SES 接收您域的电子邮件。授予权限是必需的，因为除非 Amazon SES 有权使用这些操作所需的相应 AWS 服务，否则您在收据规则中选择的操作将不起作用。

以下主题介绍了使用电子邮件接收所需的三个先决条件：

- 为 Amazon SES 接收电子邮件验证您的域
- 为 Amazon SES 电子邮件接收发布 MX 记录
- 授予 Amazon SES 电子邮件接收的权限

为 Amazon SES 接收电子邮件验证您的域

与任何您要使用 Amazon SES 发送或接收电子邮件的域相同，您必须首先证明您拥有该域。验证过程包括使用 SES 启动域验证，然后以别名记录或 TXT 格式将 DNS 记录发布到您的 DNS 提供商，具体取决于您使用何种验证方式。

通过控制台，您可以使用 Easy DKIM 或 自带 DKIM (BYODKIM) 验证您的域，并轻松复制其 DNS 记录以发布到您的 DNS 提供商 - 见 创建域身份 中的操作方法说明。或者，您可以使用 SES `VerifyDomainDkim`或`VerifyDomainIdentity` APIs。

您可以在 SES 控制台的 “已验证身份” 表中查看其状态，或者使用 SES 或，轻松确认您的域名或电子邮件地址已通过验证`GetEmailIdentity` APIs。`GetIdentityVerificationAttributes`

为 Amazon SES 电子邮件接收发布 MX 记录

邮件交换器 记录（MX 记录）是一种配置，指定哪些邮件服务器可以接受发送到您的域的电子邮件。

要让 Amazon SES 管理您的传入电子邮件，您需要将 MX 记录添加到您的域的 DNS 配置。您创建的 MX 记录是指接收您使用 Amazon SES 的 AWS 地区的电子邮件的终端节点。例如，美国西部（俄勒冈）区域的终端节点为 inbound-smtp.us-west-2.amazonaws.com。有关完整的终端节点列表，请参阅SES 区域和端点。

> **Note**
>
> 在 Amazon SES 中接收电子邮件的终端节点不是 IMAP 或 POP3 电子邮件服务器。您不能 URLs 将它们用作电子邮件客户端中的传入邮件服务器。如果您需要一种能够使用电子邮件客户端发送和接收电子邮件的解决方案，请考虑使用 Amazon WorkMail。

以下过程包括 MX 记录的常规创建步骤。创建 MX 记录的具体步骤取决于您的 DNS 或托管服务提供商。有关将 MX 记录添加到域的 DNS 配置的信息，请参阅提供商的文档。

> **Note**

将 MX 记录添加到域的 DNS 配置

1. （先决条件）要完成这些过程，您需要修改域名的 DNS 记录。如果您无法访问 DNS 记录，或者不愿意访问 DNS 记录，请联系您的系统管理员寻求帮助。
2. 登录 DNS 提供商的管理控制台。
3. 创建新的 MX 记录。
4. 对于 MX 记录的 Name (名称)，输入您的域。例如，如果您希望 Amazon SES 管理发送到域 example.com 的电子邮件，请输入以下内容：

   ```
   example.com.
   ```

   > **Note**
   >
   > 视您的 DNS 提供商而定：1) 可能不需要.在域名后缀末尾添加尾随字符。2) “名称” 字段可以称为 “主机”、“域” 或 “邮件域”。

5. 对于 Type（类型），选择 MX。

> **Note**
>
> 一些 DNS 提供商将 Type（类型）字段称为 Record Type（记录类型）或类似名称。

6. 对于 Value（值），输入以下内容：

```
10 inbound-smtp.region.amazonaws.com
```

在前面的示例中，*region*替换为您在 Amazon SES 中使用的 AWS 区域接收电子邮件的终端节点的地址。例如，如果您使用的是美国东部（弗吉尼亚北部）区域，请*region*替换为us-east-1。有关电子邮件接收终端节点的完整列表，请参阅 SES 区域和端点。

> **Note**
>
> 对于记录 Value（值）和记录 Priority（优先级），一些 DNS 提供商的管理控制台包含单独的字段。如果您的 DNS 提供商是这种情况，请输入 10 作为 Priority（优先级）值，并为 Value（值）输入传入邮件终端节点 URL。

> **Important**
>
> MX 记录的具体创建过程取决于您的 DNS 或托管提供商。有关向域名的 DNS 配置中添加 MX 记录的信息，请参阅您的提供商的文档或与他们联系。

有关为不同提供商创建 MX 记录的说明

为您的域创建 MX 记录的过程取决于您使用的 DNS 提供商。本节包含指向几个常用 DNS 提供商的文档的链接。此列表不是完整提供商列表。如果下面未列出您的提供商，您仍可以通过 Amazon SES 来使用此提供商。此列表中包含的不是对任何公司的产品或服务的认可或推荐。

DNS/托管提供商名称	文档链接
Amazon Route 53	使用 Amazon Route 53 控制台创建记录
GoDaddy	添加 MX 记录（外部链接）

DNS/托管提供商名称	文档链接
DreamHost	如何更改我的 MX 记录？（外部链接）
Cloudflare	设置电子邮件记录（外部链接）
HostGator	更改 MX 记录 - Windows（外部链接）
Namecheap	如何设置邮件服务所需的 MX 记录？（外部链接）
Names.co.uk	更改您的域的 DNS 设置（外部链接）
Wix	在您的 Wix 账户中添加或更新 MX 记录（外部链接）

授予 Amazon SES 电子邮件接收的权限

在 SES 中接收电子邮件时可以执行的某些任务，例如向亚马逊简单存储服务 (Amazon S3) Service 存储桶发送电子邮件或调用 AWS Lambda 函数，需要特殊权限。本节包含几个常用案例的示例策略。

本节中的主题：

- 设置 IAM 角色权限以执行“交付到 S3 存储桶”操作
- 授予 SES 写入 S3 存储桶的权限
- 授予 SES 使用您的 AWS KMS 密钥的权限
- 授予 SES 调用 AWS Lambda 函数的权限
- 授予 SES 发布属于不同账户的 Amazon SNS 主题的权限 AWS

设置 IAM 角色权限以执行“交付到 S3 存储桶”操作

以下几点适用于此 IAM 角色：

- 它只能用于交付到 S3 存储桶操作。
- 如果要向 SES the section called “电子邮件接收”不可用的区域中的 S3 存储桶写入数据，则必须使用该角色。

如果要写入到 S3 存储桶，您可以向 IAM 角色提供访问相关资源的权限，以便交付到 S3 存储桶操作。您还需要授予 SES 权限才能代入该角色，以便通过 IAM 信任策略执行操作，如下一节所述。

必须将此权限策略粘贴到 IAM 角色的内联策略编辑器中，请参阅交付到 S3 存储桶操作，并按照 IAM 角色项目中给出的步骤进行操作。（以下示例还包括可选权限，以防您想在 S3 操作中使用 SNS 主题通知或客户托管密钥。）

```
{
    "Version": "2012-10-17",
    "Statement": [
        // Required: allows SES to write in the bucket
        {
            "Sid": "S3Access",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-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"
        }
    ]
}
```

对前面的策略示例进行以下更改：

- *amzn-s3-demo-bucket*替换为您要写入的 S3 存储桶的名称。
- *region*替换为您创建接收规则 AWS 区域 的位置。
- 将 *111122223333* 替换为您的 AWS 账户 ID。
- *my-topic*替换为要向其发布通知的 SNS 主题的名称。
- *key-id*替换为您的 KMS 密钥的 ID。

适用于 S3 操作 IAM 角色的信任策略

应将以下信任策略添加到 IAM 角色的信任关系中，以允许 SES 代入该角色。

> **Note**
>
> 仅当您未使用交付到 S3 存储桶操作工作流程的 IAM 角色项目中给出的步骤，从 SES 控制台创建 IAM 角色时，才需要手动添加此信任策略。当从控制台创建 IAM 角色时，系统会自动生成此信任策略并将其应用于该角色，因此您无需执行此步骤。

JSON

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

对前面的策略示例进行以下更改：

- *region*替换为您创建接收规则 AWS 区域 的位置。
- 将 *111122223333* 替换为您的 AWS 账户 ID。
- *rule_set_name*替换为包含接收规则（包含传送到 Amazon S3 存储桶操作）的规则集的名称。

- *receipt_rule_name*替换为包含传送到 Amazon S3 存储桶操作的接收规则名称。

授予 SES 写入 S3 存储桶的权限

当您将以下策略应用于 S3 存储桶时，只要该存储桶位于支持 SES 电子邮件接收的区域中，它就会授予 SES 写入该存储桶的权限；如果您想写入电子邮件接收区域之外的存储桶，请参阅设置 IAM 角色权限以执行“交付到 S3 存储桶”操作。有关创建接收规则，向 Amazon S3 传输传入电子邮件的更多信息，请参阅交付到 S3 存储桶操作。

有关将策略附加到 S3 的存储桶的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用存储桶策略和用户策略。

JSON

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

对前面的策略示例进行以下更改：

- *amzn-s3-demo-bucket*替换为您要写入的 S3 存储桶的名称。

- *region*替换为您创建接收规则的 AWS 区域。
- 将 *111122223333* 替换为您的 AWS 账户 ID。
- *rule_set_name*替换为包含接收规则（包含传送到 Amazon S3 存储桶操作）的规则集的名称。
- *receipt_rule_name*替换为包含传送到 Amazon S3 存储桶操作的接收规则名称。

授予 SES 使用您的 AWS KMS 密钥的权限

为了让 SES 能够加密您的电子邮件，它必须具有使用 AWS KMS 密钥的权限，该密钥在设置接收规则时由您指定。您可以使用账户中的默认 KMS 密钥（aws/ses），也可以使用您创建的客户托管式密钥。如果您使用默认 KMS 密钥，不需要执行任何其他步骤来为 SES 提供使用权限。若要使用客户托管式密钥，您需要向 SES 提供该密钥的使用权限，方法是在密钥策略中添加一条语句。

使用以下策略语句作为密钥策略，以允许 SES 在您的域上接收电子邮件时使用您的客户托管式密钥。

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

对前面的策略示例进行以下更改：

- *region*替换为您创建接收规则的 AWS 区域。
- 将 *111122223333* 替换为您的 AWS 账户 ID。
- 替换为包含您*rule_set_name*与电子邮件接收关联的接收规则的规则集的名称。
- 替换为您*receipt_rule_name*与电子邮件接收关联的接收规则的名称。

如果您使用 AWS KMS 向启用服务器端加密的 S3 存储桶发送加密消息，则需要添加策略操作。"kms:Decrypt"使用前面的示例，将此操作添加到策略中，如下所示：

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

有关为 AWS KMS 密钥附加策略的更多信息，请参阅《AWS Key Management Service 开发人员指南》AWS KMS中的使用密钥策略。

授予 SES 调用 AWS Lambda 函数的权限

要允许 SES 调用 AWS Lambda 函数，您可以在 SES 控制台中创建接收规则时选择该函数。当您执行此操作时，SES 会自动为此函数添加必要的权限。

此外，也可以使用 AWS Lambda API 中的 AddPermission 操作将策略附加到函数。以下 AddPermission API 调用可授予 SES 调用 Lambda 函数的权限。有关向 Lambda 函数附加策略的更多信息，请参阅《AWS Lambda 开发人员指南》中的 AWS Lambda 权限。

```
{
  "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"
```

```
}
```

对前面的策略示例进行以下更改：

- *region*替换为您创建接收规则的 AWS 区域。
- 将 *111122223333* 替换为您的 AWS 账户 ID。
- *rule_set_name*替换为包含您在其中创建 Lambda 函数的接收规则的规则集的名称。
- *receipt_rule_name*替换为包含您的 Lambda 函数的接收规则的名称。

授予 SES 发布属于不同账户的 Amazon SNS 主题的权限 AWS

要在单独的 AWS 账户中发布针对某个主题的通知，您必须在 Amazon SNS 主题中附加政策。SNS 主题必须与域和接收规则集位于同一区域中。

以下政策授予 SES 使用单独 AWS 账户向 Amazon SNS 主题发布内容的权限。

JSON

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

对前面的策略示例进行以下更改：

- *topic_region*替换为 AWS 区域 创建 Amazon SNS 主题时使用的。
- *sns_topic_account_id*替换为拥有 Amazon SNS 主题的 AWS 账户的 ID。
- *topic_name*替换为您要向其发布通知的 Amazon SNS 主题的名称。
- *aws_account_id*替换为配置为接收电子邮件的 AWS 账户的 ID。
- *receipt_region*替换为您创建接收规则 AWS 区域 的位置。
- *rule_set_name*替换为包含您在其中创建 “发布到 Amazon SNS” 主题操作的接收规则的规则集的名称。
- *receipt_rule_name*替换为包含 “发布到 Amazon SNS” 主题操作的接收规则名称。

如果您的 Amazon SNS 主题 AWS KMS 用于服务器端加密，则必须向密钥策略添加权限。 AWS KMS 您可以通过将以下策略附加到 AWS KMS 密钥策略来添加权限：

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSESToUseKMSKey",
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon SES 电子邮件接收控制台演练

本节介绍了电子邮件接收控制台向导，可用于配置接收规则和 IP 地址筛选条件以管理电子邮件接收。在使用控制台向导之前，请务必先阅读 电子邮件接收概念和使用案例（以了解电子邮件接收工作原理）和 设置电子邮件接收（以确保已完成设置先决条件）。

用于配置接收规则和 IP 地址筛选条件的控制台向导解释如下：

- 创建接收规则控制台演练
- 创建 IP 地址筛选条件控制台演练

创建接收规则控制台演练

本节将指导您使用 Amazon SES 控制台创建和定义接收规则。理解接收规则工作原理的关键点在于：

- 规则集包含一组有序的接收规则；接收规则包含一组有序的操作。
- 接收规则告诉 Amazon SES 如何通过执行指定的有序操作列表来处理传入邮件。
- 可以选择根据首先匹配收件人条件来设置此有序的操作列表；如果未指定，则这些操作将应用于属于您的已验证域的所有身份。
- 接收规则在名为规则集的容器中创建和定义，虽然您可以创建多个规则集，但一次只能有一个规则集处于激活状态。
- 激活的规则集中的接收规则将按指定的顺序执行。
- 在创建接收规则之前，必须首先创建包含这些规则的规则集。

或者，您可以使用 `CreateReceiptRuleSet` API 创建一个空接收规则集，如 Amazon Simple Email Service API 参考中所述。随后，您可以使用 Amazon SES 控制台或 `CreateReceiptRule` API 来向其添加接收规则。

在继续演练之前，请确保您已满足使用基于收件人的电子邮件接收所需的所有必要先决条件。以及

先决条件

在继续使用接收规则设置基于收件人的电子邮件控制之前，必须满足以下先决条件：

1. 确保您的终端节点位于 Amazon SES 支持电子邮件接收 AWS 区域 的地方。中的电子邮件接收端点表 AWS 一般参考 列出了 SES 支持电子邮件接收的所有终端节点 AWS 区域 的电子邮件接收端点。

2. 首先需要在 Amazon SES 中创建并验证域身份。
3. 接下来，您需要通过将 MX 记录发布域的 DNS 设置来指定哪些邮件服务器可以接收域的邮件。（MX 记录应指接收您使用 Amazon SES 所在 AWS 地区的邮件的 Amazon SES 终端节点。）
4. 最后，您需要授予 Amazon SES 访问其他 AWS 资源的权限才能执行接收规则操作。

创建规则集和接收规则

本演练首先会创建包含规则的规则集，然后进入创建规则向导，以创建、定义和排序接收规则。该向导包含四个屏幕，分别用于定义规则设置、添加收件人条件、添加操作以及查看所有设置。

使用控制台创建规则集和接收规则

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下面，选择电子邮件接收。

> **Note**
>
> 如果您的账户位于 SES 不支持电子邮件接收的 AWS 区域 ，则在 SES 控制台的左侧导航窗格中将看不到电子邮件接收。请参阅the section called “先决条件”中列出的第一项。

3. 在 Email receiving（电子邮件接收）窗格的 Receipt rule sets（接收规则集）选项卡下，选择 Create rule set（创建规则集）。
4. 为规则集输入唯一的名称，然后选择创建规则集。
5. 选择创建规则，此时将打开创建规则向导。
6. 在定义规则设置页面的接收规则详细信息下，输入规则名称。
7. 对于状态，只有当您不希望 Amazon SES 在创建后运行此规则时，清除启用复选框；否则，请保持此选项处于选中状态。
8. （可选）在安全和保护选项下，对于传输层安全性 (TLS)，如果需要 Amazon SES 拒绝未通过安全连接发送的传入邮件，请选择必需。
9. （可选）对于垃圾邮件和病毒扫描，如果需要 Amazon SES 扫描传入邮件中是否有垃圾邮件和病毒，请选择启用。
10. 要继续执行下一个步骤，请选择下一步。
11. （可选）在添加收件人条件页面上，使用以下过程指定一个或多个收件人条件。每个接收规则最多可包含 100 个收件人条件。

a. 在收件人条件下，选择添加新收件人条件，可指定要应用接收规则的接收电子邮件地址或域。下表使用地址 user@example.com 来显示如何指定收件人条件。

如果要...	指定以下收件人...	备注
匹配特定的电子邮件地址。	user@example.com	包含标签的地址变体也匹配 (如 user+123@example.com 和 user+xyz@example.com)。但是，如果您指定包含标签的地址，则只会匹配该特定地址。
匹配一个域中的全部地址，但不匹配子域中的地址。	example.com	
匹配特定子域中的全部地址，但不匹配父域中的地址。	subdomain.example.com	
匹配全部子域中的全部地址，但不匹配父域中的地址。	.example.com	注意域名之前的句点 (.)。
匹配某个域中的所有地址，及其所有子域中的所有地址。	example.com .example.com	创建两个单独的收件人：一个是域名，另一个是句点后接域名。
匹配所有已验证域中的所有收件人	[无]	将收件人字段留空。

Important

如果多个 Amazon SES 账户均接收某一通用域的电子邮件（例如，同一公司的多个团队各自拥有单独的 Amazon SES 账户），Amazon SES 将同时为各个账户处理所有

匹配的接收规则。此行为可能导致一个账户生成退回邮件，而另一个账户接受该电子邮件的情况。
我们建议您与组织中其他使用 Amazon SES 的团队相互协调，以确保每个账户使用唯一的接收规则，并且这些规则不重叠。在这些情况下，最好将接收规则配置为仅使用您的群组或团队独有的电子邮件地址或子域。

b. 对要添加的每个收件人条件重复此步骤。添加完收件人条件后，选择下一步。

12. 在添加操作页面上，使用以下过程将一个或多个操作添加到接收规则。

a. 打开添加新操作菜单，然后选择以下操作类型之一：

- 添加标头 - 此操作向收到的电子邮件添加一个自定义标头。
- 返回退回邮件响应 - 此操作通过向发件人返回退回邮件响应来拒绝收到的电子邮件。
- 调用 Lambda 函数-此操作通过 AWS Lambda 函数调用您的代码。
- 交付到 S3 存储桶 - 此操作将收到的电子邮件存储在 Amazon Simple Storage Service (S3) 存储桶中。
- 发布到 Amazon SNS 主题 - 此操作将完整的电子邮件发布到 Amazon Simple Notification Service (SNS) 主题。
- 停止规则集 - 此操作终止对接收规则集的评估。
- 与 Amazon 集成 WorkMail-此操作与 Amazon 集成 WorkMail。

有关每个操作的更多信息，请参阅 操作选项。

b. 对要定义的每个操作重复此步骤。如果定义了多个操作，可以使用操作容器中的向上/向下箭头对其重新排序。选择 Next (下一步) 可继续到 Review (审核) 页面。

13. 在审核页面上，审核规则的设置和操作。如果需要进行更改，请选择编辑选项，或通过页面左侧的导航部分直接转到包含要编辑的内容的步骤。您可以选择使用重新排序列中的向上/向下箭头更改审核页面的操作表中列出的操作顺序。

14. 如果您已准备好继续，请选择创建规则。

15. 在规则集的确认页面上，如果要立即实施规则集，请选择 Set as active（设置为活动状态）。

创建后的规则修改

创建规则集后，可以编辑规则集及其包含的接收规则。不仅可以编辑，还可以选择复制规则集或其规则，以便快速创建新规则集或规则。下表显示了规则集和接收规则的可用修改：

- 规则集将列出其名称、状态和创建日期。规则集的修改选项包括：
  - 设置为激活/非激活切换按钮将在设置状态之间切换。
  - 复制按钮将复制规则集。系统将提示您提供唯一的名称。
  - 删除按钮将删除规则集。系统将提示您确认此操作不可逆。
- 接收规则将列出其名称、状态、安全性和顺序。接收规则的修改选项包括：
  - 向上/向下箭头可在规则集中重新排序规则执行。
  - 复制按钮将创建所选规则的副本。系统将提示您提供唯一的名称。
  - 编辑按钮将打开所选规则，以便可以编辑其任何参数，如规则设置、收件人条件和操作。
  - 删除按钮将删除所选规则。系统将提示您确认此操作不可逆。
  - 创建规则按钮将允许您创建新规则，并将其添加到当前规则集。

操作选项

Amazon SES 电子邮件接收的每个接收规则都包含一组有序操作。本部分介绍每个操作类型的具体选项。

有以下操作类型：

- 添加标头操作
- 返回退回邮件响应操作
- 调用 Lambda 函数操作
- 交付到 S3 存储桶操作
- 发布到 Amazon SNS 主题操作
- 停止规则集操作
- 与 Amazon 集成 acti WorkMail on

添加标头操作

Add Header 操作向收到的电子邮件添加一个自定义标头。此操作通常仅与另一个操作结合使用。此操作具有以下选项。

- 标头名称 – 要添加的标头的名称。必须介于 1 到 50 个字符 (含) 之间，并且只能包含字母数字 (a-z、A-Z、0-9) 字符和短划线。
- 标头值 – 要添加的标头的值。必须少于 2048 个字符，并且不能包含换行符 (“\r”或“\n”)。

返回退回邮件响应操作

Bounce (退回邮件) 操作通过将退回邮件响应返回给发件人来拒绝电子邮件，并通过 Amazon SNS 通知您（可选）。此操作具有以下选项。

- SMTP Reply Code – 根据 RFC 5321 定义的 SMTP 回复代码。
- SMTP Status Code – 根据 RFC 3463 定义的 SMTP 增强状态代码。
- Message – 要包含在退回邮件中的用户可读文本。
- Reply Sender (回复发件人) – 退回电子邮件的发件人电子邮件地址。这是将从中发送退回邮件的地址。它必须使用 Amazon SES 验证。
- SNS Topic (SNS 主题) – 在退回电子邮件发送后可选择通知的 Amazon SNS 主题的名称或 ARN。亚马逊 SNS 话题 ARN 的一个例子是 arn: aws: sns: us-east-1:123456789012:。MyTopic您也可以在设置操作时通过选择 Create SNS Topic（创建 SNS 主题）来创建 Amazon SNS 主题。有关 Amazon SNS 主题的更多信息，请参阅 Amazon Simple Notification Service 开发人员指南。

> **Note**
>
> 您选择的 Amazon SNS 主题必须与您用于接收电子邮件的 Amazon SES 终端节点位于同一 AWS 区域。

您可以在这些字段中键入您自己的值，也可以选择模板，根据退回邮件的原因填充 SMTP Reply Code、SMTP Status Code 和 Message 字段的值。可用模板如下：

- Mailbox Does Not Exist – SMTP Reply Code = 550，SMTP Status Code = 5.1.1
- Message Too Large – SMTP Reply Code = 552，SMTP Status Code = 5.3.4
- Mailbox Full (邮箱已满) – SMTP 回复代码 = 552、SMTP 状态代码 = 5.2.2
- Message Content Rejected – SMTP Reply Code = 500，SMTP Status Code = 5.6.1
- Unknown Failure – SMTP Reply Code = 554，SMTP Status Code = 5.0.0
- Temporary Failure – SMTP Reply Code = 450，SMTP Status Code = 4.0.0

有关在字段中键入自定义值时可能使用的其他退回邮件代码，请参阅 RFC 3463。

调用 Lambda 函数操作

Lambda 操作通过 Lambda 函数调用您的代码，并通过 Amazon SNS 通知您（可选）。此规则操作具有以下选项和要求：

选项

- Lambda function (Lambda 函数) – Lambda 函数的 ARN。Lambda 函数 ARN 的一个例子是 arn: aws: lambda: us-east-1: account-id: function:。MyFunction
- Invocation type (调用类型) – Lambda 函数的调用类型。调用类型为RequestResponse意味着函数的执行会立即得到响应。调用类型 Event 意味着该函数是异步调用的。我们建议您使用 Event 调用类型，除非您的使用案例中必需同步执行。

  有一个 30 秒的 RequestResponse 调用超时时间。

  有关更多信息，请参阅 AWS Lambda 开发人员指南中的调用 Lambda 函数。
- SNS Topic (SNS 主题) – 在触发指定的 Lambda 函数时发出通知的 Amazon SNS 主题的名称或 ARN。亚马逊 SNS 话题 ARN 的一个例子是 arn: aws: sns: us-east-1:123456789012:。MyTopic有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的创建 Amazon SNS 主题。

要求

- 您选择的 Lambda 函数必须与您用于接收电子邮件的 Amazon SES 终端节点位于同一 AWS 区域。
- 您选择的 Amazon SNS 主题必须与您用于接收电子邮件的 Amazon SES 终端节点位于同一 AWS 区域。

编写 Lambda 函数

处理电子邮件时，可以异步调用 Lambda 函数（即使用 `Event` 调用类型）。传递给 Lambda 函数的事件对象将包含与入站电子邮件事件有关的元数据。您还可以使用元数据从 Amazon S3 存储桶访问消息内容。

如果需要实际控制邮件流，则必须同步调用 Lambda 函数（即使用 `RequestResponse` 调用类型），并且 Lambda 函数必须用两个参数调用 `callback` 方法：第一个参数是 `null`，第二个参数是 `disposition` 属性，它的值设置为 `STOP_RULE`、`STOP_RULE_SET` 或 `CONTINUE`。如果第二个参数为 `null` 或没有有效的 `disposition` 属性，则邮件流将继续，并处理接下来的操作和规则，这与 `CONTINUE` 相同。

例如，您可以通过在 Lambda 函数代码的结尾写入以下行来停止接收规则集：

```
callback( null, { "disposition" : "STOP_RULE_SET" });
```

有关 AWS Lambda 代码示例，请参阅Lambda 函数示例。有关高级使用案例的示例，请参阅使用案例示例。

输入格式

Amazon SES 以 JSON 格式向 Lambda 函数传送信息。顶级对象包含一个 Records 数组，其中包含属性 eventSource、eventVersion 和 ses。ses 对象包含 receipt 和 mail 对象，这些对象的格式与通知内容中所述的 Amazon SNS 通知中的格式完全相同。

Amazon SES 传递给 Lambda 的数据包括有关邮件的元数据，以及多个电子邮件标头。但是，不包含邮件的正文。

下面是 Amazon SES 提供给 Lambda 函数的输入结构的概括视图。

```
{
   "Records": [
      {
        "eventSource": "aws:ses",
        "eventVersion": "1.0",
        "ses": {
           "receipt": {
               <same contents as SNS notification>
            },
           "mail": {
               <same contents as SNS notification>
           }
         }
     }
   ]
}
```

返回值

您的 Lambda 函数可以通过返回以下值之一来控制邮件流：

- STOP_RULE – 当前接收规则中没有进一步操作需要处理，但可以处理后续的接收规则。
- STOP_RULE_SET – 没有后续操作或接收规则需要处理。
- CONTINUE 或任何其他无效值 – 这意味着可以处理后续操作和接收规则。

以下主题涵盖了传入邮件事件的示例、高级用例示例和 AWS Lambda 代码示例：

- 使用案例示例

- Lambda 函数示例

使用案例示例

以下示例简要介绍您可以设置的一些规则，以使用 Lambda 函数结果控制邮件流。为方便演示，这些示例多数使用 S3 操作作为结果。

使用案例 1：删除所有域中的垃圾邮件

本示例演示在您的所有域中删除垃圾邮件的全局规则。规则 2 和规则 3 表示您可以在所有域中删除垃圾邮件后应用特定于域的规则。

规则 1

收件人列表：空。因此，此规则将应用到您的所有已验证域下的所有收件人。

操作

1. 如果电子邮件为垃圾邮件，Lambda 操作 (同步) 返回 STOP_RULE_SET。否则返回 CONTINUE。请参阅Lambda 函数示例中用于删除垃圾邮件的示例 Lambda 函数。

规则 2

收件人列表：example1.com

操作

1. 任何操作。

规则 3

收件人列表：example2.com

操作

1. 任何操作。

使用案例 2：退回所有域中的垃圾邮件

本示例演示在您的所有域中退回垃圾邮件的全局规则。规则 2 和规则 3 表示您可以在所有域中退回垃圾邮件后应用特定于域的规则。

规则 1

收件人列表：空。因此，此规则将应用到您的所有已验证域下的所有收件人。

操作

1. 如果电子邮件为垃圾邮件，Lambda 操作 (同步) 返回 CONTINUE。否则返回 STOP_RULE。
2. 退回邮件操作 (“500 5.6.1. Message content rejected”)。
3. 停止操作。

规则 2

收件人列表：example1.com

操作

1. 任何操作

规则 3

收件人列表：example2.com

操作

1. 任何操作

使用案例 3：应用最具体的规则

本示例演示如何使用停止操作防止电子邮件由多条规则进行处理。在此示例中，您有针对一个具体地址的一条规则，以及针对该域中所有电子邮件地址的另一条规则。通过使用停止操作，匹配具体电子邮件地址规则的消息不会由适用于整个域的通用规则进行处理。

规则 1

收件人列表：user@example.com

操作

1. Lambda 操作 (异步)。
2. 停止操作。

规则 2

收件人列表：example.com

操作

1. 任何操作。

用例 4：将邮件事件记录到 CloudWatch

本示例演示在将邮件保存到 Amazon SES 之前，如何为经过系统的所有邮件保留审计日志。

规则 1

收件人列表：example.com

操作

1. 将事件对象写入日志的 Lambda 操作（异步）。 CloudWatch Lambda 函数示例日志中的 Lambda 函数示例。 CloudWatch
2. S3 操作。

使用案例 5：删除 DKIM 失败的邮件

本示例演示如何将所有传入电子邮件保存到 Amazon S3 存储桶，但仅将发送给特定电子邮件地址并通过 DKIM 的电子邮件发送到自动运行的电子邮件应用程序。

规则 1

收件人列表：example.com

操作

1. S3 操作。
2. 如果消息 DKIM 失败，Lambda 操作 (同步) 返回 `STOP_RULE_SET`。否则返回 `CONTINUE`。

规则 2

收件人列表：support@example.com

操作

1. 触发自动运行应用程序的 Lambda 操作 (异步)。

使用案例 6：基于主题行筛选邮件

此示例演示如何删除一个域中主题行包含词语“discount”的所有传入邮件，然后用一种方式处理面向自动系统的邮件，用其他方式处理该域中发送给所有其他收件人的邮件。

规则 1

收件人列表：example.com

操作

1. 如果主题行包含词语“discount”，Lambda 操作 (同步) 返回 `STOP_RULE_SET`。否则返回 `CONTINUE`。

规则 2

收件人列表：support@example.com

操作

1. 存储桶 1 的 S3 操作。
2. 触发自动运行应用程序的 Lambda 操作 (异步)。
3. 停止操作。

规则 3

收件人列表：example.com

操作

1. 存储桶 2 的 S3 操作。
2. Lambda 操作 (异步)，用于处理域中的其他电子邮件。

Lambda 函数示例

本主题包含用于控制邮件流的 Lambda 函数的示例。

示例 1：删除垃圾邮件

此示例会停止处理包含至少一个垃圾邮件指标的消息。

```
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'});
    }
};
```

示例 2：如果找到特定标头，则继续

只有在电子邮件包含特定标头值时，此示例才继续处理当前规则。

```
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'});
};
```

示例 3：从 Amazon S3 检索电子邮件

此示例从 Amazon S3 获取原始电子邮件并对其进行处理。

> **Note**
>
> - 您必须先使用 S3 操作将该电子邮件写入 Amazon S3。
> - 确保 Lambda 函数拥有 IAM 权限从 S3 存储桶中提取对象，请参阅这AWS 篇 re: Post 文章了解更多信息。
> - 默认的 Lambda 执行超时可能对您的工作流程来说太短，请考虑增加超时时间。

```
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'})
    }
};
```

示例 4：DMARC 身份验证失败时的退回邮件

如果传入电子邮件的 DMARC 身份验证失败，此示例将发送退回邮件消息。

> **Note**
>
> - 在使用此示例时，请将 `emailDomain` 环境变量的值设置为您的电子邮件接收域。
> - 确保 Lambda 函数拥有发送退回消息的 SES 身份的`ses:SendBounce`权限。

```
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'});
    }
};
```

交付到 S3 存储桶操作

传送到 S3 存储桶操作会将邮件传送到 S3 存储桶，并且可以选择通过 SNS 等方式向您发送通知。此操作具有以下选项。

- S3 存储桶 – 用于保存收到的电子邮件的 S3 存储桶的名称。您也可以在设置操作时通过选择创建 S3 存储桶来创建新的 S3 存储桶。Amazon SES 为您提供原始、未经修改的电子邮件，通常是多用途 Internet 邮件扩展（MIME）格式。有关 MIME 格式的更多信息，请参阅 RFC 2045。

  > **Important**
  >
  > - Amazon S3 存储桶必须存在于 SES the section called “电子邮件接收” 可用的区域；否则，您必须使用下面介绍的 IAM 角色选项。
  > - 当您将电子邮件保存到 S3 存储桶时，默认的最大电子邮件大小（包括标头）为 40 MB。
  > - SES 不支持通过已配置默认保留期的对象锁定启用上传至 S3 存储桶的接收规则。
  > - 如果通过指定您自己的 KMS 密钥在 S3 桶上应用加密，请确保使用完全限定的 KMS 密钥 ARN，而不是 KMS 密钥别名；使用别名可能导致以属于请求者而非桶管理员的 KMS 密钥对数据进行加密。请参阅使用加密进行跨账户操作。

- 对象键前缀 – 在 S3 存储桶中使用的可选键名称前缀。您可以通过键名称前缀以文件夹结构组织您的 S3 存储桶。例如，如果您使用 Email 作为对象键前缀，您的电子邮件将显示在 S3 存储桶中名为 Email 的文件夹中。
- 消息加密 – 在将收到的电子邮件发送到 S3 存储桶之前对其进行加密的选项。
- KMS 加密密钥 –（如果选择了消息加密，则可用。）在将电子邮件保存到 S3 存储桶之前，SES 应使用该密 AWS KMS 钥对电子邮件进行加密。您可以使用默认的 KMS 密钥或者您在 KMS 中创建的客户自主管理型密钥。

  > **Note**
  >
  > 您选择的 KMS 密钥必须与用于接收电子邮件的 SES 终端节点位于同一 AWS 区域。

  - 要使用默认 KMS 密钥，请在设置接收规则时在 SES 控制台中选择 aws/ses。如果使用 SES API，可以通过提供 `arn:aws:kms:REGION:AWSACCOUNTID:alias/aws/ses` 形式的 ARN 指定默认 KMS 密钥。例如，如果您的 AWS 账户 ID 为 123456789012，并且您想在 us-east-1 区域使用默认 KMS 密钥，则默认 KMS 密钥的 ARN 将为。`arn:aws:kms:us-`

east-1:123456789012:alias/aws/ses如果您使用默认 KMS 密钥，则不需要执行任何其他步骤来为 SES 提供使用它的权限。

- 要使用在 KMS 中创建的客户自主管理型密钥，请提供 KMS 密钥的 ARN，并确保在密钥策略中添加一条语句，为 SES 提供使用该密钥的权限。有关提供权限的更多信息，请参阅授予 Amazon SES 电子邮件接收的权限。

有关将 KMS 与 SES 结合使用的更多信息，请参阅《AWS Key Management Service Developer Guide》。如果您未在控制台或 API 中指定 KMS 密钥，则 SES 将不会加密您的电子邮件。

> **Important**
>
> 您的邮件将使用 S3 加密客户端通过 SES 进行加密，然后再提交到 S3 进行存储。它并不使用 S3 服务器端加密进行加密。这意味着，在从 S3 中检索电子邮件后，您必须使用 S3 加密客户端解密电子邮件，因为该服务没有使用 KMS 密钥进行解密的访问权限。此加密客户端可在适用于 Java 的 AWS SDK和适用于 Ruby 的 AWS SDK中获取。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南。

- IAM 角色 – SES 用来访问传送到 S3 操作（Amazon S3 存储桶、SNS 主题和 KMS 密钥）中的资源的 IAM 角色。如果未提供，则需要明确授予 SES 单独访问每个资源的权限，请参阅授予 Amazon SES 电子邮件接收的权限。

  如果您想向一个位于 SES 电子邮件接收不可用区域的 S3 存储桶写入数据，则您必须使用一个 IAM 角色，该角色要能够将写入 S3 权限策略作为角色的内联策略。您可以直接从控制台应用此操作的权限策略：

  1. 在 IAM 角色字段中选择创建新角色，然后输入名称，接着选择创建角色。（此角色的 IAM 信任策略将在后台自动生成。）
  2. 由于 IAM 信任策略是自动生成的，因此您只需将操作的权限策略添加到角色中 - 选择 IAM 角色字段下的查看角色即可打开 IAM 控制台。
  3. 在权限选项卡下，选择添加权限，然后选择创建内联策略。
  4. 在指定权限页面上，在策略编辑器中选择 JSON。
  5. 将权限策略从 适用于 S3 操作的 IAM 角色权限 复制并粘贴到策略编辑器中，然后将红色文本中的数据替换为您自己的数据。（请务必删除编辑器中的所有示例代码。）
  6. 选择下一步。
  7. 选择创建策略，查看并创建您的 IAM 角色权限策略。

8. 选择已打开 SES 创建规则 – 添加操作页面的浏览器选项卡，然后继续执行创建规则的其余步骤。

- SNS 主题 – 在电子邮件保存到 S3 存储桶时发出通知的 Amazon SNS 主题的名称或 ARN。SNS 话题 ARN 的一个例子是 arn: aws: sns: us-east-1:12345678 9012:。MyTopic您也可以在设置操作时通过选择创建 SNS 主题来创建 SNS 主题。有关 SNS 主题的更多信息，请参阅《Amazon Simple Notification Service Developer Guide》。

> **Note**
>
> - 您选择的 SNS 主题必须与用于接收电子邮件的 SES 终端节点位于同一 AWS 区域。
> - 仅对与 SES 接收规则关联的 SNS 主题使用客户自主管理型 KMS 密钥加密，因为您需要编辑 KMS 密钥策略以允许 SES 发布到 SNS。这与 AWS 管理的 KMS 密钥策略不同，后者按设计不可编辑。

发布到 Amazon SNS 主题操作

SNS 操作使用 Amazon SNS 通知发布邮件。该通知包含完整的电子邮件内容。此操作具有以下选项。

- SNS Topic (SNS 主题) – 发布电子邮件的 Amazon SNS 主题的名称或 ARN。Amazon SNS 通知将包含原始、未经修改的电子邮件副本，通常是多用途 Internet 邮件扩展（MIME）格式。有关 MIME 格式的更多信息，请参阅 RFC 2045。

> **Important**
>
> 如果您选择通过 Amazon SNS 通知接收您的电子邮件，则最大电子邮件大小（包括标头）为 150KB。大于此大小的电子邮件将被退回。如果您预计电子邮件将大于此大小，请改为将电子邮件保存到 Amazon S3 存储桶。

亚马逊 SNS 话题 ARN 的一个例子是 arn: aws: sns: us-east-1:123456789012:。MyTopic您也可以在设置操作时通过选择 Create SNS Topic（创建 SNS 主题）来创建 Amazon SNS 主题。有关 Amazon SNS 主题的更多信息，请参阅 Amazon Simple Notification Service 开发人员指南。

> **Note**
>
> - 您选择的 Amazon SNS 主题必须与您用于接收电子邮件的 Amazon SES 终端节点位于同一 AWS 区域。

- 仅对与 SES 接收规则关联的 SNS 主题使用客户自主管理型 KMS 密钥加密，因为您需要编辑 KMS 密钥策略以允许 SES 发布到 SNS。这与 AWS 管理的 KMS 密钥策略不同，后者按设计不可编辑。

- Encoding (编码) – Amazon SNS 通知中的电子邮件所使用的编码。UTF-8 更易于使用，但如果邮件使用其他编码格式进行编码，可能不会保留所有特殊字符。Base64 保留所有特殊字符。有关 UTF-8 和 Base64 的信息，请分别参阅 RFC 3629 和 RFC 4648。

在您接收电子邮件时，Amazon SES 将执行活动接收规则集中的规则。您可以使用 Amazon SNS 配置接收规则以向您发送通知。您的接收规则可以发送两种不同类型的通知：

- 从 SNS 操作发送的通知 – 当您将 SNS 操作添加到接收规则时，它将发送有关电子邮件以及电子邮件内容的信息。如果邮件为 150KB 或更小，则此通知类型还包括电子邮件的完整 MIME 正文。
- 从其他操作类型发送的通知-当您将任何其他操作类型（包括退回、Lamb d a 、停止规则集WorkMail或操作）添加到接收规则时，您可以选择指定 Amazon SNS 主题。如果是这样，您将在执行这些操作时收到通知。这些通知包含有关电子邮件的信息，但不包含电子邮件的内容。

以下主题介绍这些通知的内容，并提供每种通知类型的示例：

- Amazon SES 电子邮件接收通知的内容
- Amazon SES 电子邮件接收通知的示例

Amazon SES 电子邮件接收通知的内容

所有接收电子邮件的通知都以对象表示法 (JSON) 格式发布到亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 主题 JavaScript 。

有关示例通知，请参阅 通知示例。

目录

  - spfVerdict 对象
  - virusVerdict 对象
- 邮件对象
  - commonHeaders 对象

顶级 JSON 对象

顶级 JSON 对象包含以下字段。

字段名称	描述
notificationType	通知类型。对于此通知类型，值始终为 Received。
receipt	包含电子邮件传送信息的对象。
mail	包含与通知关联的电子邮件的相关信息的对象。
content	包含原始、未修改电子邮件的字符串，通常是多用途 Internet 邮件扩展（MIME）格式。有关 MIME 格式的更多信息，请参阅 RFC 2045。 **Note** 仅当通知由 SNS 操作触发时才包含此字段。所有其他操作触发的通知不包含此字段。

接收对象

receipt 对象包含以下字段。

字段名称	描述
action	封装已执行操作的信息的对象。有关可能的值的列表，请参阅操作对象。

字段名称	描述
dkimVerdict	表示 DomainKeys 已识别邮件 (DKIM) 检查是否通过的对象。有关可能的值的列表，请参阅dkimVerdict 对象。
dmarcPolicy	指示发送域的基于域的邮件身份验证、报告和合规性（DMARC）设置。此字段仅当邮件未通过 DMARC 身份验证时出现。 此字段的值可能为： • none：发送域的所有者请求不对未通过 DMARC 身份验证的邮件执行任何特定操作。 • quarantine ：发送域的所有者请求接收方将未通过 DMARC 身份验证的邮件视为可疑邮件。 • reject：发送域的所有者请求拒绝未通过 DMARC 身份验证的邮件。
dmarcVerdict	用于指示是否已通过基于域的消息身份验证、报告和合规性（DMARC）检查的对象。有关可能的值的列表，请参阅dmarcVerdict 对象。
processingTimeMillis	用于指定从 Amazon SES 收到消息到触发操作所花费的时间（以毫秒为单位）的字符串。
recipients	符合有效接收规则的收件人（具体来说，信封 RCPT TO 地址）。此处列出的地址可能不同于the section called “邮件对象”中的 destination 字段所列出的地址。
spamVerdict	用于指示消息是否为垃圾邮件的对象。有关可能的值的列表，请参阅spamVerdict 对象。
spfVerdict	用于指示是否已通过发件人策略框架（SPF）检查的对象。有关可能的值的列表，请参阅spfVerdict 对象。

字段名称	描述
timestamp	用于指定触发操作的限定日期和时间（采用 ISO 8601 格式）的字符串。
virusVerdict	用于指示消息是否包含病毒的对象。有关可能的值的列表，请参阅virusVerdict 对象。

操作对象

action 对象包含以下字段。

字段名称	描述
type	用于指示已执行的操作类型的字符串。可能的值包括 S3、SNS、Bounce、Lambda、Stop 和 WorkMail。
topicArn	包含发布通知的 Amazon SNS 主题的 Amazon Resource Name (ARN) 的字符串。
bucketName	包含发布消息的 Amazon S3 存储桶名称的字符串。仅 S3 操作类型存在此字段。
objectKey	包含在 Amazon S3 存储桶中唯一标识电子邮件的名称的字符串。它与 the section called “邮件对象”中的 messageId 相同。仅 S3 操作类型存在此字段。
smtpReplyCode	包含根据 RFC 5321 定义的 SMTP 回复代码的字符串。仅退回邮件操作类型存在此字段。
statusCode	包含根据 RFC 3463 定义的 SMTP 增强状态代码的字符串。仅退回邮件操作类型存在此字段。
message	包含退回邮件信息中所包括的用户可读文本的字符串。仅退回邮件操作类型存在此字段。

字段名称	描述
sender	包含退回电子邮件的发件人电子邮件地址的字符串。此为发送退回邮件消息的地址。仅退回邮件操作类型存在此字段。
functionArn	包含已触发的 Lambda 函数的 ARN 的字符串。仅 Lambda 操作类型存在此字段。
invocationType	包含 Lambda 函数的调用类型的字符串。可能的值为 RequestResponse 和 Event。仅 Lambda 操作类型存在此字段。
organizationArn	包含亚马逊 WorkMail组织的 ARN 的字符串。仅适用于 WorkMail 操作类型。

dkimVerdict 对象

dkimVerdict 对象包含以下字段。

字段名称	描述
status	包含 DKIM 裁决的字符串。可能的值有： • PASS：邮件已通过 DKIM 身份验证。 • FAIL：邮件未通过 DKIM 身份验证。 • GRAY：邮件未经 DKIM 签名，或者发件人域和 DKIM 签名域不匹配。 • PROCESSING_FAILED ：出现了阻止 Amazon SES 检查 DKIM 签名的问题。例如，DNS 查询失败或 DKIM 签名标头格式不正确。

dmarcVerdict 对象

dmarcVerdict 对象包含以下字段。

字段名称	描述
`status`	包含 DMARC 裁决的字符串。可能的值有： • `PASS`：邮件已通过 DMARC 身份验证。 • `FAIL`：邮件未通过 DMARC 身份验证。 • `GRAY`：至少有一个 SPF 或 DKIM 通过身份验证，但发送域没有 DMARC 策略或使用 p=none 策略。 • `PROCESSING_FAILED` ：出现了阻止 Amazon SES 提供 DMARC 裁决的问题。

spamVerdict 对象

`spamVerdict` 对象包含以下字段。

字段名称	描述
`status`	包含垃圾邮件扫描结果的字符串。可能的值有： • `PASS`：垃圾邮件扫描确定邮件不太可能包含垃圾邮件。 • `FAIL`：垃圾邮件扫描确定邮件可能包含垃圾邮件。 • `GRAY`：Amazon SES 已扫描电子邮件，但没把握确定它是否为垃圾邮件。 • `PROCESSING_FAILED` ：Amazon SES 无法扫描电子邮件。例如，电子邮件不是有效的 MIME 消息。

spfVerdict 对象

`spfVerdict` 对象包含以下字段。

字段名称	描述
status	包含 SPF 裁决的字符串。可能的值有： • PASS：邮件已通过 SPF 身份验证。 • FAIL：邮件未通过 SPF 身份验证。 • GRAY：SPF 结果为 none、softfail 或 neutral。 • PROCESSING_FAILED ：出现了阻止 Amazon SES 检查 SPF 记录的问题。例如，DNS 查询失败。

virusVerdict 对象

virusVerdict 对象包含以下字段。

字段名称	描述
status	包含病毒扫描结果的字符串。可能的值有： • PASS：邮件不含病毒。 • FAIL：邮件包含病毒。 • GRAY：Amazon SES 已扫描电子邮件，但没把握确定它是否包含病毒。 • PROCESSING_FAILED ：Amazon SES 无法扫描电子邮件的内容。例如，电子邮件不是有效的 MIME 消息。

邮件对象

mail 对象包含以下字段。

字段名称	描述
destination	传入电子邮件的 MIME 标头中的所有收件人地址（包括 To: (收件人:) 和 CC: (抄送:) 收件人）的完整列表。
messageId	包含 Amazon SES 分配给该电子邮件的唯一 ID 的字符串。如果电子邮件已送达 Amazon S3，则消息 ID 同时也是用于向 Amazon S3 存储桶写入消息的 Amazon S3 对象键。
source	包含发送电子邮件的电子邮件地址 (具体来说，信封 MAIL FROM 地址) 的字符串。
timestamp	包含电子邮件接收时间的字符串，格式为 ISO8601。
headers	Amazon SES 标头和自定义的标头。每个标头具有以下字段：name 和 value。
commonHeaders	所有电子邮件的共有标头。每个标头具有以下字段：name 和 value。
headersTruncated	指定通知中的标头是否被截断，如果标头大于 10KB，则会发生截断。可能的值为 true 和 false。

commonHeaders 对象

commonHeaders 对象可以包含下表中显示的字段。此对象中存在的字段取决于传入电子邮件中存在的字段。

字段名称	描述
messageId	原始邮件的 ID。
date	Amazon SES 收到此邮件的日期和时间。

字段名称	描述
to	这些区域有：To 电子邮件的标题。
cc	这些区域有：CC 电子邮件的标题。
bcc	这些区域有：BCC 电子邮件的标题。
from	这些区域有：From 电子邮件的标题。
sender	这些区域有：Sender 电子邮件的标题。
returnPath	这些区域有：Return-Path 电子邮件的标题。
replyTo	这些区域有：Reply-To 电子邮件的标题。
subject	这些区域有：Subject 电子邮件的标题。

Amazon SES 电子邮件接收通知的示例

本节包括以下类型的通知的示例：

- 因 SNS 操作而发送的通知。
- 因另一种类型的操作而发送的通知（警报通知）。

SNS 操作的通知

本部分包含 SNS 操作通知的示例。与之前显示的警报通知不同，它包括一个包含电子邮件的 content 部分，电子邮件通常是多用途 Internet 邮件扩展（MIME）格式。

```
{
  "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"
}
```

警报通知

本部分包含可由 S3 操作触发的 Amazon SNS 通知的示例。Lambda 操作、退回邮件操作、停止操作和 WorkMail 操作触发的通知非常相似。虽然通知包含有关电子邮件的信息，但它不包含电子邮件内容本身。

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

停止规则集操作

Stop (停止) 操作终止对接收规则集的评估，并会通过 Amazon SNS 通知您（可选）。此操作具有以下选项。

- SNS Topic (SNS 主题) – 在停止操作执行完毕时发出通知的 Amazon SNS 主题的名称或 ARN。亚马逊 SNS 话题 ARN 的一个例子是 arn: aws: sns: us-east-1:123456789012:。MyTopic您也可以在设置操作时通过选择 Create SNS Topic（创建 SNS 主题）来创建 Amazon SNS 主题。有关 Amazon SNS 主题的更多信息，请参阅 Amazon Simple Notification Service 开发人员指南。

> **Note**
>
> 您选择的 Amazon SNS 主题必须与您用于接收电子邮件的 Amazon SES 终端节点位于同一 AWS 区域。

与 Amazon 集成 acti WorkMail on

该WorkMail操作与 Amazon 集成 WorkMail。如果亚马逊 WorkMail 负责处理您的所有电子邮件，您通常不会直接使用此操作，因为设置由亚马逊 WorkMail 负责。此操作具有以下选项。

- 组织 ARN — 亚马逊组织的 ARN。 WorkMailAmazon WorkMail 组织 ARNs 采用以下形式`arn:aws:workmail:`*`region`*`:`*`account_ID`*`:organization/`*`organization_ID`*，其中：
  - `region`是您使用亚马逊 SES 和亚马逊的区域 WorkMail。(您必须从同一区域使用它们。) 例如，us-east-1。
  - `account_ID`是 AWS 账户 ID。您可以在 AWS 管理控制台的 “帐户” 页面上找到您的 AWS 账户 ID。
  - `organization_ID`是 Amazon 在您创建组织时 WorkMail 生成的唯一标识符。您可以在 Amazon WorkMail 控制台的组织设置页面上找到组织 ID。

  完整的亚马逊 WorkMail 组织 ARN 的一个例子是 arn: aws: workmail: us-east-1 :123456789012: organization/m-68755160c4cb4e29a2b2f8f8f b58f359d7。有关亚马逊 WorkMail 组织的信息，请参阅《亚马逊 WorkMail 管理员指南》。
- SNS 主题 — 亚马逊采取行动时要通知的亚马逊 SNS 主题的名称或 ARN。 WorkMail 亚马逊 SNS 话题 ARN 的一个例子是 arn: aws: sns: us-east-1:123456789012:。MyTopic您也可以在设置操作时通过选择 Create SNS Topic（创建 SNS 主题）来创建 Amazon SNS 主题。有关 Amazon SNS 主题的更多信息，请参阅 Amazon Simple Notification Service 开发人员指南。

  > **Note**
  >
  > 您选择的 Amazon SNS 主题必须与您用于接收电子邮件的 Amazon SES 终端节点位于同一 AWS 区域。

> **Note**
>
> Amazon SES 仅支持在可用的地区 WorkMail 执行操作。 WorkMail 请参阅中的 Amazon WorkMail 终端节点和配额 AWS 一般参考。

创建 IP 地址筛选条件控制台演练

本节将指导您使用 Amazon SES 控制台设置 IP 地址筛选条件。IP 地址筛选条件允许您提供广泛的控制级别。这些 IP 筛选条件允许您显式阻止或允许来自特定 IP 地址或 IP 地址范围的所有邮件。

或者，您可以使用 `CreateReceiptFilter` API 创建一个 IP 地址筛选条件，如 Amazon Simple Email Service API 参考中所述。

> **Note**
>
> 如果您只希望接收来自有限已知 IP 地址列表的邮件，则请设置包含 `0.0.0.0/0` 的阻止列表，并设置包含所信任 IP 地址的允许列表。原定设置情况下，此配置会阻止全部 IP 地址，并仅允许来自您显式指定的 IP 地址的邮件。

先决条件

在继续使用 IP 地址筛选条件设置基于收件人的电子邮件控制之前，必须满足以下先决条件：

1. 首先需要在 Amazon SES 中创建并验证域身份。
2. 接下来，您需要通过将 MX 记录发布域的 DNS 设置来指定哪些邮件服务器可以接收域的邮件。（MX 记录应指接收您使用 Amazon SES 所在 AWS 地区的邮件的 Amazon SES 终端节点。）

创建 IP 地址筛选器

使用控制台创建 IP 地址筛选器

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，选择电子邮件接收。
3. 选择 IP 地址筛选条件选项卡。
4. 选择 Create Filter。
5. 输入筛选条件的唯一名称，该字段的图例将指示语法要求。（该名称必须少于 64 个字符，且只能包含字母数字、连字符 (-)、下划线 (_) 和句点 (.) 字符。名称必须以字母或数字开头和结尾。）
6. 输入 IP 地址或 IP 地址范围，该字段的图例将给出无类域间路由 (CIDR) 语法指定的示例。（单个 IP 地址如：10.0.0.1。IP 地址范围如 10.0.0.1/24。有关 CIDR 表示法的详细信息，请参阅 RFC 2317。）

7. 选择阻止或允许单选按钮来选择策略类型。
8. 选择创建筛选条件。
9. 如果要添加另一个 IP 筛选条件，请选择创建筛选条件，然后对要添加的每个额外筛选条件重复前面的步骤。
10. 如果要删除 IP 地址筛选条件，请选择该筛选条件，然后选择删除按钮。

查看 Amazon SES 电子邮件接收的指标

如果您已在 Amazon SES 中启用电子邮件接收功能，并且已经为电子邮件创建了接收规则，则可以使用 Amazon 查看这些接收规则集和规则的指标 CloudWatch。

在 CloudWatch 控制台中，您可以在 “指标” > “所有指标” > “SES” > “接收规则集指标” 和 “接收规则指标” 下找到指标。

> **Note**
>
> 如果您尚未出现以下情况，则接收规则集指标和接收规则指标将不会显示在 SES 下：
>
> - 已启用电子邮件接收
> - 创建了任何接收规则
> - 收到了任何符合您的任何规则的邮件。

可供使用的邮件指标如下：

- 邮件接收

范围	指标	描述	维度
接收规则集指标	Received	SES 成功收到一封至少包含一条适用规则的邮件。该指标的值只能为 1。	RuleSetName
接收规则指标	Received	SES 成功收到一封邮件，并将尝试处理适用的规则。该指标的值只能为 1。	RuleName

- 邮件发布

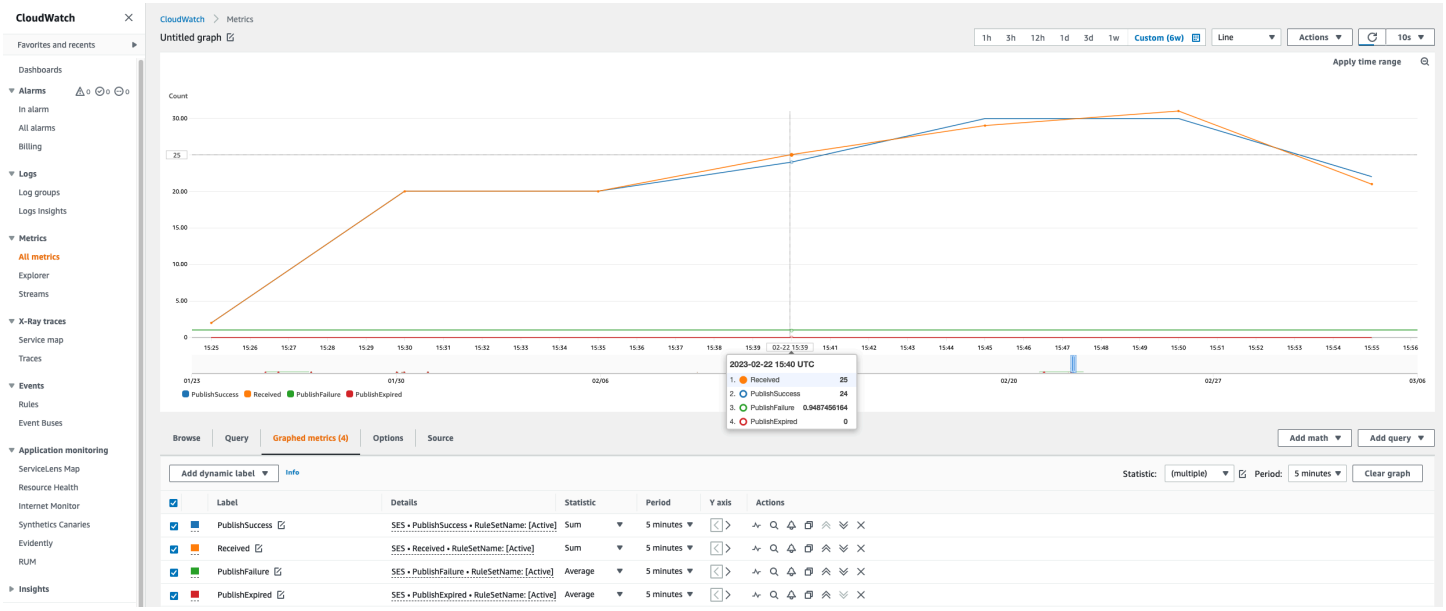
范围	指标	描述	维度
接收规则集指标	PublishSuccess	SES 成功执行了规则集内适用的所有规则。	RuleSetName
接收规则指标	PublishSuccess	SES 成功执行了适用于接收邮件的规则。	RuleName
接收规则集指标	PublishFailure	SES 在尝试在规则集内执行规则时遇到了错误，将重试执行。	RuleSetName
接收规则指标	PublishFailure	SES 在尝试执行规则中的操作时遇到错误，视错误而定，可能会重试执行。	RuleName
接收规则集指标	PublishExpired	SES 将不再重试执行规则，因为规则在 36 小时内未能成功执行，或者遇到了不可重试的错误。	RuleSetName
接收规则指标	PublishExpired	SES 将不再重新尝试执行规则的操作，因为操作在 36 小时内未能成功执行。	RuleName

Note

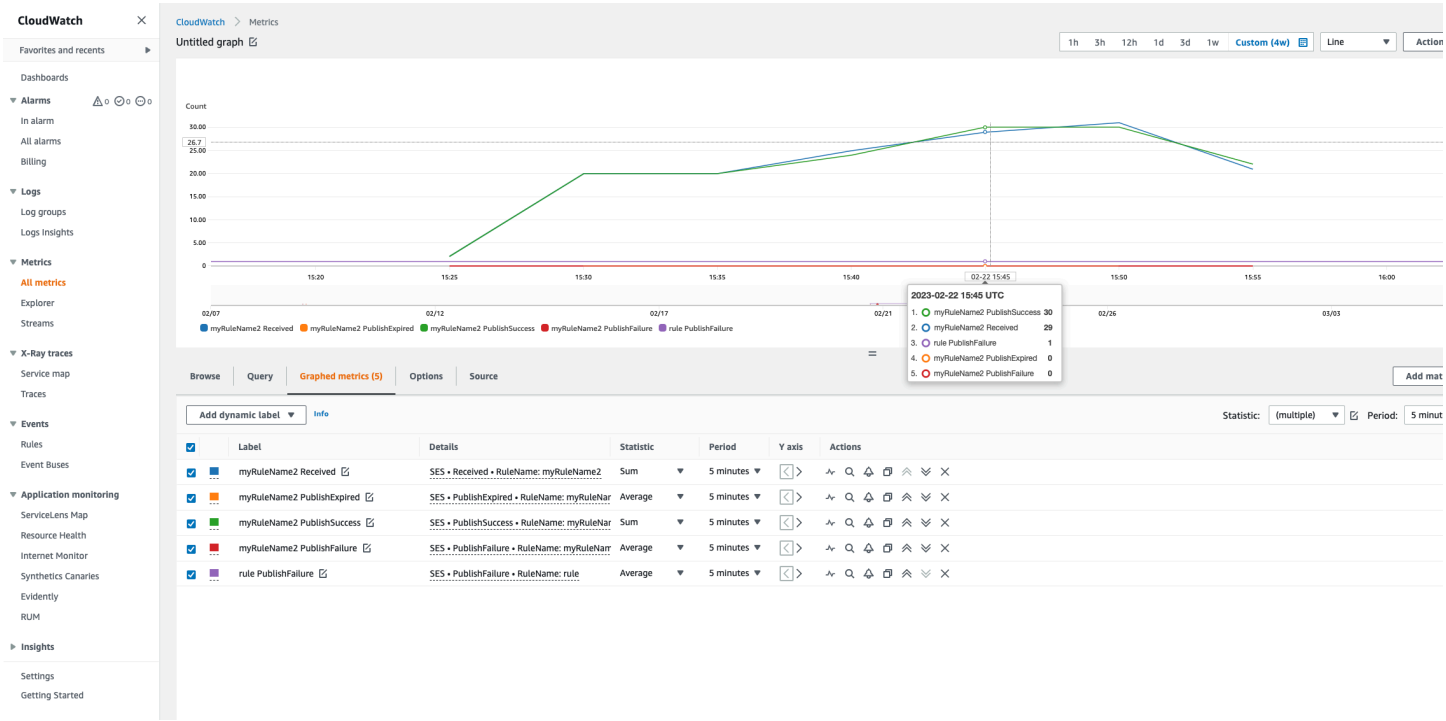
- 在前面的表格中，术语适用意味着发件人未被 IP 筛选条件列入黑名单，也不在 SES 的内部黑名单中，并且该规则具有匹配的收件人条件和匹配的 TLS 策略。
- 例如，如果您删除或撤消对于 Amazon S3 桶、Amazon SNS 主题或 Lambda 函数的权限，而某条接收规则中的一个操作已配置为使用它们，则可能发生发布失败错误。
- 由于一次只能有一个规则集处于活动状态，因此 SES 会针对在您选择的时间范围内处于活动状态的所有规则集发布一个显示为RuleSetName：[活动] 的汇总指标 CloudWatch。这样做的优势是，您可以自由更改规则集，而无需对警报设置进行任何更改。

> **Important**
>
> 您修复接收规则集所做的更改将仅应用于 Amazon SES 在更新后收到的电子邮件。在评估电子邮件时，始终使用收到电子邮件时采用的接收规则集。

SES 接收规则集的指标显示在 CloudWatch控制台中。



CloudWatch控制台中显示的 SES 接收规则的指标。



Amazon SES 中已验证的身份

在 Amazon SES 中，已验证的身份是您用来发送或接收电子邮件的域或电子邮件地址。在使用 Amazon SES 发送电子邮件之前，您必须创建并验证将要用作 From (发件人)、Source (源)、Sender (发件人) 或 Return-Path (返回路径) 地址的每个身份。通过 Amazon SES 验证身份可以确认您拥有该身份，并有助于防止未经授权的使用。

如果您的账户仍在 Amazon SES 沙盒中，则还必须验证您计划向其发送电子邮件的所有电子邮件地址，除非您要向 Amazon SES 邮箱模拟器提供的测试收件箱发送电子邮件。有关更多信息，请参阅 the section called “手动使用邮箱模拟器”。

您可以使用 Amazon SES 控制台或 Amazon SES API 来创建身份。身份验证过程取决于您选择创建的身份类型。

> **Tip**
>
> 如果您是首次使用 SES，可以使用入门向导来创建和验证您的第一个身份（电子邮件地址或域）。

内容

在 Amazon SES 中创建和验证身份

在 Amazon SES 中，您可以在域级别创建身份，也可以创建电子邮件地址身份。这些身份类型并不相互排斥。在大多数情况下，创建域身份就不需要再创建和验证单个电子邮件地址身份，除非您希望对特定电子邮件地址应用自定义配置。无论您是创建域并使用基于域的电子邮件地址，还是创建单独的电子邮件地址，这两种方法都有好处。您选择哪种方法取决于您的具体需求，如下所述。

创建和验证电子邮件地址身份是 SES 中最快的开启方式，但在域级别验证身份也有益处。当您验证电子邮件地址身份时，只有该电子邮件地址可用于发送邮件，但当您验证域身份时，您可以从已验证域的任何子域或电子邮件地址发送电子邮件，而无需单独验证每一个。例如，如果您创建并验证名为

example.com 的域身份，则无需为 a.example.com、a.b.example.com 创建单独的子域身份，也无需为 user@example.com、user@a.example.com 等创建单独的电子邮件地址身份。

然而，请记住，使用从其域继承的验证的电子邮件地址身份仅限于直接发送电子邮件。如果想要进行更高级的发送，您还必须明确地将其验证为电子邮件地址身份。高级发送包括将电子邮件地址与配置集、委托发送的策略授权以及覆盖域设置的配置一起使用。

为了帮助阐明上述验证继承和电子邮件发送功能，下表对域/电子邮件地址验证的每种组合进行了分类，并列出了每种组合的继承、发送级别和显示状态：

	仅验证域	仅验证电子邮件地址	验证域和电子邮件地址
继承级别	子域和电子邮件地址继承父域的验证。	明确验证电子邮件地址。	• 子域继承父域的验证。 • 明确验证电子邮件地址。
发送级别	电子邮件地址限制为直接发送电子邮件。	电子邮件地址可用于高级发送*。	电子邮件地址可用于高级发送*。
显示的状态	控制台/API 状态： • 域/子域 = 已验证 • 电子邮件地址 = 未验证。	控制台/API 状态： • 电子邮件地址 = 已验证	控制台/API 状态： • 域/子域 = 已验证 • 电子邮件地址 = 已验证。

*高级发送包括将电子邮件地址与配置集、委托发送的策略授权以及覆盖域设置的配置一起使用。

要从同一个域名或多个电子邮件地址发送电子邮件 AWS 区域，您必须为每个区域创建并验证单独的身份。在每个区域中，您最多可以验证 10,000 个身份。

在创建和验证域与电子邮件地址身份时，请考虑以下各项：

- 您可以从已验证域的任意子域或电子邮件地址发送电子邮件，而无需单独验证每个子域或电子邮件地址。例如，如果您为 example.com 创建并验证身份，则不需要为 a.example.com、a.b.example.com、user@example.com、user@a.example.com 等创建单独的身份。

- 按照 RFC 1034 中的规定，每个 DNS 标签最多可包含 63 个字符，域名总长度不得超过 255 个字符。
- 如果您验证共享一个根域的域、子域或电子邮件地址，则身份设置（例如反馈通知）将应用于您已验证的最精细级别。
  - 已验证的电子邮件地址身份设置会覆盖已验证的域身份设置。
  - 已验证子域身份设置会覆盖已验证的域身份设置，较低级别的子域设置会覆盖较高级别的子域设置。

    例如，假设您验证了 user@a.b.example.com、a.b.example.com、b.example.com 和 example.com。这些已验证的身份设置将用于以下场景：
    - 从 user@example.com (该电子邮件地址未经专门验证) 发送的电子邮件将使用 example.com 的设置。
    - 从 user@a.b.example.com (该电子邮件地址经过专门验证) 发送的电子邮件将使用 user@a.b.example.com 的设置。
    - 从 user@b.example.com (该电子邮件地址未经专门验证) 发送的电子邮件将使用 b.example.com 的设置。
- 您可以向已验证的电子邮件地址添加标签而无需执行额外的验证步骤。要向电子邮件地址添加标签，请在账户名称和“at”符号 (@) 之间添加加号 (+)，后跟文本标签。例如，如果您已验证 sender@example.com，则可以使用 sender+myLabel@example.com 作为您的电子邮件的“From”或“Return-Path”地址。您可以使用此特征来实施可变信封退回路径 (VERP)。然后，您可以使用 VERP 检测无法送达的电子邮件地址并从您的邮件列表中删除它们。
- 域名不区分大小写。如果您验证了 example.com，则也可以从 EXAMPLE.com 发送电子邮件。
- 电子邮件地址区分大小写。如果您验证了 sender@EXAMPLE.com，将无法从 sender@example.com 发送电子邮件，除非同时验证了 sender@example.com。
- 在每个身份中 AWS 区域，您可以验证多达 10,000 个身份（域名和电子邮件地址的任意组合）。

> **Tip**
>
> 如果您是首次使用 SES，可以使用入门向导来创建和验证您的第一个身份（电子邮件地址或域）。

内容

- 与您的 DNS 提供商一起验证 DKIM 域身份
- 创建电子邮件地址身份
- 验证电子邮件地址身份
- 创建和验证身份并同时分配默认配置集
- 使用自定义验证电子邮件模板

创建域身份

创建域名身份的一部分是配置其基于 DKIM 的验证。 DomainKeys识别邮件 (DKIM) 是一种电子邮件身份验证方法，Amazon SES 使用它来验证域名所有权，接收邮件服务器使用它来验证电子邮件的真实性。您可以选择使用 Easy DKIM 或自带 DKIM (BYODKIM) 来配置 DKIM，根据您的选择，您必须按照以下方式配置私有密钥的签名密钥长度：

- Easy DKIM - 接受 Amazon SES 默认的 2048 位，或选择 1024 位进行覆盖。
- BYODKIM - 私有密钥长度必须至少为 1024 位，最多为 2048 位。

有关 DKIM 签名密钥长度以及如何更改密钥长度的详情，请参阅 the section called “DKIM 签名密钥长度”。

以下过程介绍如何使用 Amazon SES 控制台来创建域身份。

- 如果您已创建域且只需要验证域，请跳到本页面上的过程the section called “验证域身份”。

创建域身份

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择身份。
3. 选择创建身份。
4. 在身份详细信息下，选择域作为要创建的身份类型。您必须有权访问域的 DNS 设置才能完成验证过程。
5. 在域字段中输入域或子域的名称。

> **Tip**
>
> 如果您的域是 www.example.com，请输入 example.com 作为您的域。请勿包含“www.”部分，如果包含，域验证过程将不会成功。

6. （可选）如果要分配默认配置集，请选中该复选框。

   1. 对于默认配置集，选择要分配给身份的现有配置集。如果尚未创建任何配置集，请参阅 配置集。

      > **Note**
      >
      > 只有当发送时未指定其他配置集时，Amazon SES 才会按默认应用分配的配置集。如果指定了配置集，Amazon SES 将应用指定的配置集来代替默认的配置集。

7. （可选）如果要使用自定义 MAIL FROM 域，请选中该复选框并完成以下步骤。有关更多信息，请参阅 the section called “使用自定义 MAIL FROM 域”。

   1. 对于 MAIL FROM Domain (MAIL FROM 域)，输入要用作 MAIL FROM 子域的域。此域必须是您要验证的域身份的子域。MAIL FROM 域不应是您从中发送电子邮件的域。
   2. 对于 MX 故障时的行为，请指明如果 Amazon SES 在发送时找不到所需的 MX 记录，则应采取哪些操作。请选择以下选项之一：
      - 使用默认 MAIL FROM 域 - 如果自定义 MAIL FROM 域的 MX 记录未正确设置，则 Amazon SES 将使用 amazonses.com 的子域。子域根据您使用 Amazon SES 的 AWS 区域 而变化。
      - 拒绝邮件 – 如果未正确设置自定义 MAIL FROM 域的 MX 记录，那么 Amazon SES 将返回 `MailFromDomainNotVerified` 错误。如果选择此选项，则您尝试从此域发送的电子邮件将被自动拒绝。
   3. 对于向 Route53 发布 DNS 记录，如果您的域通过 Amazon Route 53 托管，则有选择权让SES 在创建时通过勾选 Enabled 来发布关联的 TXT 和 MX 记录。如果宁愿稍后发布这些记录，请清除 Enabled 复选框。（您可以稍后再来通过编辑身份将记录发布到 Route 53 - 请参阅 the section called “使用控制台编辑身份”。）

8. （可选）要配置基于 DKIM 的自定义验证（使用 2048 位唱歌长度的 E asy DKIM 的 SES 默认设置除外），请在 “验证您的域名” 下展开 “高级 DKIM 设置”，然后选择要配置的 DKIM 类型：

   a. Easy DKIM：

i. 在 “身份类型” 下，选择 Easy DKIM。

ii. 在 DKIM 签名密钥长度字段中，选择 RSA_2048_BIT 或 RSA_1024_BIT。

iii. 对于将 DNS 记录发布到 Route53，如果您的域通过 Amazon Route 53 托管，则有选择权让SES在创建时通过勾选 Enabled (已启用)来发布关联的 CNAME 记录。如果宁愿稍后发布这些记录，请清除 Enabled 复选框。（您可以稍后再来通过编辑身份将记录发布到 Route 53 - 请参阅 the section called “使用控制台编辑身份”。）

b. 确定性 Easy DKIM（DEED）：

> **Tip**
>
> 如果您要创建全局（副本）身份，则应使用这种形式的 DKIM。DEED 将使用来自父区域的现有同名身份的 Easy DKIM 设置并签署新身份，而无需您执行其他 DNS 设置。欲了解更多信息，请参阅 DE ED。

i. 在 “身份类型” 下，选择 “确定性 Easy D KIM”。

ii. 从 “父区域” 下拉菜单中，选择一个父区域，该身份所在的 Easy DKIM 签名身份与您为全球（副本）身份输入的名称相同。（您的副本区域默认为您登录 SES 控制台时使用的区域。）

c. 提供 DKIM 身份验证令牌 (BYODKIM)：

i. 确保您已生成公有-私有密钥对，并且已添加公有密钥到您的 DNS 主机提供商。有关更多信息，请参阅 the section called “BYODKIM - 自带 DKIM”。

ii. 在 “身份类型” 下，选择 “提供 DKIM 身份验证令牌 (BYO DKIM)”。

iii. 对于 Private key（私有密钥），粘贴从您的公有-私有密钥对生成的私有密钥。私有密钥必须使用至少 1024 位 RSA 加密（至多 2048 位），并且必须使用 base64 (PEM) 编码进行编码。

> **Note**
>
> 您必须删除生成的私有密钥的第一行和最后一行（分别为 `-----BEGIN PRIVATE KEY-----` 和 `-----END PRIVATE KEY-----`）。此外，还必须删除生成的私有密钥中的换行符。产生的值是一个字符串，不包含空格或换行符。

iv. 对于选择器名称，输入您在域的 DNS 设置中指定的选择器的名称。

9. 确保在 DKIM 签名字段中，选中已启用复选框。
10. （可选）通过包含标签键和该键的可选值，向域身份添加一个或多个标签：
    1. 选择添加新标签，然后输入键。您可以选择为标签添加值。
    2. 重复此操作，但附加的标签不超过 50 个，或选择删除以删除标签。
11. 选择创建身份。

在使用 DKIM 创建和配置您的域身份以后，必须与您的 DNS 提供商完成验证过程 - 继续转到the section called “验证域身份”并执行适用于用来配置身份的 DKIM 类型的 DNS 身份验证过程。

> **Note**

与您的 DNS 提供商一起验证 DKIM 域身份

在创建使用 DKIM 配置的域身份以后，您必须和您的 DNS 提供商按照以下对应的身份验证过程为您选择的 DKIM 类型完成验证过程。

如果您尚未创建域身份，请参阅the section called “创建域身份”。

> **Note**
>
> - 验证域身份需要访问域的 DNS 设置。对这些设置的更改最长可能需要 72 小时才能生效。
> - 如果您使用 Deterministic Easy DKIM (DEED) 创建了全局（副本）身份，则无需进行额外的 DNS 设置，则可以跳过此步骤。欲了解更多信息，请参阅 DE ED。

与您的 DNS 提供商一起验证 DKIM 域身份

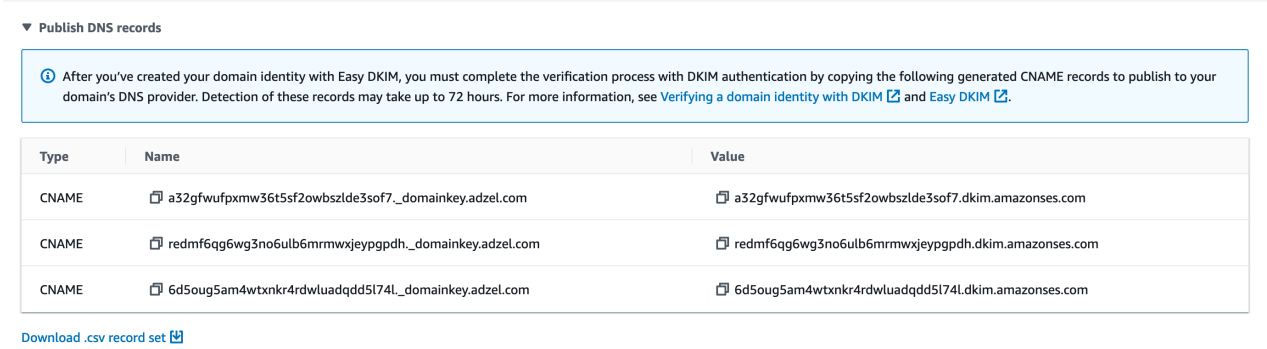
1. 在 Loaded identities（已加载的身份）表中，选择您想要验证的域。
2. 在身份详细信息页面的 Authentication（身份验证）选项卡上，展开 Publish DNS records（发布 DNS 记录）。
3. 根据配置域时使用的 DKIM 风格，Easy DKIM 或 BYODKIM，请按照对应的说明进行操作：

Easy DKIM

验证使用 Easy DKIM 配置的域

1. 在 Publish DNS records（发布 DNS 记录）表中，复制三条会在此部分显示并发布（添加）到您的 DNS 提供商的别名记录。或者，您可以选择下载 .csv 记录集以将记录副本保存到您的电脑中。

   下图显示了要发布到 DNS 提供商的别名记录的示例。



2. 添加别名记录到您的 DNS 主机提供商的各自域的 DNS 设置：

   - 所有 DNS 主机提供商（Route 53 除外）– 登录到您的域的 DNS 或 Web 托管提供商，然后添加别名记录，这些记录包含您之前复制或保存的值。不同的提供商具有不同的 DNS 记录更新过程。请在执行这些过程以后参阅 DNS/托管提供商表。

     > **Note**
     >
     > 少数 DNS 提供商不允许记录名称中包含下划线 (_)。但是，DKIM 记录名称中的下划线是必需的。如果您的 DNS 提供商不允许您在记录名称中输入下划线，请联系提供商的客户支持团队以获取帮助。

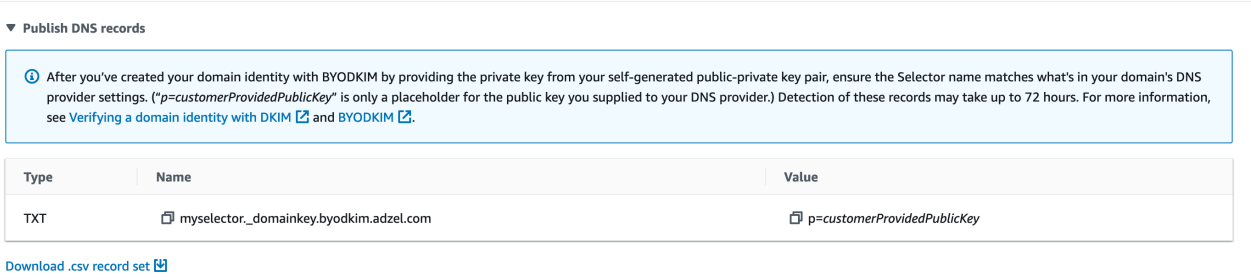
   - Route 53 是您的 DNS 主机提供商 – 如果您在使用 SES 发送电子邮件时使用的相同账户上使用的 Route 53，并且域已注册，SES 将自动更新您的域的 DNS 设置（若您在创建时启用 SES 来发布它们）。或者，您可以在创建后通过点击按钮轻松将它们发布到 Route 53 - 见 the section called “使用控制台编辑身份”。如果您的 DNS 设置未自动更新，或者您要将 CNAME 记录添加到 Route 53，但这些记录与您使用 SES 发送电子邮件时所使用的账户不同，请完成 Editing records（编辑记录）中的步骤。
   - 如果您不确定您的 DNS 提供商是谁 – 请询问您的系统管理员以获取更多信息。

BYODKIM

验证使用 BYODKIM 配置的域

1. 概括而言，当使用 BYODKIM 创建了域或使用 BYODKIM 配置了现有域时，您要在 SES 控制台的 Advance DKIM Settings（高级 DKIM 设置）页面上将私有密钥（来自自行生成的公有/私有密钥对）和选择器名称前缀添加至其对应的字段。现在，您必须通过为您的 DNS 主机提供商更新以下记录，完成验证过程。
2. 在 Publish DNS records（发布 DNS 记录）表中，复制会在 Name（名称）列显示并发布（添加）到您的 DNS 提供商的选择器名称记录。或者，您可以选择 Download .csv record set（下载 .csv 记录集）以将其副本保存到您的电脑中。

   下图显示了要发布到 DNS 提供商的选择器名称记录的示例。



3. 登录到您的域的 DNS 或 Web 托管提供商，然后添加您之前复制或保存的选择器名称记录。不同的提供商具有不同的 DNS 记录更新过程。请在执行这些过程以后参阅 DNS/托管提供商表。

   > **Note**
   >
   > 少数 DNS 提供商不允许记录名称中包含下划线 (_)。但是，DKIM 记录名称中的下划线是必需的。如果您的 DNS 提供商不允许您在记录名称中输入下划线，请联系提供商的客户支持团队以获取帮助。

4. 如果您尚未执行这些过程，请确保将来自您自行生成的公有/私有密钥对的公有密钥添加到您的域的 DNS 或 Web 托管提供商。

   请注意，在 “发布 DNS 记录” 表中，“值” 列中显示的公钥记录仅显示 “p=customerProvidedPublicKey”，作为您保存到计算机或提供给 DNS 提供商的公钥值的占位符。

> **Note**
>
> 当您将公有密钥发布（添加）到 DNS 提供商时，必须按如下所示进行格式化：
>
> - 您必须删除生成的公有密钥的第一行和最后一行（分别为 `-----BEGIN PUBLIC KEY-----` 和 `-----END PUBLIC KEY-----`）。此外，您还必须删除生成的公有密钥中的换行符。产生的值是一个字符串，不包含空格或换行符。
> - 您必须包含 p= 前缀，如 Publish DNS records（发布 DNS 记录）表中的 Value（值）列所示。

4. 对 DNS 设置的更改最长可能需要 72 小时才能生效。一旦 Amazon SES 在域的 DNS 设置中检测到全部必需的 DKIM 记录，则验证过程完成。域的 DKIM 配置显示为成功，身份状态显示为已验证。
5. 如果想配置和验证自定义 MAIL FROM 域，则按照配置自定义 MAIL FROM 域中的过程进行操作。

下表包含的链接指向几个广泛使用的 DNS 提供商的文档。此列表并不详尽无遗，也不表示认可；同样，如果您的 DNS 提供商未列出，这并不意味着您不能将该域与 Amazon SES 一起使用。

DNS/托管提供商	文档链接
GoDaddy	添加 CNAME 记录（外部链接）
DreamHost	如何添加自定义 DNS 记录？（外部链接）
Cloudflare	在 CloudFlare 中管理 DNS 记录（外部链接）
HostGator	使用 HostGator /eNOM 管理 DNS 记录（外部链接）
Namecheap	如何为我的域名添加TXT/SPF/DKIM/DMARC记录？（外部链接）
Names.co.uk	更改您的域的 DNS 设置（外部链接）
Wix	在您的 Wix 账户中添加或更新 CNAME 记录（外部链接）

域验证故障排除

如果您完成了上述步骤，但在 72 小时后域仍未通过验证，请检查以下各项：

- 请确保您在正确的字段中输入了 DNS 记录的值。某些 DNS 提供商将 Name/host (名称/主机) 字段称为 Host (主机) 或 Hostname (主机名)。此外，一些提供商将 Record value (记录值) 字段称为 Points to (指向) 或 Result (结果)。
- 确认您的提供商没有自动将您的域名附加到您在 DNS 记录中输入的名称/主机值后面。有些提供商附加了域名但未指出。如果您的提供商将您的域名附加到名称/主机值后面，则从值后将您的域名删除。您也可以尝试添加句点到 DNS 记录中值的末尾。此句点向提供商指示域名是完全限定的。
- 每个 DNS 记录的 Name/host (名称/主机) 值中都需要下划线字符 (_)。如果您的提供商不允许 DNS 记录名称中的下划线，请联系提供商的客户支持部门以获取更多帮助。
- 您必须添加到域名的 DNS 设置中的验证记录各不相同 AWS 区域。如果您想使用域名发送来自多个地区的电子邮件 AWS 区域，则必须为每个区域创建和验证单独的域身份。

创建电子邮件地址身份

完成以下过程以使用 Amazon SES 控制台创建电子邮件地址身份。

要创建电子邮件地址身份（控制台）

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择已验证身份。
3. 选择创建身份。
4. 在身份详细信息下，选择电子邮件地址作为要创建的身份类型。
5. 对于 Email address (电子邮件地址)，输入您要使用的电子邮件地址。电子邮件地址必须是您能接收邮件并且有权访问的地址。
6. （可选）如果要分配默认配置集，请选中该复选框。
   1. 对于默认配置集，选择要分配给身份的现有配置集。如果尚未创建任何配置集，请参阅 配置集。

> **Note**
>
> 只有当发送时未指定其他配置集时，Amazon SES 才会按默认应用分配的配置集。如果指定了配置集，Amazon SES 将应用指定的配置集来代替默认的配置集。

7. （可选）通过包含标签键和该键的可选值，向域身份添加一个或多个标签：
   1. 选择添加新标签，然后输入键。您可以选择为标签添加值。
   2. 重复此操作，但附加的标签不超过 50 个，或选择删除以删除标签。
8. 要创建您的电子邮件地址身份，请选择创建身份。创建后，您应在五分钟内收到验证电子邮件。下一个步骤是按照下一节中的验证程序验证您的电子邮件地址。

> **Note**
>
> 您可以自定义发送到您尝试验证的电子邮件地址的邮件。有关更多信息，请参阅 the section called “使用自定义验证电子邮件模板”。

现在您已创建电子邮件地址身份，您必须完成验证过程 – 继续the section called “验证电子邮件地址身份”。

验证电子邮件地址身份

在创建电子邮件地址身份后，您必须完成验证过程。

如果您尚未创建电子邮件地址身份，请参阅the section called “创建电子邮件地址身份”。

验证电子邮件地址身份

1. 查看用于创建身份的电子邮件地址的收件箱，并查找来自 no-reply-aws @amazon .com 的电子邮件。
2. 打开电子邮件并单击链接即可完成电子邮件地址的验证过程。完成后，身份状态将更新为已验证。

电子邮件地址验证问题排查

如果您在创建身份后五分钟内没有收到验证电子邮件，请尝试以下问题排查步骤：

- 确保您正确地输入了电子邮件地址。

- 请确保您尝试验证的电子邮件地址能够接收电子邮件。您可以使用其他电子邮件地址发送测试电子邮件到您要验证的地址来测试此地址。
- 检查您的垃圾邮件文件夹。
- 验证电子邮件中的链接将在 24 小时后过期。要发送新的验证电子邮件，请选择身份详细信息页面顶部的重新发送。

创建和验证身份并同时分配默认配置集

您可以使用 Amazon SES API v2 中的CreateEmailIdentity操作来创建新的电子邮件身份，同时设置其默认配置集。

> **Note**
>
> 在完成此部分中的过程之前，必须安装和配置 AWS CLI。有关更多信息，请参阅 用户指南。AWS Command Line Interface

要设置默认配置集，请使用 AWS CLI

- 在命令行中，输入以下命令以使用该CreateEmailIdentity操作。

```
aws sesv2 create-email-identity --email-identity ADDRESS-OR-DOMAIN --configuration-set-name CONFIG-SET
```

在前面的命令中，*ADDRESS-OR-DOMAIN*替换为要验证的电子邮件身份。*CONFIG-SET*替换为要设置为身份默认配置集的配置集的名称。

如果该命令成功执行，它将退出并且不提供任何输出。

验证您的电子邮件地址

1. 检查您正在验证的电子邮件地址的收件箱。您将收到一封带有以下主题的邮件：“Amazon Web Services-所在地区的电子邮件地址验证请求” *RegionName* AWS 区域 ，其中*RegionName*是您尝试验证电子邮件地址的名称。

   打开该邮件，然后单击其中的链接。

> **Note**
>
> 验证邮件中的链接将在发送该邮件 24 小时后过期。在您收到验证电子邮件 24 小时后，重复步骤 1–5 可收到具有有效链接的验证电子邮件。

2. 在 Amazon SES 控制台中的 Identity Management (身份管理) 下，选择 Email Addresses (电子邮件地址)。在电子邮件地址列表中，找到您要验证的电子邮件地址。如果已验证此电子邮件地址，则状态列中的值为“已验证”。

验证您的域

如果您在上述命令行过程中为 `--email-identity` 参数输入了域名，请参阅验证域身份了解更多信息。

使用自定义验证电子邮件模板

当您尝试验证电子邮件地址时，Amazon SES 会向该地址发送一封电子邮件，其内容类似下图所示的示例。

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.

有几个 Amazon SES 客户构建了通过 Amazon SES 代表其自己的客户发送电子邮件的应用程序（例如电子邮件营销套件或票证系统）。这些应用程序的最终用户可能会对电子邮件验证过程感到困惑：验证电子邮件使用的是 Amazon SES 品牌，而不是应用程序的品牌，并且这些最终用户从未直接注册使用 Amazon SES。

如果您的 Amazon SES 使用案例要求您的客户验证其电子邮件地址以便使用 Amazon SES，您可以创建自定义的验证电子邮件。这些自定义电子邮件有助于减少客户困扰，并加快您的客户完成注册过程的速度。

> **Note**
>
> 要使用此特征，您的 Amazon SES 账户必须脱离沙盒。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。

本节中的主题：

- 创建自定义验证电子邮件模板
- 编辑自定义验证电子邮件模板
- 使用自定义模板发送验证电子邮件
- 自定义验证电子邮件常见问题

创建自定义验证电子邮件模板

要创建自定义验证电子邮件，请使用 `CreateCustomVerificationEmailTemplate` API 操作。此操作使用以下输入：

属性	描述
`TemplateName`	模板名称。您指定的名称必须唯一。
`FromEmailAddress`	发出验证电子邮件的电子邮件地址。您指定的地址或域必须经过验证，才能用于您的 Amazon SES 账户。 **Note** `FromEmailAddress` 属性不支持显示名称 (也称为“友好发件人”名称)。
`TemplateSubject`	验证电子邮件的主题行。
`TemplateContent`	电子邮件正文。电子邮件正文可包含 HTML，但存在一定的限制。有关更多信息，请参阅 自定义验证电子邮件常见问题。

属性	描述
SuccessRedirectionURL	在成功验证用户电子邮件地址的情况下将用户转到的 URL。
FailureRedirectionURL	在未成功验证用户电子邮件地址的情况下将用户转到的 URL。

您可以使用 AWS SDKs 或通过 AWS CLI CreateCustomVerificationEmailTemplate操作创建自定义验证电子邮件模板。要了解更多信息 AWS SDKs，请参阅适用于 Amazon Web Services 的工具。有关更多信息 AWS CLI，请参阅AWS 命令行界面。

下一节包含使用 AWS CLI创建自定义验证电子邮件的步骤。这些步骤假定您已安装和配置 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

> **Note**
>
> 您必须使用 1.14.6 或更高版本的 AWS CLI才能完成本节中所述步骤。为了获得最佳效果，请升级到最新版本的 AWS CLI。有关更新的更多信息 AWS CLI，请参阅 AWS Command Line Interface 用户指南AWS Command Line Interface中的安装。

1. 在文本编辑器中，创建一个新文件。将以下内容粘贴到编辑器中：

```
{
  "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**
>
> 为提高可读性，上例中的 TemplateContent 属性包含了换行符。如果将以上示例粘贴到文本文件中，请先删除换行符再继续。

将 TemplateName、FromEmailAddress、TemplateSubject、TemplateContent、SuccessRedir 和 FailureRedirectionURL 的值替换为您自己的值。

> **Note**
>
> 您为 FromEmailAddress 参数指定的电子邮件地址必须经过验证，或者必须是已验证域中的地址。有关更多信息，请参阅 Amazon SES 中已验证的身份。

完成后，将文件另存为 customverificationemail.json。

2. 在命令行键入以下命令，创建自定义验证电子邮件模板：

```
aws sesv2 create-custom-verification-email-template --cli-input-json file://
customverificationemail.json
```

3. （可选）可以键入以下命令确认模板已创建：

```
aws sesv2 list-custom-verification-email-templates
```

编辑自定义验证电子邮件模板

您可以使用 UpdateCustomVerificationEmailTemplate 操作编辑自定义验证电子邮件模板。此操作接受与 CreateCustomVerificationEmailTemplate 操作相同的输入 (即 TemplateName、FromEmailAddress、TemplateSubject、TemplateContent、SuccessRedirecti 和 FailureRedirectionURL 属性)。但是，对于 UpdateCustomVerificationEmailTemplate

操作，这些属性都不是必需的。当为 TemplateName 传递与现有自定义验证电子邮件模板名称相同的值时，您指定的属性将覆盖模板中原有的属性。

使用自定义模板发送验证电子邮件

创建至少一个自定义验证电子邮件模板后，您可以通过调用 SendCustomVerificationEmailAPI 操作将其发送给客户。您可以使用 AWS SDKs 或中的任何一个来调用该SendCustomVerificationEmail操作 AWS CLI。SendCustomVerificationEmail 操作使用以下输入：

属性	描述
EmailAddress	要验证的电子邮件地址。
TemplateName	发送到待验证电子邮件地址的自定义验证电子邮件模板的名称。
ConfigurationSetName	(可选) 发送验证电子邮件时要使用的配置集的名称。

例如，假设您的客户在您的应用程序中使用表单注册您的服务。当客户填写并提交表单之后，应用程序将调用 SendCustomVerificationEmail 操作，传递客户的电子邮件地址和您要使用的模板的名称。

客户将收到一封使用您创建的自定义电子邮件模板的电子邮件。Amazon SES 会为收件人自动添加一个唯一链接，以及简短的免责声明。下图显示了使用在创建自定义验证电子邮件模板中创建的模板的示例验证电子邮件。

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.

自定义验证电子邮件常见问题

本部分包含有关自定义验证电子邮件模板特征的常见问题解答。

问题 1：我可以创建多少个自定义验证电子邮件模板？

您可以为每个 Amazon SES 账户创建最多 50 个自定义验证电子邮件模板。

问题 2：如何向收件人显示自定义验证电子邮件？

自定义验证电子邮件模板包含您在创建模板时指定的内容，后跟收件人必须单击以验证其电子邮件地址的链接。

问题 3：我是否可以预览自定义验证电子邮件？

要预览自定义验证电子邮件，请使用 `SendCustomVerificationEmail` 操作向您自己的地址发送一封验证电子邮件。如果您不单击验证链接，那么 Amazon SES 不会创建新的身份。如果您单击验证链接，您可以选择使用 `DeleteIdentity` 操作删除新创建的身份。

问题 4：我是否可以在自定义验证电子邮件模板中包含图像？

您可以使用 base64 编码在模板 HTML 中嵌入图像。当您以这种方式嵌入图像时，Amazon SES 会自动将图像转换为附件。您可以在命令行中发出以下命令之一对图像进行编码：

Linux, macOS, or Unix

```
base64 -i imagefile.png | tr -d '\n' > output.txt
```

Windows

```
certutil -encodehex -f imagefile.png output.txt 0x40000001
```

将 *imagefile.png* 替换为您要编码的文件的名称。在上面的两个命令中，base64 编码的图像都保存到 `output.txt`。

您可以在模板的 HTML 中加入以下内容以嵌入 base64 编码的图像：`<img src="`*base64EncodedImage*`"/>`

在上一个示例中，将 *png* 替换为编码的图像的文件类型 (例如 jpg 或 gif)，并将 *base64EncodedImage* 替换为 base64 编码的图像 (即前述命令之一中的 `output.txt` 的内容)。

问题 5：对自定义验证电子邮件模板中包含的内容是否有任何限制？

自定义验证电子邮件模板的大小不得超过 10 MB。此外，包含 HTML 的自定义验证电子邮件模板只能使用下表中列出的标签和属性。

HTML 标签	允许的属性
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
colgroup	class, id, span, style, title, width
dd	class, id, style, title
del	class, id, style, title

HTML 标签	允许的属性
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	class, id, style, title
i	class, id, style, title
img	align, alt, class, height, id, src, style, title, width

HTML 标签	允许的属性
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
select	class, id, style, title
small	class, id, style, title
span	class, id, style, title
strike	class, id, style, title

HTML 标签	允许的属性
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
var	class, id, style, title

> **Note**
>
> 自定义验证电子邮件模板不能包含评论标签。

问题 6：我的账户中可以有多少个经过验证的电子邮件地址？

您的 Amazon SES 账户在每个 AWS 地区最多可以有 10,000 个经过验证的身份。在 Amazon SES 中，身份包括经过验证的域和电子邮件地址。

问题 7：我是否可以使用 Amazon SES 控制台来创建自定义验证电子邮件模板？

目前，只能使用 Amazon SES API 来创建、编辑和删除自定义验证电子邮件。

问题 8：我是否可以跟踪在客户收到自定义验证电子邮件时发生的打开和单击事件？

自定义验证电子邮件不提供打开或单击跟踪能力。

问题 9：自定义验证电子邮件是否可以包含自定义标头？

自定义验证电子邮件不能包含自定义标头。

问题 10：我能否删除自定义验证电子邮件底部显示的文本？

以下文本将自动添加到每封自定义验证电子邮件的末尾，且不可删除：

如果您未请求验证此电子邮件地址，请忽略此消息。

问题 11：自定义验证电子邮件是否有 DKIM 签名？

要让验证电子邮件获得 DKIM 签名，必须配置在您创建验证电子邮件模板时在 `FromEmailAddress` 属性中指定的电子邮件地址以生成 DKIM 签名。有关为域和电子邮件地址设置 DKIM 的更多信息，请参阅the section called “使用 DKIM 对电子邮件进行身份验证”。

问题 12：为什么自定义验证电子邮件模板 API 操作不显示在软件开发工具包或 CLI 中？

如果您无法在 SDK 或中使用自定义验证电子邮件模板操作 AWS CLI，则可能使用的是旧版本的 SDK 或 CLI。自定义验证电子邮件模板操作可在以下 SDKs 和 CLIs：

- 版本 1.14.6 或更高版本 AWS Command Line Interface
- 版本 3.3.205.0 或更高版本 适用于 .NET 的 AWS SDK
- 适用于 C++ 的 SDK 的 1.3.20170531.19 或更高版本 AWS
- 版本 1.12.43 或更高版本的 适用于 Go 的 AWS SDK
- 版本 1.11.245 或更高版本的 适用于 Java 的 AWS SDK
- 版本 2.166.0 或更高版本 适用于 JavaScript 的 AWS SDK
- 版本 3.45.2 或更高版本 适用于 PHP 的 AWS SDK
- 版本 1.5.1 或更高版本 AWS SDK for Python (Boto)

- 适用于 Ruby 的 AWS SDK中的 `aws-sdk-ses` Gem 的 1.5.0 版本或更高版本

问题 13：我发送自定义验证电子邮件时，为什么会收到 **`ProductionAccessNotGranted`** 错误？

`ProductionAccessNotGranted` 错误表示您的账户仍在 Amazon SES 沙盒中。只有在您的账户已从沙盒中移出后，您才能发送自定义验证电子邮件。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。

在 Amazon SES 中管理身份

在 Amazon SES 控制台中，您可以查看为每个身份创建的身份 AWS 区域、打开身份以查看和编辑其详细设置、关联默认配置集或删除一个或多个身份。

> **Note**
>
> 本节中引用的步骤仅适用于特定 AWS 区域中的身份。要管理在多个区域创建的身份，请对每个区域重复上述步骤 AWS 区域。

内容

使用 SES 控制台查看身份

您可以使用 Amazon SES 控制台来查看已验证或待验证的域和电子邮件地址身份。您还可以查看验证失败的身份。

查看您的域和电子邮件地址身份

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。

2. 在控制台中，使用区域选择 AWS 区域 器选择要查看其身份列表的。

> **Note**
>
> 此过程仅显示选定 AWS 区域的身份列表。

3. 在导航窗格中的配置下，选择已验证身份。Loaded identities（已加载的身份）表同时显示域和电子邮件地址身份。状态列显示身份是已验证、正在等待验证还是未通过验证过程 - 所有可能状态的定义如下所示：

   - 已验证 – 您的身份已成功验证，可在 SES 中发送。
   - 失败 – SES 无法验证您的身份。如果是域，则意味着 SES 无法在 72 小时内检测到 DNS 记录。如果是电子邮件地址，则表示发送到该电子邮件地址的验证电子邮件未在 24 小时内得到确认。
   - 挂起 — SES 仍在尝试验证身份。
   - 临时故障 — 对于之前验证的域，SES 将定期检查验证所需的 DNS 记录。如果在某个时候 SES 无法检测到记录，状态将更改为临时故障。SES 将在 72 小时内重新检查 DNS 记录，如果无法检测到该记录，则域状态将更改为失败。如果能够检测到该记录，则域状态将更改为已验证。
   - 未开始 — 您尚未开始验证过程。
4. 要按验证状态对身份进行排序，请选择状态列。
5. 要查看身份的详细信息页面，请选择要查看的身份。

使用 SES 控制台删除身份

您可以使用 Amazon SES 控制台从您的选定账户中删除域名或电子邮件地址身份 AWS 区域。

删除域或电子邮件地址身份

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在控制台中，使用区域选择器选择要 AWS 区域 从中删除一个或多个身份。
3. 在导航窗格中的配置下，选择已验证身份。

   Loaded identities（已加载的身份）表显示域和电子邮件地址身份的列表。
4. 在身份列中，选择要删除的身份。选中要删除的每个身份旁边的复选框，可以删除多个身份。
5. 选择 Delete（删除）。

使用 SES 控制台编辑身份

您可以使用 Amazon SES 控制台编辑所选账户中的域名或电子邮件地址身份 AWS 区域。

编辑域或电子邮件地址身份

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在控制台中，使用区域选择器选择要 AWS 区域 从中编辑一个或多个身份。
3. 在导航窗格中的配置下，选择已验证身份。

   Loaded identities（已加载的身份）表显示域和电子邮件地址身份的列表。
4. 在 Identity（身份）列中，选择要编辑的身份（通过直接单击身份名称而不是选中其复选框）。
5. 在身份详情页面上，选择包含要编辑的类别的选项卡。
6. 在所选标签的任何类别容器中，选择要编辑的属性的编辑按钮，进行更改，然后选择保存更改。

   a. 如果您想在“身份验证”选项卡下编辑属性，并且您的域名身份托管在 Amazon Route 53 中，并且您尚未发布其 DNS 记录，则在已DomainKeys 识别邮件 (DKIM) 或自定义邮件来自域容器中的一个或两个容器中都将有一个“将 DNS 记录发布到 Route53”按钮（在“编辑”按钮旁边）。

   > **Note**
   >
   > 身份验证选项卡仅在您的账户中具有使用验证域的验证域的验证域或电子邮件地址时才显示。

   b. 您可以直接从将 DNS 记录发布到 Route53 按钮-只需点击它，就会显示确认横幅，然后将 DNS 记录发布到 Route53 按钮对于相应容器将不再可见。
7. 对要编辑的身份的每个属性重复步骤 5 和 6。

使用 SES API 编辑身份以使用默认配置集

您可以使用该PutEmailIdentityConfigurationSetAttributes操作在现有电子邮件身份中添加或删除默认配置集。

> **Note**
>
> 在完成此部分中的过程之前，必须安装和配置 AWS CLI。有关更多信息，请参阅 用户指南。AWS Command Line Interface

要添加默认配置集，请使用 AWS CLI

- 在命令行中，输入以下命令以使用该PutEmailIdentityConfigurationSetAttributes操作。

```
aws sesv2 put-email-identity-configuration-set-attributes --email-identity ADDRESS-OR-DOMAIN --configuration-set-name CONFIG-SET
```

在前面的命令中，*ADDRESS-OR-DOMAIN*替换为要验证的电子邮件身份。*CONFIG-SET*替换为要设置为身份默认配置集的配置集的名称。

如果该命令成功执行，它将退出并且不提供任何输出。

要移除默认配置集，请使用 AWS CLI

- 在命令行中，输入以下命令以使用该PutEmailIdentityConfigurationSetAttributes操作。

```
aws sesv2 put-email-identity-configuration-set-attributes --email-identity ADDRESS-OR-DOMAIN
```

在前面的命令中，*ADDRESS-OR-DOMAIN*替换为要验证的电子邮件身份。

如果该命令成功执行，它将退出并且不提供任何输出。

使用 SES API 检索身份所用的默认配置集

如果适用，您可以使用该GetEmailIdentity操作返回为电子邮件身份设置的默认配置。

> **Note**
>
> 在完成此部分中的过程之前，必须安装和配置 AWS CLI。有关更多信息，请参阅 用户指南。AWS Command Line Interface

要返回默认配置集，请使用 AWS CLI

- 在命令行输入以下命令以使用该GetEmailIdentity操作。

```
aws sesv2 get-email-identity --email-identity ADDRESS-OR-DOMAIN
```

在前面的命令中，将*ADDRESS-OR-DOMAIN*替换为您想知道其默认配置集（如果有）的电子邮件身份。

如果命令成功执行，它会提供包含电子邮件身份详细信息的 JSON 对象。

使用 SES API 覆盖身份所用的当前默认配置集

您可以使用该SendEmail操作来发送具有不同配置集的电子邮件。如果您执行此操作，那么您指定的配置集会覆盖身份的默认配置集。

> **Note**
>
> 在完成此部分中的过程之前，必须安装和配置 AWS CLI。有关更多信息，请参阅 用户指南。AWS Command Line Interface

要覆盖默认配置集，请使用 AWS CLI

- 在命令行中，输入以下命令以使用该SendEmail操作。

```
aws sesv2 send-email --destination file://DESTINATION-JSON --content file://CONTENT-JSON --from-email-address ADDRESS-OR-DOMAIN --configuration-set-name CONFIG-SET
```

在前面的命令中，*DESTINATION-JSON*替换为目标 JSON 文件、*ADDRESS-OR-DOMAIN*内容 JSON 文件、发件人电子邮件地址以及*CONFIG-SET*您要使用的配置集的名称，而不是为身份设置的默认配置。*CONTENT-JSON*

如果命令成功执行，它会输出一个 MessageId。

在 Amazon SES 中配置身份

Amazon Simple Email Service（Amazon SES）使用简单邮件传输协议（SMTP）发送电子邮件。由于 SMTP 本身不提供任何身份验证，因此垃圾邮件发送者可以发送声称来自其他任何人的电子邮件，同时隐藏其真实来源。垃圾邮件发送者通过伪造电子邮件标头并欺骗源 IP 地址，可误导收件人相信其收到的电子邮件是真实的。

大多数转 ISPs 发电子邮件流量的人都会采取措施来评估电子邮件是否合法。其中一项 ISPs 措施是确定电子邮件是否经过身份验证。身份验证要求发件人证明他们是用来发送电子邮件的账户所有者。在某些情况下，ISPs 拒绝转发未经身份验证的电子邮件。为了确保实现最佳送达率，建议您对电子邮件进行身份验证。

以下各节描述了 ISPs 使用的两种身份验证机制——发件人策略框架 (SPF) 和 DomainKeys 识别邮件 (DKIM)，并提供了如何在 Amazon SES 中使用这些标准的说明。

- 要了解 SPF (它提供了一种回溯到发送电子邮件的系统的方式)，请参阅在 Amazon SES 中使用 SPF 对电子邮件进行身份验证。
- 要了解 DKIM（一种允许您对电子邮件进行签名以 ISPs 表明您的邮件合法且在传输过程中未被修改的标准），请参阅在 Amazon SES 中使用 DKIM 对电子邮件进行身份验证。
- 要了解如何遵循基于域的消息身份验证、报告和合规性 (DMARC) (依赖于 SPF 和 DKIM)，请参阅遵守 Amazon SES 中的 DMARC 身份验证协议。

电子邮件身份验证方法

Amazon Simple Email Service（Amazon SES）使用简单邮件传输协议（SMTP）发送电子邮件。由于 SMTP 本身不提供任何身份验证，因此垃圾邮件发送者可以发送声称来自其他任何人的电子邮件，同时隐藏其真实来源。垃圾邮件发送者通过伪造电子邮件标头并欺骗源 IP 地址，可误导收件人相信其收到的电子邮件是真实的。

大多数转 ISPs 发电子邮件流量的人都会采取措施来评估电子邮件是否合法。其中一项 ISPs 措施是确定电子邮件是否经过身份验证。身份验证要求发件人证明他们是用来发送电子邮件的账号的所有者。在某些情况下，ISPs 拒绝转发未经身份验证的电子邮件。为了确保实现最佳送达率，建议您对电子邮件进行身份验证。

内容

- 使用自定义 MAIL FROM 域
- 遵守 Amazon SES 中的 DMARC 身份验证协议
- 在 Amazon SES 中使用 BIMI

在 Amazon SES 中使用 DKIM 对电子邮件进行身份验证

DomainKeys Identified Mail (DKIM) 是一种电子邮件安全标准，旨在确保声称来自特定域名的电子邮件确实得到了该域名所有者的授权。其使用公有密钥加密技术通过私有密钥对电子邮件进行签名。然后，收件人服务器就可以使用发布到域 DNS 的公有密钥来验证电子邮件的各个部分在传输过程中未被修改。

DKIM 签名为可选项。您可决定使用 DKIM 签名为您的电子邮件签名，以通过符合 DKIM 的电子邮件提供商提高送达率。Amazon SES 提供三种使用 DKIM 签名为您的邮件签名的选项：

- Easy DKIM：SES 生成公有/私有密钥对，并自动将 DKIM 签名添加到您通过该身份发送的每封邮件，请参阅Amazon SES 中的 Easy DKIM。
- D@@ eterministic Easy DKIM (DEED)： AWS 区域 通过创建副本身份，自动继承 DKIM 签名属性的副本身份作为使用 Easy DKIM 的父身份，使您能够在多个之间保持一致的 DKIM 签名，请参阅。在 Amazon SES 中使用确定性 Easy DKIM (DEED)
- BYODKIM（自带 DKIM）：您提供自己的公有/私有密钥对，而 SES 将 DKIM 签名添加到您通过该身份发送的每封邮件，请参阅提供 Amazon SES 中您自己的 DKIM 身份验证令牌 (BYODKIM)。
- 手动添加 DKIM 签名：您将自己的 DKIM 签名添加到使用 `SendRawEmail` API 发送的电子邮件，请参阅Amazon SES 中的手动 DKIM 签名。

DKIM 签名密钥长度

由于许多 DNS 提供商现在完全支持 DKIM 2048 位 RSA 加密，Amazon SES 还支持 DKIM 2048 以允许对电子邮件进行更安全的身份验证，因此当您从 API 或控制台配置 Easy DKIM 时，请将其用作原定设置的密钥长度。2048 位密钥也可以在自带 DKIM (BYODKIM) 中设置和使用，其中您的签名密钥长度必须至少为 1024 位，但不超过 2048 位。

为了安全和电子邮件的送达率，当使用 Easy DKIM 配置时，您可以选择使用 1024 和 2048 位密钥长度，并且在任何 DNS 提供商仍不支持 2048 位导致问题时，可以灵活地返回到 1024 位。创建新身份时，除非指定 1024 位，否则默认情况下将使用 DKIM 2048 创建身份。

为了保持传输中电子邮件的送达率，您可以更改 DKIM 密钥长度的频率将受到限制。这些限制包括：

- 无法切换到与已配置的密钥长度相同的密钥长度。

- 无法在 24 小时内多次切换到不同的密钥长度（除非这是该时间段内第一次降级到 1024 位）。

当您的电子邮件在传输过程中时，DNS 会使用您的公有密钥验证您的电子邮件；因此，如果您更改密钥过快或频繁，DNS 可能无法对您的电子邮件进行 DKIM 身份验证，因为前一个密钥可能已失效，因此，这些限制可防止出现这种情况。

DKIM 注意事项

当您使用 DKIM 对电子邮件进行身份验证时，以下规则适用：

- 您只需为在“发件人”地址中使用的域设置 DKIM。您无需为在“退回路径”或“回复对象”地址中使用的域设置 DKIM。
- Amazon SES 已在多个 AWS 地区推出。如果您使用多个 AWS 区域发送电子邮件，则必须在每个区域中都完成 DKIM 设置过程才能确保您的所有电子邮件都进行 DKIM 签名。
- 由于从父域继承 DKIM 属性，当您通过 DKIM 身份验证验证域时：
  - DKIM 身份验证会应用到该域的全部子域。
    - 如果您不希望子域使用 DKIM 身份验证，子域的 DKIM 设置可以通过禁用集成以及在未来重新启用的功能覆盖父域的设置。
  - DKIM 身份验证还将应用于由在其地址中引用 DKIM 已验证域的电子邮件身份发出的全部电子邮件。
    - 如果您想要发送电子邮件而不使用 DKIM 身份验证，电子邮件地址的 DKIM 设置可以通过禁用集成以及在未来重新启用的功能覆盖子域的设置（若适用）。

了解继承的 DKIM 签名属性

重要的是，首先要理解，如果域配置了 DKIM，无论使用的是 Easy DKIM 还是 BYODKIM，电子邮件地址标识都会从其父域继承其 DKIM 签名属性。因此，在电子邮件地址标识上禁用或启用 DKIM 签名实际上是基于以下关键事实覆盖域的 DKIM 签名属性：

- 如果您已为电子邮件地址所属于的域设置了 DKIM，则无需再为电子邮件地址设置 DKIM。
  - 在为域设置 DKIM 时，Amazon SES 通过从父域继承的 DKIM 属性，自动认证来自该域上每个地址的每一封邮件。
- 特定电子邮件地址标识的 DKIM 设置自动覆盖地址所属的父域或子域（如适用）的设置。

由于电子邮件地址身份的 DKIM 签名属性是从父域继承的，因此如果您计划覆盖这些属性，则必须记住覆盖的层次规则，如下表所述。

父域未启用 DKIM 签名	父域已启用 DKIM 签名
您无法在电子邮件地址身份上启用 DKIM 签名。	您可以禁用对电子邮件地址身份的 DKIM 签名。
	您可以对电子邮件地址身份重新启用 DKIM 签名。

通常不建议禁用您的 DKIM 签名，因为这有可能损害您的发件人声誉，并且会增加您发送的邮件转到垃圾邮件或垃圾邮件文件夹或域名被欺骗的风险。

但是，对于任何特定的使用情形或外围业务决策，可以覆盖电子邮件地址标识上的域继承的 DKIM 签名属性，您可能必须永久或临时禁用 DKIM 签名，或者在以后重新将其启用。请参阅the section called “覆盖电子邮件地址上的 DKIM 签名”。

Amazon SES 中的 Easy DKIM

为域身份设置 Easy DKIM 后，Amazon SES 会自动向您从该身份发送的每封电子邮件添加一个 2048 位的 DKIM 密钥。您可使用 Amazon SES 控制台或者使用 API 来配置 Easy DKIM。

> **Note**
>
> 要设置 Easy DKIM，您必须修改域的 DNS 设置。如果您使用 Route 53 作为 DNS 提供商，Amazon SES 可以自动为您创建适当的记录。如果您使用其他 DNS 提供商，请参阅您的提供商的文档来了解有关为您的域更改 DNS 设置的更多信息。

> **Warning**
>
> 如果您当前启用了 BYODKIM 并且正在迁移到 Easy DKIM，请注意，在设置 Easy DKIM 且您的 DKIM 状态处于待定状态时，Amazon SES 不会使用 BYODKIM 对您的电子邮件进行签名。从您进行调用以启用 Easy DKIM（通过 API 或控制台）的那一刻到 SES 可以确认 DNS 配置的那一刻之间，SES 可能会在没有 DKIM 签名的情况下发送您的电子邮件。因此，建议使用中间步骤从一种 DKIM 签名方法迁移到另一种（例如，使用启用了 BYODKIM 的域的子域，然后在 Easy DKIM 验证通过后将其删除），或者在应用程序停机期间（如果有）执行此活动。

为已验证的域身份设置 Easy DKIM

此部分的过程经过简化，旨在展示在您已创建的域身份上配置 Easy DKIM 所需的步骤。如果您尚未创建域身份，或要查看用于自定义域身份的全部可用选项，如使用原定设置配置集、自定义 MAIL FROM 域和标签等，请参阅the section called “创建域身份”。

创建 Easy DKIM 域身份中有一段是配置其基于 DKM 的验证，您可以选择接受 2048 位的 Amazon SES 原定设置值，或选择 1024 位覆盖默认值。有关 DKIM 签名密钥长度以及如何更改密钥长度的详情，请参阅 the section called “DKIM 签名密钥长度”。

为域设置 Easy DKIM

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择身份。
3. 在身份列表中，选择身份类型为域的身份。

> **Note**
>
> 如果需要创建或验证域，请参阅 创建域身份。

4. 在 “身份验证” 选项卡下的 “DomainKeys已识别邮件 (DKIM)” 容器中，选择 “编辑”。
5. 在高级 DKIM 设置容器中，选择身份类型字段中的 Easy DKIM 按钮。
6. 在 DKIM 签名密钥长度字段中，选择 RSA_2048_BIT 或 RSA_1024_BIT。
7. 在 DKIM 签名字段中，选中已启用复选框。
8. 选择 Save changes（保存更改）。
9. 在使用 Easy DKIM 配置您的域身份以后，必须与您的 DNS 提供商完成验证过程 - 继续转到the section called “验证域身份”并执行适用于 Easy DKIM 的 DNS 身份验证过程。

更改身份的 Easy DKIM 签名密钥长度

本部分中的流程展示了如何轻松更改签名算法所需的 Easy DKIM 位数。虽然始终首选 2048 位签名长度，以提供增强安全性，但在某些情况下，也可能需要使用 1024 位长度，例如必须使用仅支持 DKIM 1024 的 DNS 提供商。

为了保持传输中电子邮件的送达率，您可以更改或改回 DKIM 密钥长度的频率将受到限制。

当您的电子邮件在传输过程中时，DNS 会使用您的公有密钥验证您的电子邮件；因此，如果您更改密钥过快或频繁，DNS 可能无法对您的电子邮件进行 DKIM 身份验证，因为前一个密钥可能已失效，因此，以下限制可防止出现这种情况：

- 无法切换到与已配置的密钥长度相同的密钥长度。
- 无法在 24 小时内多次切换到不同的密钥长度（除非这是该时间段内第一次降级到 1024 位）。

在使用以下过程更改密钥长度时，如果您违反了其中一项限制，控制台将返回一个错误条，指出您提供的输入无效，并说明其无效的原因。

更改 DKIM 签名密钥长度位数的步骤

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的 Configuration（配置）下，选择 Verified identities（已验证身份）。
3. 在身份列表中，选择要更改其 DKIM 签名密钥长度的身份。
4. 在 “身份验证” 选项卡下的 “DomainKeys已识别邮件 (DKIM)” 容器中，选择 “编辑”。
5. 在 Advanced DKIM settings（高级 DKIM 设置）容器中，选择 DKIM signing key length（DKIM 签名密钥长度）字段中的 RSA_2048_BIT 或 RSA_1024_BIT。
6. 选择 Save changes（保存更改）。

在 Amazon SES 中使用确定性 Easy DKIM (DEED)

Deterministic Easy DKIM (DEED) 为管理多个 DKIM 配置提供了一种解决方案。 AWS 区域通过简化 DNS 管理和确保一致的 DKIM 签名，DEED 可帮助您简化多区域电子邮件发送操作，同时保持强大的电子邮件身份验证实践。

什么是确定性 Easy DKIM (DEED)？

Deterministic Easy DKIM (DEED) 是一项功能，它 AWS 区域 基于配置了 Easy DKIM 的父域在所有域中生成一致的 DKIM 令牌。这使您可以复制不同身份 AWS 区域 ，这些身份会自动继承和维护与当前使用 Easy DKIM 配置的父身份相同的 DKIM 签名配置。使用 DEED，您只需为父身份发布一次 DNS 记录，副本身份将使用相同的 DNS 记录来验证域名所有权和管理 DKIM 签名。

通过简化 DNS 管理和确保一致的 DKIM 签名，DEED 可帮助您简化多区域电子邮件发送操作，同时保持最佳的电子邮件身份验证实践。

在谈论 DEED 时使用的术语：

- 父身份 — 使用 Easy DKIM 配置的经过验证的身份，用作副本身份的 DKIM 配置来源。
- 副本身份 — 共享相同 DNS 设置和 DKIM 签名配置的父身份的副本。
- 父区域-设置家长身份 AWS 区域 的地方。
- 副本区域-设置副本身份 AWS 区域 的地方。
- DEED 身份 — 用作父身份或副本身份的任何身份。（创建新身份时，它最初被视为常规（非 Deed）身份。 但是，一旦创建了副本，该身份就会被视为 DEED 身份。）

使用 DEED 的主要好处包括：

- 简化了 DNS 管理 — 只为父身份发布一次 DNS 记录。
- 简化多区域操作-简化将电子邮件发送业务扩展到新区域的流程。
- 减少管理开销-从父身份集中管理 DKIM 配置。

确定性 Easy DKIM (DEED) 的工作原理

当您创建副本身份时，Amazon SES 会自动将 DKIM 签名密钥从父身份复制到副本身份。任何后续的 DKIM 密钥轮换或对父身份所做的密钥长度更改都会自动传播到所有副本身份。

该过程涉及以下工作流程：

1. AWS 区域 使用 Easy DKIM 在中创建家长身份。
2. 为父身份设置所需的 DNS 记录。
3. 在其他中创建副本身份 AWS 区域，指定父身份的域名和 DKIM 签名区域。
4. Amazon SES 会自动将父级的 DKIM 配置复制到副本身份。

重要注意事项：

- 您无法为已经是副本的身份创建副本。
- 父身份必须启用 E asy DKIM，您不能创建 BYODKIM 的副本或手动签名的身份。
- 在删除所有副本身份之前，无法删除父身份。

使用 DEED 设置副本身份

本节将提供示例，说明如何使用 DEED 创建和验证副本身份以及所需的必要权限。

主题

- 创建副本身份
- 验证副本身份配置
- 使用 DEED 所需的权限

创建副本身份

要创建副本身份：

1. 在要创建副本身份的 AWS 区域 位置中，打开 SES 控制台，网址为https://console.aws.amazon.com/ses/。

   （在 SES 控制台中，副本身份被称为全局身份。）
2. 在导航窗格中，选择“身份”。
3. 选择创建身份。
4. 在“身份类型”下选择“域”，然后输入使用 Easy DKIM 配置的现有身份的域名，该身份要复制并用作父级。
5. 展开“高级 DKIM 设置”，然后选择“确定性 Easy DKIM”。
6. 从“父区域”下拉菜单中，选择一个父区域，该身份所在的 Easy DKIM 签名身份与您为全球（副本）身份输入的名称相同。（您的副本区域默认为您登录 SES 控制台时使用的区域。）
7. 确保已启用 DKIM 签名。
8. （可选）向您的域名标识添加一个或多个标签。
9. 查看配置并选择创建身份。

使用 AWS CLI：

要基于使用 Easy DKIM 配置的父身份创建副本身份，您需要指定父级的域名、要创建副本身份的区域以及父级的 DKIM 签名区域，如以下示例所示：

```
aws sesv2 create-email-identity --email-identity example.com --region us-west-2 --dkim-signing-attributes '{"DomainSigningAttributesOrigin": "AWS_SES_US_EAST_1"}'
```

在上述示例中：

1. *example.com*替换为正在复制的父域身份。
2. *us-west-2*替换为要创建副本域身份的区域。

3. *AWS_SES_US_EAST_1*替换为父级的 DKIM 签名区域，该区域代表其 Easy DKIM 签名配置，该配置将被复制到副本身份。

> **Note**
>
> 前AWS_SES_缀表示 DKIM 是使用 Easy DKIM 为父身份配置US_EAST_1的，也是它的 AWS 区域 创建位置。

验证副本身份配置

创建副本身份后，您可以使用父身份的 DKIM 签名配置验证其配置是否正确。

要验证副本身份，请执行以下操作：

1. 在创建副本身份 AWS 区域 的位置，打开 SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中，选择身份，然后从身份表中选择要验证的身份。
3. 在 “身份验证” 选项卡下，“DKIM 配置” 字段将指示状态，“父区域” 字段将指示用于使用 DEED 进行身份的 DKIM 签名配置的区域。

使用 AWS CLI：

使用get-email-identity命令指定副本的域名和区域：

```
aws sesv2 get-email-identity --email-identity example.com --region us-west-2
```

响应将在SigningAttributesOrigin参数中包含父区域的值，表示已使用父身份的 DKIM 签名配置成功配置副本身份：

```
{
  "DkimAttributes": {
    "SigningAttributesOrigin": "AWS_SES_US_EAST_1"
  }
}
```

使用 DEED 所需的权限

要使用 DEED，你需要：

1. 在副本区域创建电子邮件身份的标准权限。
2. 允许从父区域复制 DKIM 签名密钥。

DKIM 复制的 IAM 策略示例

以下策略允许 DKIM 签名密钥从父身份复制到指定的副本区域：

JSON

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

最佳实践

建议采用以下最佳实践：

- 规划您的父区域和副本区域-请考虑您选择的父区域，因为这将是副本区域中使用的 DKIM 配置的真实来源。
- 使用一致的 IAM 策略 — 确保您的 IAM 策略允许在所有目标区域中复制 DKIM。
- 保持父身份处于活动状态 — 请记住，您的副本身份继承父身份的 DKIM 签名配置，由于这种依赖关系，在删除所有副本身份之前，您无法删除父身份。

故障排除

如果您在使用 DEED 时遇到问题，请考虑以下几点：

- 验证错误-确保您拥有 DKIM 复制所需的权限。
- 复制延迟 — 留出一些时间完成复制，尤其是在创建新的副本身份时。
- DNS 问题-验证父身份的 DNS 记录设置和传播是否正确。

提供 Amazon SES 中您自己的 DKIM 身份验证令牌 (BYODKIM)

作为使用 Easy DKIM 的替代方法，您可以改为使用自己的公有-私有密钥对配置 DKIM 身份验证。此过程称为自带 DKIM (BYODKIM)。

使用 BYODKIM，可以使用单个 DNS 记录为您的域配置 DKIM 身份验证，而不是使用 Easy DKIM，后者要求您发布三个单独的 DNS 记录。此外，使用 BYODKIM，您可以根据需要经常为您的域轮换 DKIM 密钥。

本节中的主题：

- 步骤 1：创建密钥对
- 步骤 2：将选择器和公有密钥添加到 DNS 提供商的域配置中
- 步骤 3：配置并验证域以使用 BYODKIM

> **Warning**
>
> 如果您当前启用了 Easy DKIM 并且正在迁移到 BYODKIM，请注意，在设置 BYODKIM 且您的 DKIM 状态处于待定状态时，Amazon SES 将不会使用 Easy DKIM 对您的电子邮件签名。从您进行调用以启用 BYODKIM（通过 API 或控制台）的那一刻到 SES 可以确认 DNS 配置的那一刻之间，SES 可能会在没有 DKIM 签名的情况下发送您的电子邮件。因此，建议使用中间步骤从一种 DKIM 签名方法迁移到另一种（例如，使用启用了 Easy DKIM 的域的子域，然后在 BYODKIM 验证通过后将其删除），或者在应用程序停机期间（如果有）执行此活动。

步骤 1：创建密钥对

要使用自带 DKIM 功能，您需要先创建 RSA 密钥对。

生成的私有密钥必须为 PKCS #1 或 PKCS #8 两种格式之一，使用至少 1024 位 RSA 加密（至多 2048 位），并使用 base64（PEM）编码进行编码。有关 DKIM 签名密钥长度以及如何更改密钥长度的详情，请参阅 the section called “DKIM 签名密钥长度”。

> **Note**
>
> 您可以使用第三方应用程序和工具生成 RSA 密钥对，只要私有密钥是使用至少 1024 位 RSA 加密（至多 2048 位）生成并使用 base64 (PEM) 编码进行编码的。

在以下过程中，使用大多数 Linux、macOS 或 Unix 操作系统中内置的 `openssl genrsa` 命令创建密钥对的示例代码将自动使用 base64 (PEM) 编码。

使用 Linux、macOS 或 Unix 命令行来创建密钥对

1. 在命令行中，输入以下命令生成私钥，*nnnn*替换为至少为 1024 且最多 2048 的位长度：

```
openssl genrsa -f4 -out private.key nnnn
```

2. 在命令行中，输入以下命令可生成公有密钥：

```
openssl rsa -in private.key -outform PEM -pubout -out public.key
```

步骤 2：将选择器和公有密钥添加到 DNS 提供商的域配置中

现在您已创建密钥对，您必须将公有密钥作为 TXT 记录添加到您的域的 DNS 配置中。

将公有密钥添加到您的域的 DNS 配置中

1. 登录您的 DNS 提供商或托管提供商的管理控制台。
2. 将新的文本记录添加到您的域的 DNS 配置中。记录应该使用以下格式：

名称	类型	值
selector。 _domainkey。*example.com*	TXT	p= *yourPublicKey*

在前面的示例中，进行以下更改：

- *selector*替换为标识密钥的唯一名称。

> **Note**
>
> 少数 DNS 提供商不允许记录名称中包含下划线 (_)。但是，DKIM 记录名称中的下划线是必需的。如果您的 DNS 提供商不允许您在记录名称中输入下划线，请联系提供商的客户支持团队以获取帮助。

- *example.com*用您的域名替换。
- *yourPublicKey*替换为您之前创建的公钥，并添加p=前缀，如上面的 “值” 列所示。

> **Note**
>
> 当您将公有密钥发布（添加）到 DNS 提供商时，必须按如下所示进行格式化：
>
> - 您必须删除生成的公有密钥的第一行和最后一行（分别为 `-----BEGIN PUBLIC KEY-----` 和 `-----END PUBLIC KEY-----`）。此外，您还必须删除生成的公有密钥中的换行符。产生的值是一个字符串，不包含空格或换行符。
> - 您必须包含 p= 前缀，如上表中的 Value（值）列所示。

不同的提供商具有不同的 DNS 记录更新过程。下表包含的链接指向几个广泛使用的 DNS 提供商的文档。此列表并不详尽无遗，也不表示认可；同样，如果您的 DNS 提供商未列出，这并不意味着您不能将该域与 Amazon SES 一起使用。

DNS/托管提供商	文档链接
Amazon Route 53	《Amazon Route 53 开发人员指南》中的编辑记录
GoDaddy	添加 TXT 记录（外部链接）
DreamHost	如何添加自定义 DNS 记录？（外部链接）
Cloudflare	在 CloudFlare 中管理 DNS 记录（外部链接）
HostGator	使用 HostGator /eNOM 管理 DNS 记录（外部链接）

DNS/托管提供商	文档链接
Namecheap	如何为我的域名添加TXT/SPF/DKIM/DMARC 记录？（外部链接）
Names.co.uk	更改您的域的 DNS 设置（外部链接）
Wix	在您的 Wix 账户中添加或更新 TXT 记录（外部链接）

步骤 3：配置并验证域以使用 BYODKIM

您可以使用控制台或 AWS CLI为新域（即当前不用于通过 Amazon SES 发送电子邮件的域）以及现有域（即您已设置为与 Amazon SES 一起使用的域）设置 BYODKIM。在使用本节中的 AWS CLI 步骤之前，必须首先安装和配置 AWS CLI。有关更多信息，请参阅《AWS Command Line Interface 用户指南》。

选项 1：创建使用 BYODKIM 的新域身份

此部分包含用于创建使用 BYODKIM 的新域身份的过程。新域身份是您之前尚未设置为使用 Amazon SES 发送电子邮件的域。

如果要配置现有域以使用 BYODKIM，请改为完成 选项 2：配置现有域身份 中的过程。

从控制台使用 BYODKIM 创建身份

- 按照创建域身份中的过程操作，当您到达步骤 8 时，请按照 BYODKIM 特定的说明进行操作。

要使用 BYODKIM 创建身份，请访问 AWS CLI

要配置新域，请使用 Amazon SES API 中的 `CreateEmailIdentity` 操作。

1. 在文本编辑器中，粘贴以下代码：

```
{
    "EmailIdentity":"example.com",
    "DkimSigningAttributes":{
        "DomainSigningPrivateKey":"privateKey",
        "DomainSigningSelector":"selector"
    }
```

```
}
```

在前面的示例中，进行以下更改：

- *example.com*替换为您要创建的域。
- *privateKey*用您的私钥替换。

> **Note**
>
> 您必须删除生成的私有密钥的第一行和最后一行（分别为 -----BEGIN PRIVATE KEY----- 和 -----END PRIVATE KEY-----）。此外，还必须删除生成的私有密钥中的换行符。产生的值是一个字符串，不包含空格或换行符。

- *selector*替换为您在域名的 DNS 配置中创建 TXT 记录时指定的唯一选择器。

完成后，将文件另存为 create-identity.json。

2. 在命令行输入以下命令：

```
aws sesv2 create-email-identity --cli-input-json file://path/to/create-identity.json
```

在前面的命令中，*path/to/create-identity.json*替换为您在上一步中创建的文件的完整路径。

选项 2：配置现有域身份

此部分包含用于更新现有域身份以使用 BYODKIM 的过程。现有域身份是您已设置以使用 Amazon SES 发送电子邮件的域。

从控制台使用 BYODKIM 更新域身份

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的 Configuration（配置）下，选择 Verified identities（已验证身份）。
3. 在身份列表中，选择身份类型为域的身份。

> **Note**
>
> 如果需要创建或验证域，请参阅 创建域身份。

4. 在 “身份验证” 选项卡下的 “DomainKeys 已识别邮件 (DKIM)” 窗格中，选择 “编辑”。
5. 在 Advanced DKIM settings（高级 DKIM 设置）窗格中，选择 Identity type（身份类型）字段中的 Provide DKIM authentication token (BYODKIM) [提供 DKIM 身份验证令牌 (BYODKIM)] 按钮。
6. 对于 Private key（私有密钥），粘贴先前生成的私有密钥。

> **Note**
>
> 您必须删除生成的私有密钥的第一行和最后一行（分别为 `-----BEGIN PRIVATE KEY-----` 和 `-----END PRIVATE KEY-----`）。此外，还必须删除生成的私有密钥中的换行符。产生的值是一个字符串，不包含空格或换行符。

7. 对于选择器名称，输入您在域的 DNS 设置中指定的选择器的名称。
8. 在 DKIM 签名字段中，选中已启用复选框。
9. 选择 Save changes（保存更改）。

要使用 BYODKIM 更新域身份，请访问 AWS CLI

要配置现有域，请使用 Amazon SES API 中的 `PutEmailIdentityDkimSigningAttributes` 操作。

1. 在文本编辑器中，粘贴以下代码：

```
{
    "SigningAttributes":{
        "DomainSigningPrivateKey":"privateKey",
        "DomainSigningSelector":"selector"
    },
    "SigningAttributesOrigin":"EXTERNAL"
}
```

在前面的示例中，进行以下更改：

- *privateKey*用您的私钥替换。

> **Note**
>
> 您必须删除生成的私有密钥的第一行和最后一行（分别为 `-----BEGIN PRIVATE KEY-----` 和 `-----END PRIVATE KEY-----`）。此外，还必须删除生成的私有密钥中的换行符。产生的值是一个字符串，不包含空格或换行符。

- *selector*替换为您在域名的 DNS 配置中创建 TXT 记录时指定的唯一选择器。

完成后，将文件另存为 `update-identity.json`。

2. 在命令行输入以下命令：

```
aws sesv2 put-email-identity-dkim-signing-attributes --email-identity example.com
 --cli-input-json file://path/to/update-identity.json
```

在前面的命令中，进行以下更改：

- *path/to/update-identity.json*替换为您在上一步中创建的文件的完整路径。
- *example.com*替换为您要更新的域名。

验证使用 BYODKIM 的域的 DKIM 状态

从控制台验证域的 DKIM 状态

配置域以使用 BYODKIM 后，您可以使用 SES 控制台验证已正确地配置 DKIM。

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择已验证身份。
3. 在身份列表中，选择要验证其 DKIM 状态的身份。
4. 对 DNS 设置的更改最长可能需要 72 小时才能生效。一旦 Amazon SES 在域的 DNS 设置中检测到全部必需的 DKIM 记录，则验证过程完成。如果所有配置都正确，则您的域名的 DKIM 配置字段将在已DomainKeys识别邮件 (DKIM) 窗格中显示成功，而身份状态字段在 “摘要” 窗格中显示已验证。

要验证域名的 DKIM 状态，请使用 AWS CLI

将域配置为使用 BYODKIM 后，您可以使用该 GetEmailIdentity操作来验证 DKIM 的配置是否正确。

- 在命令行输入以下命令：

```
aws sesv2 get-email-identity --email-identity example.com
```

在前面的命令中，*example.com*用您的域名替换。

此命令返回一个 JSON 对象，该对象包含类似于以下示例的部分。

```
{
    ...
    "DkimAttributes": {
        "SigningAttributesOrigin": "EXTERNAL",
        "SigningEnabled": true,
        "Status": "SUCCESS",
        "Tokens": [ ]
    },
    ...
}
```

如果以下所有条件均成立，则表示已为域正确配置 BYODKIM：

- `SigningAttributesOrigin` 属性的值为 `EXTERNAL`。
- `SigningEnabled` 的值为 `true`。
- `Status` 的值为 `SUCCESS`。

管理 Easy DKIM 和 BYODKIM

使用基于 Web 的 Amazon SES 控制台或 Amazon SES API，您可以为已通过 Easy DKIM 或 BYODKIM 验证的身份管理 DKIM 设置。您可以使用其中任一方法来获取身份的 DKIM 记录，或者为身份启用或禁用 Easy DKIM 签名。

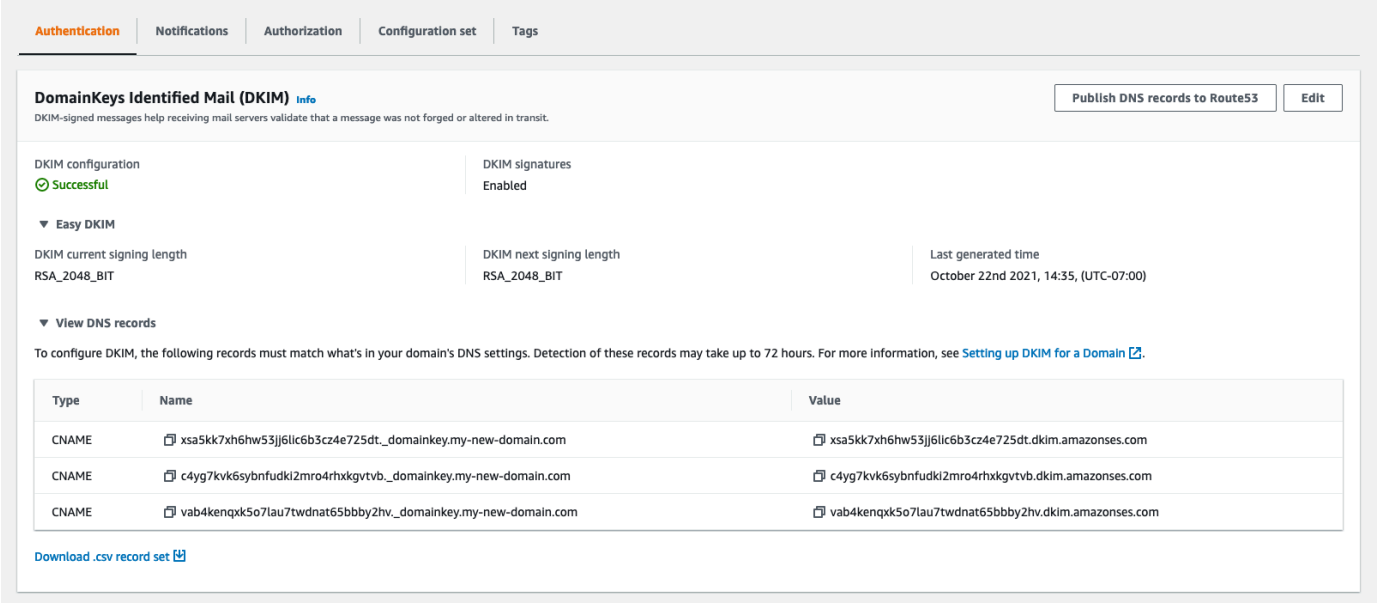
获取身份的 DKIM 记录

您可以随时使用 Amazon SES 控制台获取您的域或电子邮件地址的 DKIM 记录。

使用控制台获取身份的 DKIM 记录

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择已验证身份。
3. 在身份列表中，选择要获取其 DKIM 记录的身份。
4. 在身份详细信息页面的身份验证选项卡上，展开查看 DNS 记录。
5. 复制在此部分显示的三条别名记录（如果您使用 Easy DKIM）或 TXT 记录（如果您使用 BYODKIM）。或者，您可以选择下载 .csv 记录集以将记录副本保存到您的电脑中。

   下图显示了展开后的 View DNS records（查看 DNS 记录）部分显示的与 Easy DKIM 关联的别名记录的示例。



您还可以使用 Amazon SES API 来获取身份的 DKIM 记录。与 API 交互的常用方法是使用 AWS CLI。

要获取身份的 DKIM 记录，请使用 AWS CLI

1. 在命令行处，键入以下命令：

```
aws ses get-identity-dkim-attributes --identities "example.com"
```

   在前面的示例中，*example.com*替换为您想要获取 DKIM 记录的身份。您可以指定电子邮件地址或域。

2. 此命令的输出包含一个 DkimTokens 部分，如以下示例所示：

```
{
    "DkimAttributes": {
        "example.com": {
            "DkimEnabled": true,
            "DkimVerificationStatus": "Success",
            "DkimTokens": [
                "hirjd4exampled5477y22yd23ettobi",
                "v3rnz522czcl46quexamplek3efo5o6x",
                "y4examplexbhyhnsjcmtvzotfvqjmdqoj"
            ]
        }
    }
}
```

您可以使用令牌创建添加到域的 DNS 设置中的 CNAME 记录。要创建 CNAME 记录，请使用以下模板：

```
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
```

将的*token1*每个实例替换为您运行get-identity-dkim-attributes命令时收到的列表中的第一个标记，将的所有实例替换为列表中的第二个标记，并将的所有实例替换为列表中的第三个标记。*token2 token3*

例如，对上述示例中所示的令牌应用此模板会生成以下记录：

```
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**
>
> 并非所有人都 AWS 区域 使用默认的 SES DKIM 域名，`dkim.amazonses.com`要查看您的地区是否使用特定区域的 DKIM 域，请查看中的 DKIM 域名表。AWS 一般参考

为身份禁用 Easy DKIM

您可以使用 Amazon SES 控制台快速为身份禁用 DKIM 身份验证。

为身份禁用 DKIM

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择已验证身份。
3. 在身份列表中，选择要为其禁用 DKIM 的身份。
4. 在 “身份验证” 选项卡下的 “DomainKeys已识别邮件 (DKIM)” 容器中，选择 “编辑”。
5. 在 Advanced DKIM settings（高级 DKIM 设置）中，选中 DKIM signatures（DKIM 签名）字段中的 Enabled（已启用）复选框。

您还可以使用 Amazon SES API 为身份禁用 DKIM。与 API 交互的常用方法是使用 AWS CLI。

要禁用身份的 DKIM，请使用 AWS CLI

- 在命令行处，键入以下命令：

```
aws ses set-identity-dkim-enabled --identity example.com --no-dkim-enabled
```

在前面的示例中，*example.com*替换为要禁用 DKIM 的身份。您可以指定电子邮件地址或域。

为身份启用 Easy DKIM

如果您之前为身份禁用了 DKIM，则可以使用 Amazon SES 控制台再次启用它。

为身份启用 DKIM

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。

2. 在导航窗格中的配置下，选择已验证身份。
3. 在身份列表中，选择要为其启用 DKIM 的身份。
4. 在 “身份验证” 选项卡下的 “DomainKeys已识别邮件 (DKIM)” 容器中，选择 “编辑”。
5. 在高级 DKIM 设置中，选中 DKIM 签名字段中的已启用复选框。

您还可以使用 Amazon SES API 为身份启用 DKIM。与 API 交互的常用方法是使用 AWS CLI。

要为身份启用 DKIM，请使用 AWS CLI

- 在命令行处，键入以下命令：

```
aws ses set-identity-dkim-enabled --identity example.com --dkim-enabled
```

在前面的示例中，*example.com*替换为要为其启用 DKIM 的身份。您可以指定电子邮件地址或域。

覆盖在电子邮件地址身份上继承的 DKIM 签名

在此部分中，您将学习如何覆盖（禁用或启用）来自您已使用 Amazon SES 验证的特定电子邮件地址身份上的父域继承的 DKIM 签名属性。您只能对属于您已拥有的域的电子邮件地址身份执行此操作，因为 DNS 设置是在域级别配置的。

> **Important**
>
> 您无法为电子邮件地址身份禁用/启用 DKIM 签名……
>
> - 在您不拥有的域上。例如，您无法为注册 gmail.com 或 hotmail.com 地址切换 DKIM。
> - 在您拥有但尚未在 Amazon SES 中验证的域名上，
> - 在您拥有但尚未在域上启用 DKIM 签名的域名上。

本节包含以下主题：

- 了解继承的 DKIM 签名属性
- 覆盖对电子邮件地址身份的 DKIM 签名（控制台）
- 覆盖对电子邮件地址身份的 DKIM 签名 (AWS CLI)

了解继承的 DKIM 签名属性

重要的是，首先要理解，如果域配置了 DKIM，无论使用的是 Easy DKIM 还是 BYODKIM，电子邮件地址标识都会从其父域继承其 DKIM 签名属性。因此，在电子邮件地址标识上禁用或启用 DKIM 签名实际上是基于以下关键事实覆盖域的 DKIM 签名属性：

- 如果您已为电子邮件地址所属于的域设置了 DKIM，则无需再为电子邮件地址设置 DKIM。
  - 在为域设置 DKIM 时，Amazon SES 通过从父域继承的 DKIM 属性，自动认证来自该域上每个地址的每一封邮件。
- 特定电子邮件地址标识的 DKIM 设置自动覆盖地址所属的父域或子域（如适用）的设置。

由于电子邮件地址身份的 DKIM 签名属性是从父域继承的，因此如果您计划覆盖这些属性，则必须记住覆盖的层次规则，如下表所述。

父域未启用 DKIM 签名	父域已启用 DKIM 签名
您无法在电子邮件地址身份上启用 DKIM 签名。	您可以禁用对电子邮件地址身份的 DKIM 签名。
	您可以对电子邮件地址身份重新启用 DKIM 签名。

通常不建议禁用您的 DKIM 签名，因为这有可能损害您的发件人声誉，并且会增加您发送的邮件转到垃圾邮件或垃圾邮件文件夹或域名被欺骗的风险。

但是，对于任何特定的使用情形或外围业务决策，可以覆盖电子邮件地址标识上的域继承的 DKIM 签名属性，您可能必须永久或临时禁用 DKIM 签名，或者在以后重新将其启用。

覆盖对电子邮件地址身份的 DKIM 签名（控制台）

以下 SES 控制台过程向您介绍如何覆盖（禁用或启用）来自您已使用 Amazon SES 验证的特定电子邮件地址身份上的父域继承的 DKIM 签名属性。

使用控制台禁用/启用电子邮件地址身份的 DKIM 签名

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择已验证身份。

3. 在身份列表中，选择 Identity type（身份类型）为 Email address（电子邮件地址）并且属于已验证域之一的身份。
4. 在 “身份验证” 选项卡下的 “DomainKeys 已识别邮件 (DKIM)” 容器中，选择 “编辑”。

> **Note**
>
> 仅当选定的电子邮件地址身份属于已经过 SES 验证的域时，才会显示 Authentication（身份验证）选项卡。如果您尚未验证域，请参阅 创建域身份。

5. 在 DKIM signatures（DKIM）签名字段中的 Advanced DKIM settings（高级 DKIM 设置）下，清除 Enabled（已启用）复选框以禁用 DKIM 签名，或者选中它以重新启用 DKIM 签名（如果先前已覆盖）。
6. 选择 Save changes（保存更改）。

覆盖对电子邮件地址身份的 DKIM 签名 (AWS CLI)

以下示例使用 AWS CLI 带有 SES API 命令和参数，这些命令和参数将覆盖（禁用或启用）从父域继承的 DKIM 签名属性，该属性已通过您已通过 SES 验证的特定电子邮件地址身份。

要禁用/启用电子邮件地址身份的 DKIM 签名 AWS CLI

- 假设你拥有 example.com 域名，并且要禁用域的其中一个电子邮件地址的 DKIM 签名，请在命令行键入以下命令：

```
aws sesv2 put-email-identity-dkim-attributes --email-identity marketing@example.com
 --no-signing-enabled
```

  a. *marketing@example.com*替换为要禁用 DKIM 签名的电子邮件地址身份。

  b. `--no-signing-enabled` 将禁用 DKIM 签名。要重新启用 DKIM 签名，请使用 `--signing-enabled`。

Amazon SES 中的手动 DKIM 签名

作为使用 Easy DKIM 的替代方案，您可以手动将 DKIM 签名添加到您的邮件，然后使用 Amazon SES 发送这些邮件。如果您选择手动对邮件进行签名，则必须首先创建 DKIM 签名。创建消息和 DKIM 签名后，您可以使用 SendRawEmailAPI 发送消息。

如果您决定手动对电子邮件进行签名，请考虑以下因素：

- 您使用 Amazon SES 发送的每封邮件均包含引用 amazonses.com 的签名域的 DKIM 标头（即，它包含以下字符串：`d=amazonses.com`）。因此，如果您手动对邮件进行签名，邮件将包括两个 DKIM 标头：一个用于您的域，一个由 Amazon SES 自动创建用于 amazonses.com。
- Amazon SES 不验证您手动添加到邮件中的 DKIM 签名。如果邮件中的 DKIM 签名出现错误，则它可能会被电子邮件提供商拒绝。
- 在对邮件进行签名时，应使用至少为 1024 位的位长度。
- 不要对以下字段进行签名：Message-ID (邮件 ID)、Date (日期)、Return-Path (退回路径)、Bounces-To (退回到)。

> **Note**
>
> 如果您使用电子邮件客户端通过 Amazon SES SMTP 接口发送电子邮件，则您的客户端可能会自动对邮件进行 DKIM 签名。某些客户端可能会对其中一些字段进行签名。有关预设情况下对哪些字段进行签名的信息，请参阅您的电子邮件客户端文档。

在 Amazon SES 中使用 SPF 对电子邮件进行身份验证

发件人策略框架 (SPF) 是一种电子邮件验证标准，旨在防止电子邮件欺骗。域所有者使用 SPF 来告知电子邮件提供商，允许哪些服务器从其域发送电子邮件。SPF 是在 RFC 7208 中定义的。

您通过 Amazon SES 发送的邮件将自动使用 `amazonses.com` 的子域作为原定设置的 MAIL FROM 域。SPF 身份验证成功验证这些邮件，因为原定设置的 MAIL FROM 域与发送电子邮件的应用程序（在这种情况下为 SES）相匹配。因此，在 SES 中，SPF 是为您隐式设置的。

但是，如果您不想使用 SES 默认的 MAIL FROM 域，而是希望使用您拥有的域的子域，则在 SES 中称为使用自定义 MAIL FROM 域。为此，它要求您为您的自定义 MAIL FROM 域发布您自己的 SPF 记录。此外，SES 还要求您设置 MX 记录，以便您的自定义 MAIL FROM 域可以接收电子邮件提供商向您发送的退回邮件和投诉通知。

了解如何设置 SPF 身份验证

提供了使用 SPF 配置域以及如何在 the section called “使用自定义 MAIL FROM 域” 中发布 MX 和 SPF（类型为 TXT）记录的说明。

使用自定义 MAIL FROM 域

当您发送电子邮件时，它具有两个指示其来源的地址：一个显示给邮件收件人的 From 地址，以及一个指示邮件来源的 MAIL FROM 地址。MAIL FROM 地址有时称作 envelope sender、envelope

from、bounce address 或 Return Path 地址。邮件服务器使用 MAIL FROM 地址返回退回邮件和其他错误通知。通常，收件人只有在查看邮件的源代码时才能查看 MAIL FROM 地址。

Amazon SES 将您发送的邮件的 MAIL FROM 域设置为原定设置值，除非您指定自己的（自定义）域。本部分讨论设置自定义 MAIL FROM 域的好处，并包括设置过程。

为什么使用自定义 MAIL FROM 域？

您通过 Amazon SES 发送的邮件将自动使用 `amazonses.com` 的子域作为原定设置的 MAIL FROM 域。发件人策略框架（SPF）身份验证成功验证这些邮件，因为原定设置的 MAIL FROM 域与发送电子邮件的应用程序（在这种情况下为 SES）相匹配。

如果您不想使用 SES 原定设置 MAIL FROM 域，而是希望使用您拥有的域的子域，则在 SES 中称为使用自定义 MAIL FROM 域。为此，它要求您为您的自定义 MAIL FROM 域发布您自己的 SPF 记录。此外，SES 还要求您设置 MX 记录，以便您的域可以接收电子邮件提供商向您发送的退回邮件和投诉通知。

通过使用自定义 MAIL FROM 域，您可以灵活地使用 SPF 和/或 DKIM 来实现基于域的消息身份验证、报告和一致性（DMARC）验证。DMARC 使发件人的域能够指示从该域发送的电子邮件受一个或多个身份验证系统的保护。可通过以下两种方法实现 DMARC 验证：the section called “通过 SPF 遵守 DMARC”和the section called “通过 DKIM 遵守 DMARC”。

选择自定义 MAIL FROM 域

在下文中，“MAIL FROM dom ain” 一词始终指您拥有的域的子域名——您用于自定义 MAIL FROM 域的子域不得用于其他任何用途，并且必须满足以下要求：

- MAIL FROM 域必须是经验证身份（电子邮件地址或域）的父域的子域。
- MAIL FROM 域不应是您也用来从中发送电子邮件的子域。
- MAIL FROM 域不应是您用于接收电子邮件的子域。

将 SPF 与自定义 MAIL FROM 域结合使用

发件人策略框架 (SPF) 是一种电子邮件验证标准，旨在防止电子邮件欺骗。您可以使用 SPF 配置您的自定义 MAIL FROM 域，以告诉电子邮件提供商允许哪些服务器从您的自定义 MAIL FROM 域发送电子邮件。SPF 是在 RFC 7208 中定义的。

要设置 SPF，您要将 TXT 记录发布到您的自定义 MAIL FROM 域的 DNS 配置。此记录包含您授权使用您的自定义 MAIL FROM 域发送电子邮件的服务器的列表。当电子邮件提供商从您的自定义 MAIL FROM 域接收到邮件时，它将检查该域的 DNS 记录，以确保电子邮件是从授权服务器发送的。

如果您想使用此 SPF 记录来遵守 DMARC，则发件人地址中的域必须与 MAIL FROM 域匹配。请参阅 the section called “通过 SPF 遵守 DMARC”。

下一节the section called “配置自定义 MAIL FROM 域”将介绍如何为您的自定义 MAIL FROM 域设置 SPF。

配置自定义 MAIL FROM 域

设置自定义 MAIL FROM 域的过程要求您将记录添加到该域的 DNS 配置中。SES 要求您发布 MX 记录，以便您的域可以接收电子邮件提供商向您发送的退回邮件和投诉通知。您还必须发布 SPF 记录（类型为 TXT），以证明 Amazon SES 经授权可从您的域发送电子邮件。

您可以为整个域或子域，也可以为单个电子邮件地址设置自定义 MAIL FROM 域。以下过程演示如何使用 Amazon SES 控制台来配置自定义 MAIL FROM 域。您也可以使用 SetIdentityMailFromDomainAPI 操作配置自定义 MAIL FROM 域。

为已验证的域设置自定义 MAIL FROM 域

这些过程向您演示了如何为整个域或子域配置自定义 MAIL FROM 域，以便从该域地址发送的所有消息都将使用此自定义 MAIL FROM 域。

将经验证的域配置为使用指定的自定义 MAIL FROM 域

1. 打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航面板中的配置下，选择身份。
3. 在身份列表中，选择要配置的身份，其中 Identity type（身份类型）是 Domain（域），以及 Status（状态）为 Verified（已验证）。
   - 如果 Status（状态）是 Unverified（未核实），请在 与您的 DNS 提供商一起验证 DKIM 域身份 完成过程以验证电子邮件地址的域。
4. 在屏幕底部的 “来自域的自定义邮件” 窗格中，选择 “编辑”。
5. 在 General details（一般细节）窗格中，执行以下操作：
   a. 选择 Use a custom MAIL FROM domain（使用自定义 MAIL FROM 域）复选框。
   b. 对于 MAIL FROM Domain (MAIL FROM 域)，输入要用作 MAIL FROM 子域的域。
   c. 对于 Behavior on MX failure（MX 失败时的行为），选择下列选项之一：
      - 使用原定设置的 MAIL FROM 域名 – 如果未正确设置自定义 MAIL FROM 域的 MX 记录，那么 Amazon SES 将使用 `amazonses.com` 的子域。子域名因您使用的 AWS 区域 Amazon SES 而异。

- 拒绝邮件 – 如果未正确设置自定义 MAIL FROM 域的 MX 记录，那么 Amazon SES 将返回 `MailFromDomainNotVerified` 错误。您尝试从此域发送的电子邮件将被自动拒绝。

d. 选择 Save changes（保存更改）- 您将会回到上一个屏幕。

6. 将 MX 和 SPF（类型为 TXT）记录发布到自定义 MAIL FROM 域的 DNS 服务器：

在 Custom MAIL FROM domain（自定义 MAIL FROM 域）窗格中，Publish DNS records（发布 DNS 记录）表现在会显示您必须发布（添加）到域的 DNS 配置中的 MX 和 SPF（类型为 TXT）记录。这些记录使用下表中显示的格式。

名称	类型	值
subdomain .*domain*.*com*	MX	10 feedback-smtp.*region*.amazonses.com
subdomain .*domain*.*com*	TXT	“v=spf1 include:amazonses.com ~全部”

在以上记录中，

- *subdomain*。 *domain*。 *com*将填充你的 MAIL FROM 子域名
- *region*将填充您要验证 MAIL FROM 域的名称（例如`us-west-2`、`us-east-1`、`eu-west-1`或，等）AWS 区域
- 与 MX 值一起列出的数字 10 是邮件服务器的优先顺序，需要按照 DNS 提供商的 GUI 指定的单独值字段输入
- SPF 的 TXT 记录值通常必须包含引号，但某些 DNS 提供商不需要引号。

从 Publish DNS records（发布 DNS 记录）表中，通过选择每个值旁边的复制图标复制 MX 和 SPF（类型为 TXT）记录，然后将它们粘贴到 DNS 提供商 GUI 中的相应字段中。或者，您可以选择下载 .csv 记录集以将记录副本保存到您的电脑中。

> **Important**
>
> - 发布 MX 和 SPF（类型 TXT）记录的具体程序取决于您的 DNS 或托管服务提供商。有关将这些记录添加到域名的 DNS 配置的信息，请参阅您的提供商的文档或与他们联系。
> - 要使用 Amazon SES 成功设置自定义 MAIL FROM 域，您必须仅将一条 MX 记录发布到您的 MAIL FROM 域的 DNS 服务器。如果 MAIL FROM 域有多条 MX 记录，则使用 Amazon SES 设置自定义 MAIL FROM 将失败。

如果 Route 53 为您的 MAIL FROM 域提供 DNS 服务，并且您使用与 Route 53 相同的账户登录，则选择使用 R oute 53 发布记录。 AWS Management Console DNS 记录将自动应用于您的域的 DNS 配置。

如果使用其他 DNS 提供商，则必须手动将 DNS 记录发布到 MAIL FROM 域的 DNS 服务器。将 DNS 记录添加到域的 DNS 服务器的过程因您的网络托管服务或 DNS 提供商而异。

为您的域发布 DNS 记录的过程取决于您使用的 DNS 提供商。下表包含的链接指向几个广泛使用的 DNS 提供商的文档。此列表并不详尽无遗，并不表示认可；同样，如果您的 DNS 提供商没有列出，这并不意味着他们不支持 MAIL FROM 域配置。

DNS/托管提供商名称	文档链接
GoDaddy	• MX：添加 MX 记录（外部链接） • TXT：添加 TXT 记录（外部链接）
DreamHost	• MX：如何更改我的 MX 记录？（外部链接） • TXT：如何添加自定义 DNS 记录？（外部链接）
Cloudflare	• MX：如何添加或编辑邮件或 MX 记录？（外部链接） • TXT：在 Cloudflare 中管理 DNS 记录（外部链接）

DNS/托管提供商名称	文档链接
HostGator	• MX：设置 MX 记录（外部链接） • TXT：使用 HostGator /eNOM 管理 DNS 记录（外部链接）
Namecheap	• MX：如何设置邮件服务所需的 MX 记录？（外部链接） • TXT：如何为我的域名添加TXT/SPF/DKIM/DMARC记录？（外部链接）
Names.co.uk	• MX：更改您的域的 DNS 设置（外部链接） • TXT：更改您的域的 DNS 设置（外部链接）
Wix	• MX：在您的 Wix 账户中添加或更新 MX 记录（外部链接） • TXT：在您的 Wix 账户中添加或更新 TXT 记录（外部链接）

当 Amazon SES 检测到记录已生效时，您会收到一封电子邮件，通知您自定义 MAIL FROM 域已成功设置。视您的 DNS 提供商而定，Amazon SES 检测 MX 记录之前可能会延迟最多 72 个小时。

为已验证的电子邮件地址设置自定义 MAIL FROM 域

您还可以为特定电子邮件地址设置自定义 MAIL FROM 域。要为电子邮件地址设置自定义 MAIL FROM 域，您必须修改与该电子邮件地址相关联的域的 DNS 记录。

> **Note**
>
> 您无法为不归您所有的域上的地址设置自定义 MAIL FROM 域（例如，您无法为 `gmail.com` 域上的地址创建自定义 MAIL FROM 域），因为您无法将必要的 DNS 记录添加到该域中）。

将经验证的电子邮件地址配置为使用指定的 MAIL FROM 域

1. 打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航面板中的配置下，选择身份。
3. 在身份列表中，选择要配置的身份，其中 Identity type（身份类型）是 Email address（电子邮件地址），以及 Status（状态）为 Verified（已验证）。
   - 如果 Status（状态）是 Unverified（未核实），请在 验证电子邮件地址身份 完成过程以验证电子邮件地址的域。
4. 在 MAIL FROM Domain（MAIL FROM 域）选项卡下方，选择 Custom MAIL FROM domain（自定义 MAIL FROM 域）窗格中的 Edit（编辑）。
5. 在 General details（一般细节）窗格中，执行以下操作：
   a. 选择 Use a custom MAIL FROM domain（使用自定义 MAIL FROM 域）复选框。
   b. 对于 MAIL FROM Domain (MAIL FROM 域)，输入要用作 MAIL FROM 子域的域。
   c. 对于 Behavior on MX failure（MX 失败时的行为），选择下列选项之一：
      - 使用原定设置的 MAIL FROM 域名 – 如果未正确设置自定义 MAIL FROM 域的 MX 记录，那么 Amazon SES 将使用 `amazonses.com` 的子域。子域名因您使用的 AWS 区域 Amazon SES 而异。
      - 拒绝邮件 – 如果未正确设置自定义 MAIL FROM 域的 MX 记录，那么 Amazon SES 将返回 `MailFromDomainNotVerified` 错误。您尝试从此电子邮件地址发送的电子邮件将被自动拒绝。
   d. 选择 Save changes（保存更改）- 您将会回到上一个屏幕。
6. 将 MX 和 SPF（类型为 TXT）记录发布到自定义 MAIL FROM 域的 DNS 服务器：

   在 Custom MAIL FROM domain（自定义 MAIL FROM 域）窗格中，Publish DNS records（发布 DNS 记录）表现在会显示您必须发布（添加）到域的 DNS 配置中的 MX 和 SPF（类型为 TXT）记录。这些记录使用下表中显示的格式。

名称	类型	值
subdomain .*domain*.*com*	MX	10 feedback-smtp.*region*.amazonses.com

名称	类型	值
subdomain .*domain*.*com*	TXT	“v=spf1 include:amazonses.com ~全部”

在以上记录中，

- *subdomain*。 *domain*。 *com*将填充你的 MAIL FROM 子域名
- *region*将填充您要验证 MAIL FROM 域的名称（例如us-west-2us-east-1eu-west-1、或，等） AWS 区域
- 与 MX 值一起列出的数字 10 是邮件服务器的优先顺序，需要按照 DNS 提供商的 GUI 指定的单独值字段输入
- SPF 的 TXT 记录值必须包含引号

从 Publish DNS records（发布 DNS 记录）表中，通过选择每个值旁边的复制图标复制 MX 和 SPF（类型为 TXT）记录，然后将它们粘贴到 DNS 提供商 GUI 中的相应字段中。或者，您可以选择下载 .csv 记录集以将记录副本保存到您的电脑中。

> **Important**
>
> 要使用 Amazon SES 成功设置自定义 MAIL FROM 域，您必须仅将一条 MX 记录发布到您的 MAIL FROM 域的 DNS 服务器。如果 MAIL FROM 域有多条 MX 记录，则使用 Amazon SES 设置自定义 MAIL FROM 将失败。

如果 Route 53 为您的 MAIL FROM 域提供 DNS 服务，并且您使用与 Route 53 相同的账户登录，则选择使用 R oute 53 发布记录。 AWS Management Console DNS 记录将自动应用于您的域的 DNS 配置。

如果使用其他 DNS 提供商，则必须手动将 DNS 记录发布到 MAIL FROM 域的 DNS 服务器。将 DNS 记录添加到域的 DNS 服务器的过程因您的网络托管服务或 DNS 提供商而异。

为您的域发布 DNS 记录的过程取决于您使用的 DNS 提供商。下表包含的链接指向几个广泛使用的 DNS 提供商的文档。此列表并不详尽无遗，并不表示认可；同样，如果您的 DNS 提供商没有列出，这并不意味着他们不支持 MAIL FROM 域配置。

DNS/托管提供商名称	文档链接
GoDaddy	• MX：添加 MX 记录（外部链接） • TXT：添加 TXT 记录（外部链接）
DreamHost	• MX：如何更改我的 MX 记录？（外部链接） • TXT：如何添加自定义 DNS 记录？（外部链接）
Cloudflare	• MX：如何添加或编辑邮件或 MX 记录？（外部链接） • TXT：在 Cloudflare 中管理 DNS 记录（外部链接）
HostGator	• MX：更改 MX 记录 - Windows（外部链接） • TXT：使用 HostGator /eNOM 管理 DNS 记录（外部链接）
Namecheap	• MX：如何设置邮件服务所需的 MX 记录？（外部链接） • TXT：如何为我的域名添加TXT/SPF/DKIM/DMARC记录？（外部链接）
Names.co.uk	• MX：更改您的域的 DNS 设置（外部链接） • TXT：更改您的域的 DNS 设置（外部链接）
Wix	• MX：在您的 Wix 账户中添加或更新 MX 记录（外部链接） • TXT：在您的 Wix 账户中添加或更新 TXT 记录（外部链接）

当 Amazon SES 检测到记录已生效时，您会收到一封电子邮件，通知您自定义 MAIL FROM 域已成功设置。视您的 DNS 提供商而定，Amazon SES 检测 MX 记录之前可能会延迟最多 72 个小时。

使用 Amazon SES 的自定义 MAIL FROM 域设置状态

在您将身份配置为使用自定义 MAIL FROM 域后，设置的状态为 pending (待处理)，并且 Amazon SES 会尝试在您的 DNS 设置中检测所需的 MX 记录。之后，状态因 Amazon SES 是否检测到 MX 记录而异。下表介绍了电子邮件发送行为以及与每个状态关联的 Amazon SES 操作。每当状态发生变化时，Amazon SES 都会向与您关联的电子邮件地址发送通知 AWS 账户。

状态	电子邮件发送行为	Amazon SES 操作
待处理	使用自定义 MAIL FROM 回退设置	Amazon SES 尝试在 72 小时内检测所需的 MX 记录。如果失败，则状态将更改为“失败”。
成功	使用自定义 MAIL FROM 域	Amazon SES 持续检查所需的 MX 记录是否到位。
Temporary Failure	使用自定义 MAIL FROM 回退设置	Amazon SES 尝试在 72 小时内检测所需的 MX 记录。如果失败，则状态将更改为“已失败”；如果成功，则状态将更改为“成功”。

状态	电子邮件发送行为	Amazon SES 操作
失败	使用自定义 MAIL FROM 回退设置	Amazon SES 不再尝试检测所需的 MX 记录。要使用自定义 MAIL FROM 域，您必须重新开始配置自定义 MAIL FROM 域中的设置过程。

遵守 Amazon SES 中的 DMARC 身份验证协议

基于域名的邮件认证、报告和一致性 (DMARC) 是一种电子邮件认证协议，它使用发件人策略框架 (SPF) 和 DomainKeys 识别邮件 (DKIM) 来检测电子邮件欺骗和网络钓鱼。为了符合 DMARC 的要求，消息必须通过 SPF 或 DKIM 进行身份验证，但理想情况下，当两者都与 DMARC 结合使用时，您将确保您的电子邮件发送获得尽可能高级别的保护。

让我们简要回顾一下每种方法的作用，以及 DMARC 是如何将它们联系在一起的：

- SPF – 标识哪些邮件服务器可以代表您的自定义 MAIL FROM 域，通过 DNS 使用的 DNS TXT 记录发送邮件。收件人邮件系统参考 SPF TXT 记录，以确定来自您自定义域的消息是否来自授权的消息发送服务器。基本上，SPF 的设计初衷是帮助防止欺骗，但实际上存在 SPF 容易遭受的欺骗技术，这就是为什么您还需要结合 DMARC 使用 DKIM 的原因。
- DKIM – 在电子邮件标头中为您的出站消息添加数字签名。接收电子邮件系统可以使用此数字签名来协助验证传入的电子邮件是否由该域拥有的密钥签名。但是，当接收电子邮件系统转发消息时，消息的信封会被更改，从而使 SPF 身份验证失效。由于数字签名是电子邮件标头的一部分，会随电子邮件消息一同保留，因此即使在邮件服务器之间转发消息（只要邮件内容未被修改），DKIM 也能发挥作用。
- DMARC – 确保域与至少一个 SPF 和 DKIM 保持一致。仅使用 SPF 和 DKIM 并不能确保发件人地址经过身份验证（这是您的收件人在其电子邮件客户端中看到的电子邮件地址）。SPF 仅检查 MAIL FROM 地址中指定的域（您的收件人看不到此域）。DKIM 仅检查 DKIM 签名中指定的域（您的收件人也看不到此域）。DMARC 通过要求 SPF 或 DKIM 上的域一致性正确来解决这两个问题：

- 为了使 SPF 通过 DMARC 一致性检查，发件人地址中的域必须与 MAIL FROM 地址（也称为 Return-Path 和 Envelope-from 地址）中的域匹配。在转发邮件时，这几乎不可能实现，因为转发邮件会删除原始的发件人地址信息；在使用第三方批量电子邮件提供商发送邮件时也不可能实现，因为 Return-Path（MAIL FROM）用于处理退信和投诉，而提供商（SES）会使用他们自己的地址来跟踪这些退信和投诉。
- 为了使 DKIM 通过 DMARC 一致性检查，DKIM 签名中指定的域必须与发件人地址中的域相匹配。如果您使用代表您发送邮件的第三方发件人或服务，则可以通过确保第三方发件人已正确配置为使用 DKIM 签名，并且已在域中添加了相应的 DNS 记录来实现。然后，接收邮件服务器将能够验证由您的第三方发送的电子邮件，就好像该电子邮件是由授权使用域内地址的人发送的邮件一样。

与 DMARC 协同运行

我们在上面讨论的 DMARC 一致性检查展示了 SPF、DKIM 和 DMARC 是如何协同工作的，以增加对您的域的信任以及确保将您的电子邮件传送到收件箱。DMARC 通过确保收件人看到的发件人地址已通过 SPF 或 DKIM 进行身份验证，来实现这一目标：

- 如果所述的 SPF 或 DKIM 检查中的一个或两个都成功，则消息通过 DMARC 验证。
- 如果所述的 SPF 或 DKIM 检查均失败，则邮件将无法通过 DMARC 验证。

因此，为了让 DMARC 能够最大限度地对您发送的电子邮件进行身份验证，SPF 和 DKIM 都是必不可少的，通过同时使用这三者，有助于确保您的发送域得到全面保护。

DMARC 还允许您通过设置的策略，指示电子邮件服务器在 DMARC 身份验证失败时如何处理电子邮件。下一节（the section called “对域设置 DMARC 策略”）将对此进行解释，其中包含有关如何配置您的 SES 域，从而使您发送的电子邮件同时通过 SPF 和 DKIM 符合 DMARC 身份验证协议的信息。

对域设置 DMARC 策略

要设置 DMARC，您必须修改域的 DNS 设置。域的 DNS 设置应包括指定域的 DMARC 设置的 TXT 记录。向 DNS 配置中添加 TXT 记录的过程取决于您使用的 DNS 或托管提供商。如果您使用 Route 53，请参阅《Amazon Route 53 开发人员指南》中的使用记录。如果使用的是其他提供商的产品，请参阅提供商的 DNS 配置文档。

您创建的 TXT 记录的名称应为 `_dmarc.`*`example.com`*，其中 *`example.com`* 是域。TXT 记录的值包含应用于域的 DMARC 策略。以下是包含 DMARC 策略的 TXT 记录的示例：

名称	类型	值
_dmarc.example.com	TXT	"v=DMARC1;p=quarantine;rua=mailto:my_dmarc_report@example.com"

在前面的 DMARC 策略示例中，该策略要求电子邮件提供商执行以下操作：

- 对于任何未通过身份验证的消息，请按策略参数 p=quarantine 的指定将其发送到垃圾邮件文件夹。其他选项包括使用 p=none 不执行任何操作，或者使用 p=reject 直接拒绝消息。
  - 下一节将讨论如何以及何时使用这三种策略设置 - 在错误的时间使用错误的策略设置，可能会导致您的电子邮件无法传送，请参阅the section called “实施 DMARC”。
- 根据报告参数 rua=mailto:my_dmarc_report@example.com（rua 代表聚合报告的报告 URI）的规定，以摘要形式（即汇总某一时间段内的数据并生成一份报告，而不是为每个事件单独发送报告）发送所有身份验证失败的电子邮件的报告。电子邮件提供商通常每天发送一次此类汇总报告，但具体政策因提供商而异。

要了解更多有关如何为域配置 DMARC 的信息，请参阅 DMARC 网站上的概述。

有关 DMARC 系统的完整规范，请参阅国际互联网工程任务组（IETF）DMARC 草案。

实施 DMARC 的最佳实践

最好是分阶段逐步实施您的 DMARC 策略执行，以免影响其他邮件流。制定并实施一个遵循以下步骤的推广计划。在进行下一步之前，先对每个子域执行这些步骤，最后再对组织中的顶级域执行。

1. 监控实施 DMARC 的影响（p=none）。
   - 首先，为一个子域或域创建一个简单的监控模式记录，该记录要求邮件接收组织向您发送他们使用该域观察到的消息的统计信息。监控模式记录是一个 DMARC TXT 记录，其策略设置为无 p=none。
   - 通过 DMARC 生成的报告将提供通过和未通过这些检查的消息的数量和来源。您可以轻松地了解您的合法流量受这些检查覆盖的情况。您会看到转发的迹象，因为如果内容被修改，转发的消息将无法通过 SPF 和 DKIM 检查。您还将开始看到发送了多少欺诈性消息，以及这些消息是从哪里发送的。

- 此步骤的目标是了解在实施接下来的两个步骤中的任何一个时，哪些电子邮件会受到影响，并确保任何第三方或授权发件人调整其 SPF 或 DKIM 策略以保持一致。
- 最适合现有域的使用。

2. 请求外部邮件系统隔离未通过 DMARC 的邮件（p=quarantine）。
   - 当您认为您所有或大部分的合法流量均已通过 SPF 或 DKIM 实现了域对齐，并且您了解了实施 DMARC 的影响后，您可以实施隔离策略。隔离策略是一种 DMARC TXT 记录，其策略设置为隔离 `p=quarantine`。通过这样做，您要求 DMARC 接收方将您域下未能通过 DMARC 的邮件放入本地等同于垃圾邮件文件夹的位置，而不是您客户的收件箱。
   - 最适合用于在第 1 步期间已经分析了 DMARC 报告的域转换。
3. 请求外部邮件系统不接受未通过 DMARC 的邮件（p=reject）。
   - 实施拒绝策略通常是最后一步。拒绝策略是指其策略设置为拒绝 `p=reject` 的 DMARC TXT 记录。当您这样做时，您要求 DMARC 接收方不接受未能通过 DMARC 检查的消息，这意味着这些消息甚至不会被隔离到垃圾邮件文件夹中，而是会被直接拒绝。
   - 当使用拒绝策略时，由于拒绝会导致 SMTP 退信，您将能够确切地知道哪些消息未能通过 DMARC 策略。而使用隔离策略时，汇总数据会提供有关通过或未通过 SPF、DKIM 和 DMARC 检查的电子邮件百分比的信息。
   - 最适合已完成前两个步骤的新域或现有域。

通过 SPF 遵守 DMARC

要使电子邮件基于 SPF 遵守 DMARC，必须满足以下两个条件：

- 该消息必须基于您已发布到自定义 MAIL FROM 域的 DNS 配置中的有效 SPF（类型 TXT）记录，通过 SPF 检查。
- 电子邮件标头中收件人地址的域必须与 MAIL FROM 地址中指定的域或其子域一致（匹配）。为了实现与 SES 的 SPF 对齐，域的 DMARC 策略不得指定严格的 SPF 策略（aspf=s）。

要满足上述要求，请完成以下步骤：

- 通过完成 the section called “使用自定义 MAIL FROM 域” 中的过程设置自定义 MAIL FROM 域。
- 确保您的发送域使用宽松的 SPF 策略。如果您尚未更改域的策略遵守方式，它将默认使用宽松的策略，就像 SES 一样。

> **Note**
>
> 您可以在命令行中输入以下命令，将 *example.com* 替换为您的域名，从而确定您的域的 DMARC 与 SPF 的符合情况：
>
> ```
> dig TXT _dmarc.example.com
> ```
>
> 在此命令的输出中，在 Non-authoritative answer 下查找以 v=DMARC1 开头的记录。如果此记录包含字符串 aspf=r，或者 aspf 字符串根本不存在，则您的域为 SPF 使用宽松遵守。如果记录包含字符串 aspf=s，则您的域为 SPF 使用严格遵守。您的系统管理员需要从域的 DNS 配置的 DMARC TXT 记录中删除此标签。
> 或者，你可以使用基于网络的DMARC查询工具，例如 dmarcian网站上的DMARC In spector 或网站上的 DMARC检查工具来确定你的域名与SPF的政策一致性。 MxToolBox

通过 DKIM 遵守 DMARC

要使电子邮件基于 DKIM 遵守 DMARC，必须满足以下两个条件：

- 消息必须具有有效的 DKIM 签名并通过 DKIM 检查。
- DKIM 签名中指定的域必须与发件人地址中的域一致（匹配）。如果域的 DMARC 策略指定了 DKIM 的严格遵守方式，则这些域必须完全匹配（SES 默认使用严格的 DKIM 策略）。

要满足上述要求，请完成以下步骤：

- 通过完成 the section called “Easy DKIM” 中的过程设置 Easy DKIM。使用 Easy DKIM 时，Amazon SES 会自动对您的电子邮件签名。

> **Note**
>
> 除了使用 Easy DKIM 以外，您也可以为邮件手动签名。但是，如果您选择这样做，请务必小心，因为 Amazon SES 不验证您构建的 DKIM 签名。因此，我们强烈建议您使用 Easy DKIM。

- 确保 DKIM 签名中指定的域与发件人地址中的域一致。或者，如果从发件人地址中域的子域发送，请确保您的 DMARC 策略设置为宽松的一致性。

> **Note**
>
> 您可以在命令行中输入以下命令，将 *example.com* 替换为您的域名，从而确定您的域的 DMARC 与 DKIM 的符合情况：
>
> ```
> dig TXT _dmarc.example.com
> ```
>
> 在此命令的输出中，在 Non-authoritative answer 下查找以 v=DMARC1 开头的记录。如果此记录包含字符串 adkim=r，或者 adkim 字符串根本不存在，则您的域为 DKIM 使用宽松遵守。如果记录包含字符串 adkim=s，则您的域为 DKIM 使用严格遵守。您的系统管理员需要从域的 DNS 配置的 DMARC TXT 记录中删除此标签。
> 或者，你可以使用基于网络的DMARC查询工具，例如d marcian网站上的DMARC In spector 或网站上的 DMARC检查工具来确定你的域名与DKIM的政策一致性。 MxToolBox

在 Amazon SES 中使用 BIMI

用于邮件识别的品牌指标（BIMI）是一种电子邮件规范，它使电子邮件收件箱能够在支持的电子邮件客户端中将品牌徽标显示在该品牌经过身份验证的电子邮件旁边。

BIMI 是一种直接连接到身份验证的电子邮件规范，但它不是一个独立的电子邮件身份验证协议，因为它要求所有电子邮件都符合 DMARC 身份验证。

虽然BIMI需要DMARC，但DMARC要求你的域名必须有SPF或DKIM记录才能保持一致，但为了提高安全性，最好同时包含SPF和DKIM记录，也因为一些电子邮件服务提供商（ESPs）在使用BIMI时需要同时包含SPF和DKIM记录。以下章节介绍在 Amazon SES 中实现 BIMI 的步骤。

在 SES 中设置 BIMI

您可以为您拥有的电子邮件域配置 BIMI，在 SES 中，此类域称为自定义 MAIL FROM 域。配置完成后，您从该域发送的所有邮件将在支持 BIMI 的电子邮件客户端中显示您的 BIMI 徽标。

要使您的电子邮件显示 BIMI 徽标，就必须在 SES 中满足一些先决条件：在以下过程中，这些先决条件是概括性的，并将引用详细介绍这些主题的专门章节。这里将详细介绍特定于 BIMI 的步骤以及在 SES 中对其进行配置所需的内容。

在自定义 MAIL FROM 域中设置 BIMI

1. 必须在 SES 中配置自定义 MAIL FROM 域，并同时为该域发布 SPF（类型 TXT）和 MX 记录。如果您没有自定义 MAIL FROM 域，或者想为您的 BIMI 徽标创建一个新的域，请参阅 the section called “使用自定义 MAIL FROM 域”。
2. 使用 Easy DKIM 配置您的域。请参阅 the section called “Easy DKIM”。
3. 通过向 DNS 提供商发布 TXT 记录（其中包含 BIMI 所需的以下执行策略细节），使用 DMARC 配置您的域，类似于以下两个示例之一：

名称	类型	值
`_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`

在前面的 BIMI 所需的 DMARC 策略示例中：

- *`example.com`* 应替换为您的域或子域。
- p= 值可以是：
  - quarantine，pct 值设置为 100，如上所示，或者
  - reject，如上所示。
- 如果您从子域发送，BIMI 要求父域也必须具有此执行策略。子域将属于父域的策略范围。但是，如果您除了为父域发布的内容外，还为您的子域添加了 DMARC 记录，则您的子域也必须具有相同的执行策略才能有资格获得 BIMI。
- 如果您从未为您的域设置 DMARC 策略，请参阅the section called “使用 DMARC 对电子邮件进行身份验证”，确保您只使用特定于 BIMI 的 DMARC 策略值，如上所示。

4. 将你的 BIMI 徽标制作成可扩展矢量图形 (SVG) `.svg` 文件——BIMI 所需的特定 SVG 配置文件定义为 SVG）。Portable/Secure (SVG P/S为了使您的徽标显示在电子邮件客户端中，它必须完全符合这些规范。请参阅 BIMI Group 有关创建 SVG 徽标文件的指南和建议使用的 SVG 转换工具。
5. （可选）获取认证标志证书（VMC）。有些公司 ESPs，例如Gmail和Apple，要求VMC提供证据，证明您拥有BIMI徽标的商标和内容。尽管这不是在您的域上实施 BIMI 的必要条件，但如果您

向其发送邮件的 ESP 强制要求遵守 VMC，则您的 BIMI 徽标将不会显示在电子邮件客户端中。请参阅 BIMI Group 的有关参与证书颁发机构的参考资料，为您的徽标获取 VMC。

6. 将您的 BIMI 徽标的 SVG 文件托管在您可以访问的服务器上，使其可通过 HTTPS 公开访问。例如，您可以将其上传到 Amazon S3 桶。
7. 创建并发布包含您的徽标 URL 的 BIMI DNS 记录。当支持 BIMI 的 ESP 检查您的 DMARC 记录时，它还会查找一个 BIMI 记录，其中包含您徽标的 `.svg` 文件的 URL，如果已配置，则还包括 VMC 的 `.pem` 文件的 URL。如果记录匹配，它们将显示您的 BIMI 徽标。

   使用 BIMI 配置您的域，方法是向您的 DNS 提供商发布具有如下所示的值的 TXT 记录：从域发送（如第一个示例所示）；从子域发送（如第二示例所示）：

名称	类型	值
`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`		

   在前面的 BIMI 记录示例中：

   - 名称值应按字面意思将 `default._bimi.` 指定为 *`example.com`* 的子域或 *`marketing.example.com`*，或者应将其替换为您的域或子域。
   - `v=` 值是 BIMI 记录的版本。
   - `l=` 值是一个徽标，表示指向图像的 `.svg` 文件的 URL。
   - `a=` 值是一个颁发机构，表示指向您的证书的 `.pem` 文件的 URL。

   您可以使用 BIMI Group 的 BIMI Inspector 等工具验证您的 BIMI 记录。

此过程的最后一步是定期向支持BIMI徽标放置的 ESPs 发送模式。你的域名应该有规律的交付节奏，并且应该在你要发送的域名 ESPs 中享有良好的声誉。BIMI徽标的放置可能需要一段时间才能填充到你没有既定声誉或发送节奏 ESPs 的地方。

您可以通过 BIMI Group 组织找到更多与 BIMI 有关的信息和资源。

为 Amazon SES 设置事件通知

要使用 Amazon SES 来发送电子邮件，必须有一个系统可用于管理退回邮件和投诉。Amazon SES 可通过以下三种方式通知退回邮件或投诉事件：发送通知电子邮件、通知 Amazon SNS 主题或发布发送事件。本节包含有关将 Amazon SES 设置为通过电子邮件或通过通知 Amazon SNS 主题来发送特定类型通知的信息。有关发布发送事件的更多信息，请参阅使用 Amazon SES 事件发布监控电子邮件发送。

可以使用 Amazon SES 控制台或 Amazon SES API 来设置通知。

主题

重要注意事项

设置 Amazon SES 发送通知时，有几点务必要注意：

- 电子邮件和 Amazon SNS 通知适用于独立身份（用于发送电子邮件的已验证电子邮件地址或域）。当您为某一身份启用通知时，Amazon SES 仅针对从该身份发送的电子邮件发送通知，并且仅在您配置通知的 AWS 区域发送通知。
- 必须启用一种接收退回邮件或投诉通知的方式。可以将通知发送到生成退回邮件或投诉的域或电子邮件地址，或者发送到 Amazon SNS 主题。还可以使用事件发布将有关各种不同类型事件（包括退回邮件、投诉、送达等）的通知发送到 Amazon SNS 主题或 Firehose 流。

  如果未设置任何一种接收退回邮件或投诉通知的方法，Amazon SES 会自动将退回邮件和投诉通知转发到出现退回邮件或投诉事件的电子邮件中的退回路径地址（或来源地址，如果未指定退回路径地址），即使禁用了电子邮件反馈转发也是如此。

  如果禁用电子邮件反馈转发并启用事件发布，则必须对发送的所有电子邮件应用包含事件发布规则的配置集。在此情况下，如果不使用配置集，Amazon SES 会将退回邮件和投诉通知转发到出现退回邮件或投诉事件的电子邮件中的退回路径或来源地址。
- 如果将 Amazon SES 设置为使用多种方法（例如通过发送电子邮件通知和使用发送事件）发送退回邮件和投诉事件，那么可能收到有关同一事件的多个通知。

通过电子邮件接收 Amazon SES 通知

当您收到退回邮件和投诉时，Amazon SES 可以使用称为电子邮件反馈转发的流程为您发送电子邮件。

要使用 Amazon SES 来发送电子邮件，必须将它配置为使用以下方法之一发送退回邮件和投诉通知：

- 启用电子邮件反馈转发。此节中包含此类通知的设置过程。
- 将通知发送到 Amazon SNS 主题。有关更多信息，请参阅 使用 Amazon SNS 接收 Amazon SES 通知。
- 发布事件通知。有关更多信息，请参阅 使用 Amazon SES 事件发布监控电子邮件发送。

> **Important**
>
> 有关通知的若干要点，请参阅为 Amazon SES 设置事件通知。

主题

启用电子邮件反馈转发

默认情况下会启用电子邮件反馈转发。如果您之前禁用了该功能，您可以使用本节中的以下步骤来启用它。

使用 Amazon SES 控制台通过电子邮件启用退回邮件和投诉转发

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择已验证身份。
3. 在经验证的电子邮件地址或域的列表中，选择要为其配置退回邮件和投诉通知的电子邮件地址或域。
4. 在详细信息窗格中，展开通知部分。
5. 选择 Edit Configuration。

6. 在 Email Feedback Forwarding 下，选择 Enabled。

> **Note**
>
> 在此页面上进行的更改可能需要几分钟才能生效。

您还可以使用 SetIdentityFeedbackForwardingEnabledAPI 操作通过电子邮件启用退回通知和投诉通知。

禁用电子邮件反馈转发

如果设置了其他提供退回邮件和投诉通知的方法，则可禁用电子邮件反馈转发，以便在退回邮件或投诉事件发生时，不会收到多个通知。

使用 Amazon SES 控制台通过电子邮件禁用退回邮件和投诉转发

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择已验证身份。
3. 在经验证的电子邮件地址或域的列表中，选择要为其配置退回邮件和投诉通知的电子邮件地址或域。
4. 在详细信息窗格中，展开通知部分。
5. 选择 Edit Configuration。
6. 在 Email Feedback Forwarding 下，选择 Disabled。

> **Note**
>
> 必须配置一种接收退回邮件和投诉通知的方法，才能通过 Amazon SES 发送电子邮件。如果禁用电子邮件反馈转发，则必须启用通过 Amazon SNS 发送通知，或者使用事件发布将退回邮件和投诉事件发送到 Amazon SNS 主题或 Firehose 流。如果使用事件发布，还必须对发送的每封电子邮件应用包含事件发布规则的配置集。如果未设置接收退回邮件和投诉通知的方法，Amazon SES 将自动通过电子邮件将反馈通知转发到 Return-Path (退回路径) 字段（或 Source (来源) 字段，如果未指定退回路径地址）中发生退回邮件或投诉事件的邮件的地址。在此情况下，即使禁用了电子邮件反馈通知，Amazon SES 也会转发退回邮件和投诉通知。

7. 保存您的通知配置，请选择 Save Config。

> **Note**
>
> 在此页面上进行的更改可能需要几分钟才能生效。

您还可以使用 SetIdentityFeedbackForwardingEnabledAPI 操作禁用通过电子邮件发送的退信和投诉通知。

电子邮件反馈转发目标

当您通过电子邮件接收通知时，Amazon SES 会重写 From 标头，并向您发送通知。Amazon SES 将通知转发到的地址取决于您发送原始邮件的方式。

如果您使用了 SMTP 接口发送邮件，则通知会根据以下规则传送：

- 如果您在 SMTP DATA 部分指定了 Return-Path 标头，通知将发送至该地址。
- 否则，通知将发送到您在发出 MAIL FROM 命令时指定的地址。

如果您是使用 SendEmail API 操作发送邮件的，则通知会根据以下规则传送：

- 如果您向 SendEmail API 调用中指定了可选 ReturnPath 参数，则通知会发送至该地址。
- 否则，通知将发送至 Source 的必需 SendEmail 参数中指定的地址。

如果您是使用 SendRawEmail API 操作发送邮件的，则通知会根据以下规则传送：

- 如果您在原始邮件中指定了 Return-Path 标头，通知将发送至该地址。
- 否则，如果您在对 SendRawEmail API 的调用中指定了 Source 参数，则通知会发送至该地址。
- 不然，通知会发送至原始邮件 From 标头中的地址。

> **Note**
>
> 当您指定电子邮件中的 Return-Path 地址时，该地址会收到通知。但是，收件人收到的消息版本中包含 Return-Path 标头，其中包含一个匿名电子邮件地址（如 a0b1c2d3e4f5a6b7-c8d9e0f1-a2b3-c4d5-e6f7-a8b9c0d1e2f3-000000@amazonses.com）。无论您以何种方式发送电子邮件，该电子邮件地址均为匿名。

使用 Amazon SNS 接收 Amazon SES 通知

可以将 Amazon SES 配置为在您收到退回邮件或投诉时或者传递电子邮件时通知 Amazon SNS 主题。Amazon SNS 通知采用JavaScript 对象表示法 (JSON) 格式，这使您能够以编程方式处理通知。

要使用 Amazon SES 来发送电子邮件，必须将它配置为使用以下方法之一发送退回邮件和投诉通知：

- 将通知发送到 Amazon SNS 主题。此节中包含此类通知的设置过程。
- 启用电子邮件反馈转发。有关更多信息，请参阅 通过电子邮件接收 Amazon SES 通知。
- 发布事件通知。有关更多信息，请参阅 使用 Amazon SES 事件发布监控电子邮件发送。

> **Important**
>
> 有关通知的重要信息，请参阅为 Amazon SES 设置事件通知。

主题

为 Amazon SES 配置 Amazon SNS 通知

Amazon SES 可以通过 Amazon Simple Notification Service（Amazon SNS）向您通知有关退回邮件、投诉和送达的情况。

您可以在 Amazon SES 控制台中或者使用 Amazon SES API 来配置通知。

本节中的主题：

先决条件

在 Amazon SES 中设置 Amazon SNS 通知之前，请完成以下步骤：

1. 在 Amazon SNS 中创建主题。有关更多信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的创建主题。

> **Important**
>
> 当您使用 Amazon SNS 创建主题时，对于 Type (类型)，仅选择 Standard (标准)。（SES 不支持 FIFO 类型主题。）

无论是创建新的 SNS 主题还是选择现有主题，都需要授予 SES 访问权限，才能向该主题发布通知。

要授予 Amazon SES 发布主题通知的权限，请在 SNS 控制台的编辑主题屏幕上，展开访问策略，并在 JSON 编辑器中，添加以下权限策略：

JSON

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

对前面的策略示例进行以下更改：

- *topic_region*替换为您创建 SNS 主题的 AWS 区域。
- 将 *111122223333* 替换为您的 AWS 账户 ID。
- *topic_name*替换为您的 SNS 主题的名称。
- *identity_name*替换为您订阅 SNS 主题的经过验证的身份（电子邮件地址或域名）。

2. 使用至少一个终端节点订阅主题。例如，如果您希望通过短信接收通知，则使用 SMS 终端节点（即，移动电话号码）订阅主题。要通过电子邮件接收通知，使用电子邮件终端节点（电子邮件地址）订阅主题。

   有关更多信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的入门。

3. （可选）如果您的 Amazon SNS 主题使用 AWS Key Management Service (AWS KMS) 进行服务器端加密，则必须向密钥策略添加权限。 AWS KMS 您可以通过将以下策略附加到 AWS KMS 密钥策略来添加权限：

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSESToUseKMSKey",
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

使用 Amazon SES 控制台来配置通知

使用 Amazon SES 控制台来配置通知

1. 打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择身份。
3. 在 Identities（身份）容器中，选择您希望在从此身份发送的消息导致退回、投诉或送达时接收反馈通知的已验证身份。

> **Important**
>
> 经验证的域通知设置将应用于从该域中的电子邮件址（经验证的电子邮件地址除外）发送的所有邮件。

4. 在您选择的已验证身份的详细信息屏幕中，选择 Notifications（通知）选项卡，然后选择 Feedback notifications（反馈通知）容器中的 Edit（编辑）。
5. 展开要接收通知的每种反馈类型的 SNS 主题列表框，然后选择您拥有的 SNS 主题之一、No SNS topic（无 SNS 主题），或者 SNS topic you don't own（你不拥有的 SNS 主题）。
   - 如果您选择了 SNS topic you don't own（您不拥有的 SNS 主题），SNS topic ARN（SNS 主题 ARN）将显示字段，您必须在其中输入委托发件人与您共享的 SNS 主题 ARN。（只有您的委托发件人才会收到这些通知，因为他们拥有 SNS 主题。要了解有关委托发送的更多信息，请参阅发送授权概览。）

> **Important**
>
> 您用于退货、投诉和送达通知的 Amazon SNS 主题必须与您使用 Amazon SES 时 AWS 区域 使用的主题相同。
> 此外，您必须订阅主题的一个或多个终端节点才能接收通知。例如，如果您要将通知发送到电子邮件地址，则必须使用电子邮件终端节点订阅主题。有关更多信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的入门。

6. （可选）如果您希望主题通知包括原始电子邮件中的标头，请选中 Include original email headers（包括原始电子邮件标头）框直接位于每种反馈类型的 SNS 主题名称下方。此选项仅在您已将 Amazon SNS 主题分配到关联的通知类型时可用。有关原始电子邮件标头的内容的信息，请参阅 通知内容中的 `mail` 对象。
7. 选择 Save changes（保存更改）。您对通知设置所做的更改可能需要几分钟才能生效。

8. （可选）如果您为退回邮件和投诉都选择 Amazon SNS 主题通知，则可以完全禁用电子邮件通知，以免通过电子邮件和 SNS 通知接收双重通知。要禁用退回邮件和投诉的电子邮件通知，请在 Email Feedback Forwarding（电子邮件反馈转发）容器中已验证身份的详细信息屏幕中的 Notifications（通知）选项卡下方选择 Edit（编辑）、取消勾选 Enabled（已启用）方框，然后选择 Save changes（保存更改）。

在配置您的设置之后，您将开始接收 Amazon SNS 主题的退回邮件、投诉和送达通知。这些通知采用 JavaScript对象表示法 (JSON) 格式，并遵循中描述的结构通知内容。

您将需要按标准 Amazon SNS 费率为退回邮件、投诉和送达通知付费。有关更多信息，请参阅 Amazon SNS 定价页面。

> **Note**
>
> 如果由于主题已被删除或您 AWS 账户 不再有权向该主题发布内容而尝试向该主题发布内容失败，Amazon SES 会删除该主题的配置，前提是该主题已配置为退回或投诉（不是传送——对于送达通知，SES 不会删除 SNS 主题配置设置）。此外，Amazon SES 还会为身份重新启用退回邮件和投诉电子邮件通知，并且您将通过电子邮件收到更改通知。如果将多个身份配置为使用该主题，则在每个身份无法发布到该主题时，更改每个身份的主题配置。

使用 Amazon SES API 来配置通知

您也可以使用 Amazon SES API 来配置退回邮件、投诉和送达通知。使用以下操作来配置通知：

- SetIdentityNotificationTopic
- SetIdentityFeedbackForwardingEnabled
- GetIdentityNotificationAttributes
- SetIdentityHeadersInNotificationsEnabled

您可以使用这些 API 操作编写用于通知的自定义前端应用程序。有关与通知相关的 API 操作的完整描述，请参阅 Amazon Simple Email Service API 参考。

反馈通知故障排除

未收到通知

如果您没有收到通知，请确保您已使用终端节点订阅发送通知的主题。当您使用电子邮件端点节点订阅主题时，您会收到一封电子邮件，要求您确认订阅。在开始接收电子邮件通知之前，您必须确认订阅。有关更多信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的入门。

`InvalidParameterValue`选择主题时的 错误

如果您收到表明发生 `InvalidParameterValue` 错误的错误消息，请检查 Amazon SNS 主题以查看它是否使用 AWS KMS进行了加密。如果是，则必须修改 AWS KMS 密钥的策略。请参阅 先决条件 以获取示例策略。

为 Amazon SES 配置 Amazon SNS 通知

退货、投诉和送达通知以 JavaScript 对象表示法 (JSON) 格式发布到亚马逊简单通知服务 (Amazon SNS) Simple Notification S ervice 主题。顶级 JSON 对象包含一个 `notificationType` 字符串，一个 `mail` 对象，以及一个 `bounce` 对象、`complaint` 对象或 `delivery` 对象。

有关不同类型对象的描述，请参阅以下部分：

- 顶级 JSON 对象
- `mail` 对象
- `bounce` 对象
- `complaint` 对象
- `delivery` 对象

以下是有关 Amazon SES 的 Amazon SNS 通知内容的一些重要说明：

- 对于给定的通知类型，您可能会收到针对多个收件人的 Amazon SNS 通知，或者可能会收到针对每个收件人的 Amazon SNS 通知。您的代码应能够解析 Amazon SNS 通知并处理这两种情况；Amazon SES 不保证对通过 Amazon SNS 发送的通知进行排序或批处理。但是，不同的 Amazon SNS 通知类型（例如，退回邮件和投诉）不会合并为一个通知。
- 您可能会针对一个收件人收到多个类型的 Amazon SNS 通知。例如，接收邮件服务器可能接受电子邮件 (触发送达通知)，但在处理电子邮件后，接收邮件服务器可能会确定该电子邮件实际导致了退回邮件 (触发退回邮件通知)。不过，这些通知始终单独发送，因为它们属于不同的类型。
- Amazon SES 保留在通知中添加其他字段的权利。同样地，解析这些通知的应用程序必须足够灵活以处理未知字段。
- Amazon SES 在发送电子邮件时会覆盖邮件标头。您可以通过 `mail` 对象的 `headers` 和 `commonHeaders` 字段检索原始邮件的标头。

顶级 JSON 对象

Amazon SES 通知中的顶级 JSON 对象包含以下字段。

字段名称	描述
`notificationType`	一个字符串，包含由 JSON 对象表示的通知类型。可能的值为 `Bounce`、`Complaint` 或 `Delivery`。 如果你设置事件发布，此字段命名为 `eventType` 。
`mail`	一个 JSON 对象，包含有关通知所属的原始邮件的信息。有关更多信息，请参阅 邮件对象。
`bounce`	此字段仅在 `notificationType` 为 `Bounce`，且包含的 JSON 对象保存有关退回邮件的信息时显示。有关更多信息，请参阅 退回邮件对象。
`complaint`	此字段仅在 `notificationType` 为 `Complaint` ，且包含的 JSON 对象保存有关投诉的信息时显示。有关更多信息，请参阅 投诉对象。
`delivery`	此字段仅在 `notificationType` 为 `Delivery`，且包含的 JSON 对象保存有关送达的信息时显示。有关更多信息，请参阅 送达对象。

邮件对象

每个退回邮件、投诉或送达通知均在 `mail` 对象中包含有关原始电子邮件的信息。包含有关 `mail` 对象的信息的 JSON 对象具有以下字段。

字段名称	描述
timestamp	原始消息的发送时间（采用 ISO86 01 格式）。
messageId	Amazon SES 分配给电子邮件的唯一 ID。Amazon SES 在您发送邮件时已向您返回此值。 **Note** 此邮件 ID 已由 Amazon SES 分配。您可以在 mail 对象的 headers 字段中找到原始电子邮件的邮件 ID。
source	发送原始电子邮件的电子邮件地址 (信封 MAIL FROM 地址)。
sourceArn	已用于发送电子邮件的身份的 Amazon Resource Name（ARN）。在发送授权的情况下，sourceArn 是身份拥有者授权委托发件人用于发送电子邮件的身份的 ARN。有关发送授权的更多信息，请参阅电子邮件身份验证方法。
sourceIp	已执行到 Amazon SES 的电子邮件发送请求的客户端的原始公有 IP 地址。
sendingAccountId	用于发送电子邮件的账户的 AWS 账户 ID。在发送授权的情况下，sendingAccountId 是委托发件人的账户 ID。
callerIdentity	发送电子邮件的 Amazon SES 用户的 IAM 身份。
destination	作为原始邮件的收件人的电子邮件地址的列表。
headersTruncated	仅当您将通知设置配置为包括原始电子邮件中的标头时，此对象才存在。

字段名称	描述
	指示通知中的标头是否截断。当原始邮件中的标头大小为 10 KB 或更大时，Amazon SES 会在截断通知中的标头。可能的值为 `true` 和 `false`。
`headers`	仅当您将通知设置配置为包括原始电子邮件中的标头时，此对象才存在。 电子邮件的原始标头的列表。列表中的每个标头均有一个 `name` 字段和一个 `value` 字段。 **Note** `headers` 对象内的任何邮件 ID 均来自您传递至 Amazon SES 的原始邮件。Amazon SES 随后分配给电子邮件的邮件 ID 位于 `mail` 对象的 `messageId` 字段中。
`commonHeaders`	仅当您将通知设置配置为包括原始电子邮件中的标头时，此对象才存在。 包括有关原始电子邮件中的常用电子邮件标头的信息，包括“发件人”、“收件人”和“主题”字段。在此对象中，每个标头是一个键。“发件人”和“收件人”字段由可包含多个值的数组表示。 **Note** 对于事件，`commonHeaders` 字段中的任何邮件 ID 是 Amazon SES 随后在邮件对象的 `messageId` 字段中分配给邮件的邮件 ID。通知将包含原始电子邮件的邮件 ID。

下面是包含原始电子邮件标头的 mail 对象的示例。当此通知类型未配置为包含原始电子邮件标头时，mail 对象不会包含 headersTruncated、headers 和 commonHeaders 字段。

```
{
   "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"
   }
}
```

退回邮件对象

包含有关退回邮件的信息的 JSON 对象具有以下字段。

字段名称	描述
bounceType	退回邮件的类型，由 Amazon SES 确定。有关更多信息，请参阅 退信类型。
bounceSubType	退回邮件的子类型，由 Amazon SES 确定。有关更多信息，请参阅 退信类型。
bouncedRecipients	包含已退回的原始邮件收件人相关信息的列表。有关更多信息，请参阅 退信的收件人。
timestamp	发送退件的日期和时间（采用 ISO86 01 格式）。请注意，这是 ISP 发送通知的时间，而不是 Amazon SES 接收通知的时间。
feedbackId	退信的唯一 ID。

如果 Amazon SES 已能够联系远程 Message Transfer Authority（MTA），则以下字段也将显示。

字段名称	描述
remoteMtaIp	Amazon SES 尝试将电子邮件传输到的 MTA 的 IP 地址。

如果退回邮件已附加传输状态通知 (DSN)，则以下字段也可能存在。

字段名称	描述
reportingMTA	DSN 中的 Reporting-MTA 字段的值。这是尝试执行 DSN 所述的传输、中继或网关操作的 MTA 的值。

以下是 bounce 对象的示例。

```
{
   "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"
}
```

退信的收件人

退回邮件通知可能与一个或多个收件人有关。bouncedRecipients 字段包含对象列表 (与退回邮件通知相关的每个收件人均有一个对象) 而且始终包含以下字段。

字段名称	描述
emailAddress	收件人的电子邮件地址。如果 DSN 可用，这将是 DSN 中的 Final-Recipient 字段的值。

（可选）如果 DSN 已附加到退信，则以下字段也可能存在。

字段名称	描述
action	DSN 中的 Action 字段的值。这表示 Reporting-MTA 因尝试将邮件传输至此收件人而需执行的操作。
status	DSN 中的 Status 字段的值。这是每个收件人与传输无关的状态代码，用于指示邮件的传输状态。
diagnosticCode	报告 MTA 发放的状态代码。这是 DSN 中的 Diagnostic-Code 字段的值。此字段可能不在 DSN 中（因此也不在 JSON 中）。

以下是可能位于 bouncedRecipients 列表中的对象的示例。

```
{
    "emailAddress": "recipient@example.com",
    "action": "failed",
    "status": "5.0.0",
    "diagnosticCode": "X-Postfix; unknown user"
}
```

退信类型

退回邮件对象包含的退回邮件类型为 Undetermined、Permanent 或 Transient。Permanent 和 Transient 退回邮件类型也可以包含多种退回邮件子类型之一。

当您收到退回邮件类型为 Transient 的退回邮件通知时，如果解决了导致退回邮件的问题，您也许可在以后将邮件发送给该收件人。

当您收到退回邮件类型为 Permanent 的退回邮件通知时，您以后也可能无法将电子邮件发送给该收件人。因此，您应该立即从邮件列表中删除其地址造成了退回邮件的收件人。

> **Note**
>
> 发生软退回邮件（与收件人的收件箱已满等临时问题相关的退回邮件）时，Amazon SES 会在一定时间内尝试重新传送电子邮件。在这段时间结束时，如果 Amazon SES 仍然无法传送电子邮件，则会停止尝试。
>
> Amazon SES 为查无此人的邮件以及停止尝试传送的软退回邮件提供通知。如果您希望在每次发生软退回邮件时收到通知，请启用事件推送并将其配置为出现送达延迟事件时发送通知。

bounceType	bounceSubType	描述
Undetermined	Undetermined	收件人的电子邮件提供商发送退回邮件消息。退回邮件中未包含足够的消息以供 Amazon SES 确定退回邮件的原因。退回邮件电子邮件 (发送到电子邮件的 Return-Path 标头中地址) 可能包含有关导致电子邮件退回的问题的其他信息。
Permanent	General	收件人的电子邮件提供商发送了硬退信。 **Important** 收到此类退回邮件通知时，您应立即从邮件列表中删除该收件人的电子邮件地址。将邮件发送到造成硬退回邮件的地址会对您作为发件人的声誉造成负面影响。如果您继续将电子邮件发送到造成硬退回邮件的地址，我们可能会暂停您

bounceType	bounceSubType	描述
		发送更多电子邮件的功能。请参阅the section called “使用账户级黑名单”。
Permanent	NoEmail	无法从退回邮件中检索收件人的电子邮件地址。
Permanent	Suppressed	由于收件人的电子邮件地址近期有造成查无此人的邮件的历史记录，因此该地址已进入 Amazon SES 黑名单。要覆盖全局黑名单，请参阅 使用 Amazon SES 账户级黑名单。
Permanent	OnAccountSuppressionList	Amazon SES 已禁止发送到此地址，因为该地址已被加入账户级黑名单。这不计入您的跳出率指标。
Transient	General	收件人的电子邮件提供商发送一般退回邮件消息。如果解决了导致邮件退回邮件的问题，您也许可在以后将邮件发送给相同的收件人。 **Note** 如果您将电子邮件发送到激活了自动回复规则 (例如“外出”邮件) 的收件人，则可能收到这种类型的通知。即使回复中具有 Bounce 类型的通知，Amazon SES 在计算账户的邮件退回率时也不会计入自动回复。
Transient	MailboxFull	由于收件人的收件箱已满，收件人的电子邮件提供商发送退回邮件。以后当收件人的邮箱有空闲空间时，您可能可以向相同的收件人发送电子邮件。
Transient	MessageTooLarge	由于您发送的邮件太大，收件人的电子邮件提供商发送退回邮件。如果您减小邮件的大小，则也许能够向相同的收件人发送邮件。

bounceType	bounceSubType	描述
Transient	ContentRejected	由于您发送的内容包含收件人电子邮件提供商不允许的内容，该提供商发送了退回邮件。如果您更改邮件的内容，则也许能够向相同的收件人发送电子邮件。
Transient	AttachmentRejected	由于邮件包含不可接受的附件，收件人的电子邮件提供商发送了退回邮件。例如，一些电子邮件提供商可能会拒绝具有特定文件类型附件的电子邮件，或者具有超大附件的电子邮件。如果您删除附件或更改其内容，则也许能够向相同的收件人发送电子邮件。

投诉对象

包含有关投诉的信息的 JSON 对象具有以下字段。

字段名称	描述
complainedRecipients	包含提起投诉的收件人相关信息的列表。有关更多信息，请参阅 已投诉的收件人。
timestamp	ISP 发送投诉通知时的日期和时间，采用 ISO8601 格式。此字段中的日期和时间可能与 Amazon SES 收到通知时的日期和时间不同。
feedbackId	与投诉关联的唯一 ID。
complaintSubType	complaintSubType 字段的值可以为 null 或 OnAccountSuppressionList 。如果该值为 OnAccountSuppressionList ，则表示 Amazon SES 已接受邮件，但未尝试发送邮件，因为该地址已被加入账户级黑名单。

此外，如果反馈报告已附加到投诉，则以下字段也可能存在。

字段名称	描述
userAgent	反馈报告中的 User-Agent 字段的值。此值表示生成了报告的系统的名称和版本。
complaintFeedbackType	从 ISP 收到的反馈报告中的 Feedback-Type 字段的值。此值包含反馈的类型。
arrivalDate	反馈报告中Arrival-Date 或Received-Date 字段的值（采用 ISO86 01 格式）。此字段可能不在报告中（因此也不在 JSON 中）。

以下是 complaint 对象的示例。

```
{
   "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"
}
```

已投诉的收件人

complainedRecipients 字段包含可能已提交投诉的收件人的列表。您应使用此信息来确定哪位收件人提交了投诉，然后立即将该收件人从您的邮件列表中删除。

> **Important**
>
> 大多数人会从投诉通知中 ISPs 删除提交投诉的收件人的电子邮件地址。因此，此列表包含可能发送了投诉的收件人的信息，这基于原始邮件的收件人以及我们收到投诉的 ISP。Amazon SES 对原始邮件执行查找以确定此收件人列表。

此列表中的 JSON 对象包含以下字段。

字段名称	描述
emailAddress	收件人的电子邮件地址。

以下是已投诉收件人对象的示例。

```
{ "emailAddress": "recipient1@example.com" }
```

> **Note**
>
> 由于此行为，如果限制为向每个收件人发送一封电子邮件 (而不是向“密件抄送”行中的 30 个不同的电子邮件地址发送同一封电子邮件)，您可以更加确定地知道哪个地址投诉了您的邮件。

投诉类型

根据complaintFeedbackType互联网编号分配机构网站，您可在由报告 ISP 分配的 字段中看到以下投诉类型：

- abuse - 指示不请自来的电子邮件或某种其他类型的垃圾邮件。
- auth-failure - 电子邮件身份验证失败报告。
- fraud - 指示某种欺诈或网络钓鱼活动。
- not-spam – 指示提供报告的实体不将邮件视为垃圾邮件。这可用于更正被错误地标记或分类为垃圾邮件的邮件。
- other - 指示不适合其他已注册类型的任何其他反馈。
- virus - 报告在原始邮件中发现病毒。

送达对象

包含送达情况相关信息的 JSON 对象始终具有以下字段。

字段名称	描述
timestamp	Amazon SES 将电子邮件发送到收件人的邮件服务器的时间（格式为 ISO86 01）。
processingTimeMillis	Amazon SES 接受来自发件人的请求与将邮件传递到收件人邮件服务器之间的时间，以毫秒为单位。
recipients	电子邮件送达通知适用的目标收件人的列表。
smtpResponse	从 Amazon SES 接受电子邮件的远程 ISP 的 SMTP 响应消息。此消息因电子邮件、接收邮件服务器以及接收 ISP 而异。
reportingMTA	发送邮件的 Amazon SES 邮件服务器的主机名。
remoteMtaIp	Amazon SES 将电子邮件送达的 MTA 的 IP 地址。

以下是 delivery 对象的示例。

```
{
   "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 SES 的 Amazon SNS 通知示例

以下各节提供了三种类型的通知的示例：

- 有关退回邮件通知示例，请参阅 Amazon SNS 退回邮件通知示例。
- 有关投诉通知示例，请参阅 Amazon SNS 投诉通知示例。
- 有关送达通知示例，请参阅 Amazon SNS 送达通知示例。

Amazon SNS 退回邮件通知示例

本节包含退回邮件通知的示例，这些通知带有和不带由发送反馈的电子邮件接收方提供的传输状态通知 (DSN)。

带 DSN 的退回邮件通知

下面是一个包含 DSN 和原始电子邮件标头的退回邮件通知的示例。当退回邮件通知未配置为包含原始电子邮件标头时，这些通知内的 `mail` 对象不会包含 `headersTruncated`、`headers` 和 `commonHeaders` 字段。

```
{
    "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"
        }
      }
```

```
    }
```

不带 DSN 的退回邮件通知

下面是一个包含原始电子邮件标头但不包含 DSN 的退回邮件通知的示例。当退回邮件通知未配置为包含原始电子邮件标头时，这些通知内的 `mail` 对象不会包含 `headersTruncated`、`headers` 和 `commonHeaders` 字段。

```
{
   "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 投诉通知示例

本节包含投诉通知的示例，这些通知带有和不带由发送反馈的电子邮件接收方提供的反馈报告。

带反馈报告的投诉通知

下面是一个包含反馈报告和原始电子邮件标头的投诉通知的示例。当投诉通知未配置为包含原始电子邮件标头时，这些通知内的 `mail` 对象不会包含 `headersTruncated`、`headers` 和 `commonHeaders` 字段。

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

不带反馈报告的投诉通知

下面是一个包含原始电子邮件标头但不包含反馈报告的投诉通知的示例。当投诉通知未配置为包含原始电子邮件标头时，这些通知内的 `mail` 对象不会包含 `headersTruncated`、`headers` 和 `commonHeaders` 字段。

```
  {
     "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 送达通知示例

下面是包含原始电子邮件标头的送达通知的示例。当送达通知未配置为包含原始电子邮件标头时，这些通知内的 `mail` 对象不会包含 `headersTruncated`、`headers` 和 `commonHeaders` 字段。

```
   {
      "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"
      }
   }
```

使用 Amazon SES 中的身份授权

身份授权策略通过以下方式定义个人已验证的身份如何使用 Amazon SES：指定允许或拒绝对身份执行哪些 SES API 操作，以及在哪些条件下对身份执行这些操作。

通过使用这些授权策略，您可以随时更改或撤销权限，从而维护您对身份的控制。您甚至可以授权其他用户通过其自有 SES 账户使用您拥有的身份（域或电子邮件地址）。

主题

- Amazon SES 策略剖析
- 在 Amazon SES 中创建身份授权策略
- Amazon SES 中的身份策略示例
- 在 Amazon SES 中管理您的身份授权策略

Amazon SES 策略剖析

策略遵循特定的结构，包含元素，并且必须满足特定的要求。

策略结构

每个授权策略是一个附加到身份的 JSON 文档。每个策略包含以下部分：

- 位于文档顶部的策略级信息。
- 一个或多个单独的语句，每个语句描述一组权限。

以下示例策略授予 AWS 账户 ID 为 123456789012 的权限，这些权限在 “操作” 部分中为已验证的域名 example.com 指定。

JSON

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

您可以在身份策略示例中找到更多授权策略的示例。

策略元素

本部分介绍身份授权策略中包含的元素。首先，我们将介绍策略级元素，然后介绍仅适用于语句的元素。我们接下来将讨论如何在语句中添加条件。

有关元素语法的特定信息，请参阅《IAM 用户指南》中的 IAM 策略语言的语法。

策略级信息

有两种策略级元素：Id 和 Version。下表提供了有关这些元素的信息。

名称	描述	必填	有效值
Id	唯一标识策略。	否	任何字符串
Version	指定策略访问语言的版本。	否	任何字符串。作为最佳实践，我们建议为此字段使用“2012-10-17”值。

特定于策略的语句

身份授权策略需要至少一个语句。每个语句可以包含下表中所述的元素。

名称	描述	必填	有效值
Sid	唯一标识语句。	否	任何字符串。
Effect	指定在评估期间您希望策略语句返回的结果。	是	“Allow”或“Deny”。
Resource	指定将应用策略的身份。 （对于发送授权，这是身份所有者授权委托发件人使用的电子邮件地址或域。）	是	身份的 Amazon 资源名称（ARN）。
Principal	在语句中指定接收权限的 AWS 账户、用户或 AWS 服务。	是	有效的 AWS 账户 ID、用户 ARN 或 AWS 服务。 AWS 账户 IDs 和用户 ARNs 是使用"AWS"（例如，"AWS":

名称	描述	必填	有效值
			["123456789012"] 或"AWS": ["arn:aws:iam::123456789012:root"] ）指定的。 AWS 服务名是使用"Service" （例如"Service": ["cognito-idp.amazonaws.com"] ）指定的。 有关用户格式的示例 ARNs，请参阅AWS 一般参考。

名称	描述	必填	有效值
Action	指定语句所适用的操作。	是	“ses：BatchGetMetricData”、“ses：CancelExportJob”、“ses：”、CreateDeliverabilityTestReport “ses：”、CreateEmailIdentityPolicy “ses：”、CreateExportJob “ses：”、DeleteEmailIdentity “ses：”、DeleteEmailIdentityPolicy “ses：”、GetDomainStatisticsReport “ses：”、GetEmailIdentity “ses：”、GetEmailIdentityPolicies “ses：”、GetExportJob “ses：”、“ses：ListExportJobs”、“ses：ListRecommendations”、“ses：PutEmailIdentityConfigurationSetAttributes”、“ses：”、PutEmailIdentityDkimAttributes “ses：”、PutEmailIdentityDkimSigningAttributes “ses：PutEmailIdentityFeedbackAttribu

名称	描述	必填	有效值
			tes”、“ses：PutEmail IdentityMailFromAt tributes”、“ses：Tag Resource”，“ses：UntagResourc e”，“ses：U pdateEmailIdentity Policy” （发送授权操作：“ses：”、SendEmai l “ses：”、Se ndRawEmail “ses：”、“ses：SendTemplatedEm ail”、“ses：SendBulk TemplatedEmail”） 您可以指定一个或多个这些操作。
`Condition`	指定任何有关权限的限制条件或详细信息。	否	请参阅下表中有关条件的信息。

Conditions

条件是语句中有关权限的任何限制。语句中指定条件的部分可能为各部分中最为详尽的部分。密钥是作为访问限制基础的具体特征，例如，请求的日期和时间。

您需使用条件和密钥一起明确说明限制。例如，如果需要限制委托发件人，使其不能在 2019 年 7 月 30 日后代表您对 Amazon SES 发出请求，则可以使用名为 `DateLessThan` 的条件。您使用的密钥名为 `aws:CurrentTime`，并将其值设置为 `2019-07-30T00:00:00Z`。

SES 仅实现以下 AWS范围的策略密钥：

- aws:CurrentTime
- aws:EpochTime
- aws:SecureTransport
- aws:SourceIp
- aws:SourceVpc
- aws:SourceVpce
- aws:UserAgent
- aws:VpcSourceIp

有关这些键的更多信息，请参阅 IAM 用户指南。

策略要求

策略必须满足下列所有要求：

- 每个策略必须包括至少一个语句。
- 每个策略必须包括至少一个有效主体。
- 每个策略必须指定一个资源，该资源必须是附加策略的身份的 ARN。
- 身份所有者可以为每个唯一的身份关联最多 20 个策略。
- 策略不得超过 4 千字节 (KB)。
- 策略名称不得超过 64 个字符。此外，它们只能包含字母数字字符、连字符和下划线。

在 Amazon SES 中创建身份授权策略

身份授权策略由声明组成，这些声明指定允许或拒绝对身份执行哪些 API 操作，以及在哪些条件下允许或拒绝执行这些 API 操作。

要授权使用您拥有的 Amazon SES 域或电子邮件地址身份，您必须创建授权策略，然后将该策略附加到身份。一个身份可以有零个、一个或多个策略。但是，一个策略只能与一个身份关联。

有关可在身份授权策略中使用的 API 操作的列表，请参阅the section called “特定于策略的语句”表中的 Action（操作）行。

您可以通过下列方法来创建身份授权策略：

- 通过使用策略生成器 - 您可以在 SES 控制台中使用策略生成器创建简单的策略。除了允许或拒绝对 SES API 操作的权限外，您还可以使用条件限制操作。您还可以使用策略生成器快速创建策略的基本结构，然后通过编辑策略对其进行自定义。
- 通过创建自定义策略 — 如果您想包含更高级的条件或使用 AWS 服务作为委托人，则可以使用 SES 控制台或 SES API 创建自定义策略并将其附加到身份。

主题

使用策略生成器

您可以使用策略生成器创建简单授权策略，步骤如下。

使用策略生成器创建策略

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择已验证身份。
3. 在 Verified identities（已验证身份）屏幕上的 Identities（身份）容器中，选择您希望为之创建授权策略的已验证身份。
4. 在上一步中选择的已验证身份的详细信息屏幕中，选择 Authorization（授权）选项卡。
5. 在 Authorization policies（授权策略）窗格中，选择 Create policy（创建策略），并从下拉列表中选择 Use policy generator（使用策略生成器）。
6. 在 Create statement（创建语句）窗格中，选择 Effect（效果）字段中的 Allow（允许）。（如果要创建策略以限制此身份，请改为选择 Deny（拒绝）。）
7. 在委托人字段中，输入要获得您要为该AWS 账户 身份授权的权限的 ID、IAM 用户 ARN AWS 或服务，然后选择添加。（如果您希望授权多个委托人，请为每个委托人重复此步骤。）
8. 在 Actions（操作）字段中，选中要为委托人授权的每种操作对应的复选框。
9. （可选）如果希望向权限添加限定语句，请展开 Specify conditions（指定条件）。

    a. 从 Operator（运算符）下拉菜单中选择运算符。

    b. 从 Key（类型）下拉菜单中选择类型。

    c. 根据您选择的键类型，在 Value（值）字段中输入其值。（如果想添加更多条件，请选择 Add new condition（添加新条件），然后为每个额外的步骤重复此步骤。）

10. 选择 Save statement（保存语句）。
11. （可选）如果希望向策略中添加更多语句，请展开 Create another statement（创建另一个语句），并重复步骤 6 - 10。
12. 选择 Next（下一步），则 Customize policy（自定义策略）屏幕中，Edit policy details（编辑策略详细信息）容器将具有您可以自己更改或自定义策略的 Name（名称）和 Policy document（策略文档）的字段。
13. 选择 Next（下一步），在 Review and apply（查看并应用）屏幕上，Overview（概述）容器将显示您要授权的已验证身份以及此策略的名称。在 Policy document（策略文档）窗格将是您刚刚写的实际政策以及您添加的任何条件——查看策略，如果看起来正确，请选择 Apply policy（申请策略）。（如果您需要更改或更正某些内容，请选择 Previous（上一页）并在 Edit policy details（编辑策略详细信息）容器中操作。）

创建自定义策略

如果要创建自定义策略并将其附加到身份，您有以下选择：

- 使用 Amazon SES API – 在文本编辑器中创建策略，然后使用 Amazon Simple Email Service API 参考中所述的 `PutIdentityPolicy` API 将策略附加到身份。
- 使用 Amazon SES 控制台 – 在文本编辑器中创建策略，并在 Amazon SES 控制台中将其粘贴到自定义策略编辑器，将策略附加到身份。以下步骤介绍这种方法。

使用自定义策略编辑器创建自定义策略

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择已验证身份。
3. 在 Verified identities（已验证身份）屏幕上的 Identities（身份）容器中，选择您希望为之创建授权策略的已验证身份。
4. 在上一步中选择的已验证身份的详细信息屏幕中，选择 Authorization（授权）选项卡。
5. 在 Authorization policies（授权策略）窗格中，选择 Create policy（创建策略），并从下拉列表中选择 Create custom policy（创建自定义策略）。
6. 在 Policy document（策略文档）窗格中，键入或粘贴 JSON 格式的策略文本。您还可以使用策略生成器快速创建策略的基本结构，然后在此处对其进行自定义。

7. 选择应用策略。（如果您需要修改自定义策略，只需在 Authorization（授权）选项卡，选择 Edit（编辑），然后在 Policy document（策略文档）窗格中更改，然后是窗格 Save changes（保存更改）。

Amazon SES 中的身份策略示例

您可以使用身份授权指定允许或拒绝对身份执行 API 操作的精细条件。

以下示例介绍如何编写策略来控制不同方面的 API 操作：

- 指定主体
- 限制操作
- 使用多个语句

指定主体

委托人（即您向其授予权限的实体）可以是 AWS 账户、 AWS Identity and Access Management (IAM) 用户或属于同一账户的 AWS 服务。

以下示例显示了一个简单的策略，该策略允许 AWS ID 123456789012 控制同样归123456789012 拥有的经过验证的身份 e xample.com。 AWS 账户

JSON

```
{
  "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"
```

```
      ]
    }
  ]
}
```

以下示例策略向两个用户授予控制已验证的身份 example.com 的权限。用户通过其 Amazon 资源名称（ARN）指定。

JSON

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

限制操作

根据您要授权的控制级别，可以在身份授权策略中指定多种操作：

```
"BatchGetMetricData",
"ListRecommendations",
"CreateDeliverabilityTestReport",
"CreateEmailIdentityPolicy",
```

```
"DeleteEmailIdentity",
"DeleteEmailIdentityPolicy",
"GetDomainStatisticsReport",
"GetEmailIdentity",
"GetEmailIdentityPolicies",
"PutEmailIdentityConfigurationSetAttributes",
"PutEmailIdentityDkimAttributes",
"PutEmailIdentityDkimSigningAttributes",
"PutEmailIdentityFeedbackAttributes",
"PutEmailIdentityMailFromAttributes",
"TagResource",
"UntagResource",
"UpdateEmailIdentityPolicy"
```

身份授权策略还使您能够将主体限制为只能执行其中一项操作。

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

使用多个语句

身份授权策略可以包含多个语句。以下示例策略包含两个语句。第一个语句拒绝两个用户在同一个账户 123456789012 内通过 sender@example.com 访问 getemailidentity。第二个语句拒绝主体 Jack 在同一个账户 123456789012 内访问 UpdateEmailIdentityPolicy。

JSON

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

在 Amazon SES 中管理您的身份授权策略

除了创建策略和将策略附加到身份之外，您还可以按照以下各部分的说明编辑、删除、列出和检索身份的策略。

使用 Amazon SES 控制台管理策略

管理 Amazon SES 策略需要使用 Amazon SES 控制台查看、编辑或删除附加到身份的策略。

使用 Amazon SES 控制台管理策略

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，选择已验证身份。
3. 在身份列表中，选择您要管理的身份。
4. 在身份详细信息页面上，导航到 Authorization（授权）选项卡。在这里，您可以找到附上此身份的所有策略的列表。
5. 通过选中要管理的策略复选框来选择该策略。
6. 根据所需的管理任务，选择相应的按钮，如下所示：

   a. 要查看策略，请选择 View policy（查看策略）。如果需要其副本，请选择 Copy（复制）按钮，它将被复制到剪贴板。

   b. 要编辑策略，请选择 Edit（编辑）。在 Policy document（策略文档）中，编辑策略，然后选择 Save changes（保存更改）。

   > **Note**
   >
   > 要撤销权限，您可以编辑策略或删除它。

   c. 要移除策略，请选择 Delete（删除）。

   > **Important**
   >
   > 删除策略是永久性操作。我们建议您通过复制策略并粘贴到文本文件中来备份策略，然后再将其删除。

使用 Amazon SES API 管理策略

管理 Amazon SES 策略需要使用 Amazon SES API 查看、编辑或删除附加到身份的策略。

使用 Amazon SES API 列出和查看策略

- 您可以使用 ListIdentityPolicies API 操作列出附加到身份的策略。您也可以使用 GetIdentityPoliciesAPI 操作自行检索策略。

要使用 Amazon SES API 来编辑策略

- 您可以使用 PutIdentityPolicy API 操作编辑附加到身份的策略。

要使用 Amazon SES API 删除策略

- 您可以使用 DeleteIdentityPolicy API 操作删除附加到身份的策略。

使用 Amazon SES 的发送授权

您可以配置 Amazon SES 来授权其他用户使用其自有 Amazon SES 账户从您拥有的身份（域或电子邮件地址）发送电子邮件。发送授权功能可用于维护您对身份的控制，以便随时更改或撤销权限。例如，如果您是业务负责人，您可以使用发送授权来允许第三方 (例如电子邮件营销公司) 从您的域发送电子邮件。

本章介绍了发送授权的细节，该授权取代了传统的跨账户通知功能。您应该首先了解使用授权策略进行基于身份的授权的基础知识，如使用 Amazon SES 中的身份授权中所述，其中介绍了授权策略剖析以及如何管理策略等重要主题。

跨账户通知旧式支持

与委托发件人发送的电子邮件（该电子邮件由身份拥有者授权委托发件人从其经验证的身份之一发送）相关联的退回邮件、投诉和送达的反馈通知，传统上是使用跨账户通知配置的，其中，委托发件人将主题与他们不拥有的身份（即跨账户）相关联。但是，跨账户通知已被使用配置集以及与委托发送相关联的经验证身份所取代，在这种情况下，委托发件人已获得身份拥有者的授权，可以使用其经验证的身份之一发送电子邮件。这种新方法允许灵活地通过以下两种结构配置退回邮件、投诉、送达和其它事件通知，具体取决于您是委托发件人还是经验证身份的拥有者：

- Configuration sets（配置集）— 委托发件人可以在自己的配置集中设置事件发布，他可以在使用他不拥有但已获得身份所有者通过授权策略授权发送电子邮件时指定该配置集。事件发布允许向亚马逊、亚马逊 Data Firehose、Amazon Pinpoint 和亚马逊 SNS 发布退信 CloudWatch、投诉、送达和其他事件通知。请参阅 创建事件目标。

- Verified identities（验证身份）— 除了让身份所有者授权委托发件人使用其经过验证的身份之一发送电子邮件之外，他还可以应委托发件人的请求，对共享身份配置反馈通知，以使用委托发件人拥有的 SNS 主题。只有委托发件人才会收到这些通知，因为他们拥有 SNS 主题。请参阅第 14 步以了解如何在授权策略程序中配置“您不拥有的 SNS 主题”。

> **Note**
>
> 为了兼容性，您账户中当前使用的旧版跨账户通知支持跨账户通知。此支持仅限于能够修改和使用您在 Amazon SES 经典控制台中创建的任何当前跨账户；但是，您无法再创建新的跨账户通知。要在 Amazon SES 新控制台中创建新控制台，请使用新的委托发送方法与配置集一起使用事件发布，或使用经过验证的身份使用自己的 SNS 主题进行配置。

主题

Amazon SES 发送授权概览

本主题概述了发送授权流程，然后介绍了 Amazon SES 的电子邮件发送功能（例如发送配额和通知）如何与发送授权结合使用。

本部分使用了以下术语：

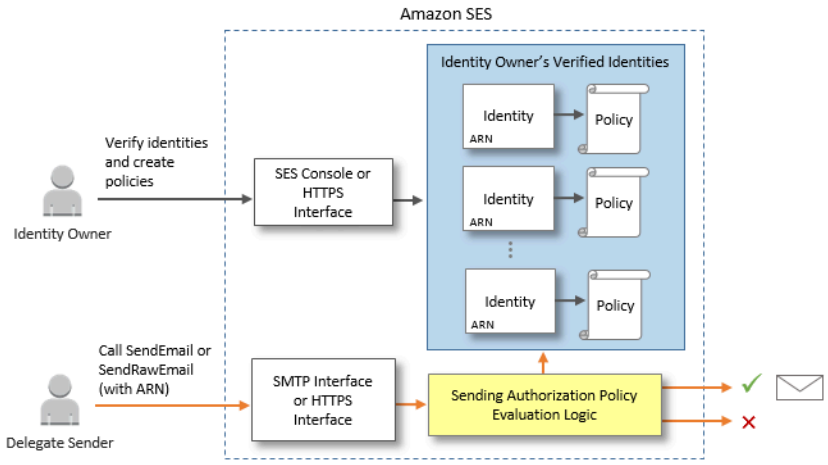
- 身份 – Amazon SES 用户用来发送电子邮件的地址或域。
- 身份拥有者 – 已按照已验证的身份中所述过程验证了电子邮件地址或域所有权的 Amazon SES 用户。
- 委托发件人 — 通过授权策略授权代表身份所有者发送电子邮件的 AWS 账户、 AWS Identity and Access Management (IAM) 用户或 AWS 服务。
- 发送授权策略 - 一种附加到身份的文档，用于指定哪些人、在何种条件下可以使用该身份发送。
- Amazon 资源名称 (ARN) — 一种在所有 AWS 服务中唯一标识 AWS 资源的标准化方法。对于发送授权而言，资源是身份拥有者希望委托发件人使用的身份。ARN 的某个示例为 arn:aws:ses:us-east-1:123456789012:identity/example.com。

发送授权的流程

发送授权以发送授权策略为基础。如果要允许委托发件人代表您发送邮件，您可使用 Amazon SES 控制台或 Amazon SES API 创建发送授权策略，并将该策略与您的身份相关联。当委托发件人尝试通过 Amazon SES 代表您发送电子邮件时，委托发件人将在请求中或在电子邮件标头中传递您的身份的 ARN。

当 Amazon SES 收到发送电子邮件的请求时，它会检查您的身份策略（如果有）以确定您是否已授权该委托发件人代表该身份发送邮件。如果委托发件人已获得授权，Amazon SES 会接受电子邮件；否则，Amazon SES 将返回错误消息。

下图显示了发送授权概念之间的概述关系：



发送授权过程包括以下步骤：

1. 身份所有者选择经过验证的身份供委托发件人使用。（如果您没有验证身份，请参阅已验证的身份）。

   > **Note**
   >
   > 您为委托发件人选择的经过验证的身份不能为其分配原定设置的配置集。

2. 委托发送者让身份所有者知道他们要使用哪个 AWS 账户 ID 或 IAM 用户 ARN 进行发送。
3. 如果身份拥有者同意允许委托发送者从身份拥有者的一个账户发送，则身份拥有者创建发送授权策略，并使用 Amazon SES 控制台或 Amazon SES API 将该策略附加到所选身份。
4. 身份拥有者将身份的 ARN 提供给委托发件人，以便委托发件人在发送电子邮件时向 Amazon SES 提供 ARN。

5. 代理发送者可以通过在代理发送期间指定的配置集中启用的事件发布来设置退回和投诉通知。身份所有者还可以为退回邮件和投诉事件设置电子邮件反馈通知，以便发送到委托发件人的 Amazon SNS 主题。

> **Note**
>
> 如果身份拥有者禁用发送事件通知，则委托发件人必须设置事件发布，以将退回邮件和投诉事件发送到 Amazon SNS 主题或 Firehose 流。发件人还必须对他们发送的每封电子邮件应用包含事件发布规则的配置集。如果身份拥有者和委托发件人都未设置针对退回邮件和投诉事件发送通知的方法，则 Amazon SES 将自动通过电子邮件将事件通知发送到电子邮件的 Return-Path (退回路径) 字段中的地址（或者 Source (来源) 字段中的地址，如果未指定退回路径地址），即使身份拥有者禁用了电子邮件反馈转发也是如此。

6. 委托发件人通过在请求中或在电子邮件标头中传递身份拥有者的身份的 ARN，尝试代表身份拥有者经由 Amazon SES 发送电子邮件。委托发件人可以使用 Amazon SES SMTP 接口或 Amazon SES API 发送电子邮件。收到请求后，Amazon SES 会检查附加到身份的所有策略，如果委托发件人有使用指定“发件人”地址和“退回路径”地址的授权，则接受电子邮件；否则，Amazon SES 会返回错误，并且不接受该邮件。

> **Important**
>
> 必须先将委托发件人的 AWS 帐户从沙箱中删除，然后才能将其用于向未经验证的地址发送电子邮件。

7. 如果身份所有者需要取消对委托发件人的授权，则身份所有者将编辑发送授权策略或完全删除此策略。身份拥有者可通过使用 Amazon SES 控制台或 Amazon SES API 来执行上述任一操作。

有关身份拥有者或委托发件人如何执行这些任务的信息，请分别参阅身份拥有者任务或委托发件人任务。

电子邮件发送功能的属性

对于各种 Amazon SES 电子邮件发送功能（例如每日发送配额、退回邮件和投诉、DKIM 签名、反馈转发等），了解委托发件人和身份拥有者的角色很重要。属性如下：

- 发送配额 – 使用身份拥有者的身份发送的电子邮件会计入委托发件人的配额。
- 退回邮件和投诉 – 退回邮件和投诉事件记录在委托发件人的 Amazon SES 账户中，因此可能影响委托发件人的声誉。

- DKIM 签名 – 如果身份拥有者已为某个身份启用 Easy DKIM 签名，发自该身份的所有电子邮件将进行 DKIM 签名，包括由委托发件人发送的电子邮件。只有身份所有者能够控制电子邮件是否进行 DKIM 签名。
- 通知 – 身份拥有者和委托发件人都可针对退回邮件和投诉设置通知。电子邮件身份所有者还可以启用电子邮件反馈转发。有关设置通知的信息，请参阅监控您的 Amazon SES 发送活动。
- 验证 – 身份拥有者有责任执行已验证的身份中的以下过程，以验证其拥有授权委托发件人使用的电子邮件地址和域。委托发件人不需要验证专用于发送授权的任何电子邮件地址或域。

  > **Important**
  >
  > 必须先将委托发件人的 AWS 帐户从沙箱中删除，然后才能将其用于向未经验证的地址发送电子邮件。

- AWS 区域-委托发件人必须从验证身份所有者身份的 AWS 地区发送电子邮件。给委托发件人授予权限的发送授权策略必须附加到该区域中的身份。
- 计费 – 从委托发件人发送的所有邮件，包括委托发件人使用身份所有者的地址发送的电子邮件，将计入委托发件人的账户中。

Amazon SES 发送授权的身份拥有者任务

本部分介绍身份所有者在配置发送授权时必须执行的步骤。

主题

为 Amazon SES 发送授权验证身份

配置发送授权的第一步是证明您拥有委托发件人将用于发送电子邮件的电子邮件地址或域。验证过程如已验证的身份中所述。

您可以通过在的 “已验证身份” 部分或使用 `GetIdentityVerificationAttributes` API 操作检查其状态来确认电子邮件地址https://console.aws.amazon.com/ses/或域名是否已通过验证。

您必须先提交一份请求，要求将您的账户移出 Amazon SES 沙盒，之后您或委托发件人才能向未经验证的电子邮件地址发送电子邮件。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。

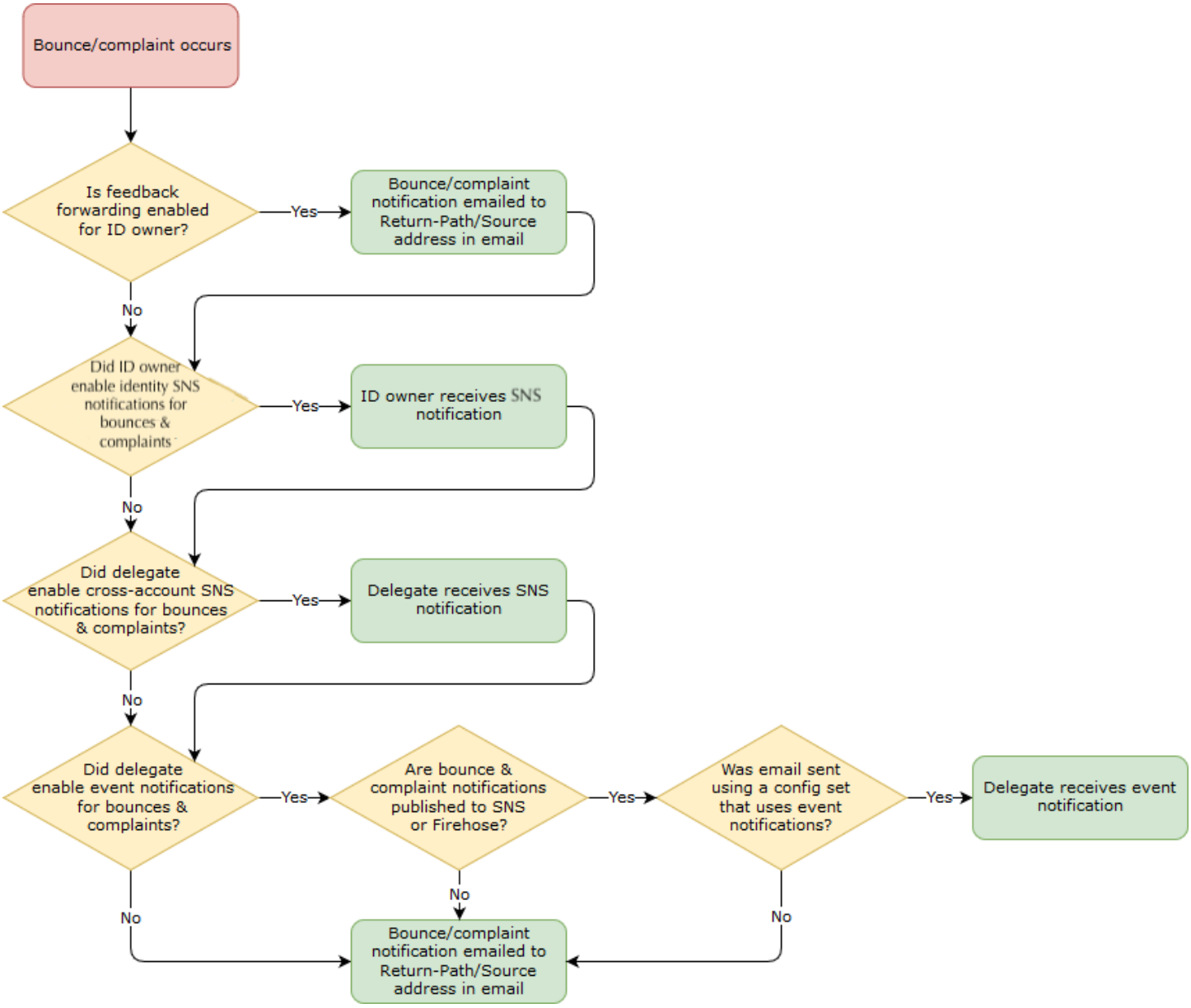
> **Important**
>
> - 必须先将 AWS 账户 委托发件人从沙箱中移除，然后才能将其用于向未经验证的地址发送电子邮件或从未经验证的地址发送电子邮件。
> - 如果您的账户位于沙盒中，则无法向账户中未经验证的电子邮件地址发送电子邮件，即使这些域或电子邮件地址已在身份账户中经过验证也是如此。

设置 Amazon SES 发送授权的身份拥有者通知

如果您授权委托发件人代表您发送电子邮件，则 Amazon SES 会将这些电子邮件产生的退回邮件或投诉计入委托发件人的而不是您的退回邮件和投诉限制。但是，如果您的 IP 地址因代理发件人发送的邮件而出现在第三方反垃圾邮件、基于 DNS 的黑洞列表 (DNSBLs) 中，则您的身份信誉可能会受到损害。为此，如果您是身份所有者，应为您的身份（包括为委托发送授权的身份）设置电子邮件反馈转发。有关更多信息，请参阅 通过电子邮件接收 Amazon SES 通知。

委托发件人可以且应当针对您已授权其使用的身份设置其自己的退回邮件和投诉通知。他们还可以设置事件发布来将退回邮件和投诉事件发布到 Amazon SNS 主题或 Firehose 流中。

如果身份拥有者和委托发件人都未设置针对退回邮件和投诉事件发送通知的方法，或者发件人未应用使用事件发布规则的配置集，则 Amazon SES 将自动通过电子邮件将事件通知发送到电子邮件的 Return-Path (退回路径) 字段中的地址（或者 Source (来源) 字段中的地址，如果未指定退回路径地址），即使禁用了电子邮件反馈转发也是如此。下图阐述了此过程。



获取委托发件人的信息，用于 Amazon SES 发送授权

您的发送授权策略必须指定至少一个主体，该主体是您授予访问权限的代理发件人的实体，以便他们可以代表您的一个已验证身份发送。对于 Amazon SES 发送授权策略，委托人可以是您的委托发件人 AWS 账户或 AWS Identity and Access Management (IAM) 用户 ARN，也可以是 AWS 服务。

一个简单的方法是，主体（代理发件人）是被授权人，而您（身份所有者）是授权策略中的授权人，您授予他们允许从您拥有的资源（已验证身份）发送电子邮件、原始电子邮件、模板电子邮件或批量模板电子邮件的任意组合的权限。

如果需要进行精细控制，请让委托发件人设置一个 IAM 用户，从而只允许一个委托发件人代表您发送电子邮件，而不是委托发件人的 AWS 账户中的任何用户。委托发件人可在《IAM 用户指南》的在您的 AWS 账户中创建 IAM 用户中找到有关设置 IAM 用户的信息。

向您的委托发件人询问 AWS 账户 ID 或 IAM 用户的亚马逊资源名称 (ARN)，以便您可以将其包含在发送授权策略中。您可以推荐委托发件人参考向身份拥有者提供信息中提供的说明，以便查找这些信息。如果委托发送者是 AWS 服务，请参阅该服务的文档以确定服务名称。

以下示例策略说明了身份所有者为授权委托发件人从身份所有者的资源发送而创建的策略中所需的基本元素。身份所有者将进入已验证身份工作流程，然后在授权下，使用策略生成器以最简单的形式创建以下基本策略，允许委托发件人代表身份所有者拥有的资源发送：

JSON

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

对于上述政策，以下图例解释了关键元素以及谁拥有这些要素：

- Principal（主体）— 此字段使用委托发件人的 IAM 用户 ARN 填充。
- Action（操作）— 此字段填充了两个 SES 操作 (`SendEmail` & `SendRawEmail`) 身份所有者允许委托发件人使用身份所有者的资源执行操作。

- Resource（资源）— 此字段填充了身份所有者已验证的资源，他们授权委托发件人发送该资源。

在 Amazon SES 中创建发送授权策略

与在 Amazon SES 中创建任何授权策略类似，如创建身份授权策略中所述，要授权某个委托发件人使用您拥有的电子邮件地址或域（身份）发送电子邮件，您可以创建指定了 SES 发送 API 操作的策略，然后将该策略附加到身份。

有关可在发送授权策略中指定的 API 操作列表，请参阅the section called “特定于策略的语句”表中的操作行。

您可以使用策略生成器或通过创建自定义策略来创建发送授权策略。两种方法都提供了创建发送授权策略的特定过程。

> **Note**
>
> - 您附加到电子邮件地址标识的发送授权策略优先于附加到其相应域标识的策略。例如，如果您为 example.com 创建一个禁止委托发件人的策略，并且为 sender@example.com 创建一个允许委托发件人的策略，那么委托发件人可以从 sender@example.com 发送电子邮件，但是不能从 example.com 域上的任何其他地址发送电子邮件。
> - 如果您为example.com 创建一个允许委托发件人的策略，并且为sender@example.com 创建一个禁止委托发件人的策略，则委托发件人可以从 example.com 域上的任何地址发送电子邮件，但 sender@example.com 除外。
> - 如果您不熟悉 SES 授权策略的结构，请参阅策略剖析。
> - 作为全球终端节点功能的一部分，如果您正在授权的身份在次要区域中复制，则需要在主区域和次要区域中针对该身份创建发送授权策略，以便委托发送者有权使用此身份在两个区域进行发送。

使用策略生成器创建发送授权策略

您可以使用策略生成器创建发送授权策略，步骤如下。

使用策略生成器创建发送授权策略

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择身份。

3. 在 Verified identities（已验证身份）屏幕上的 Identities（身份）容器中，选择您希望授权代理发件人代表您发送的已验证身份。
4. 选择已验证身份的授权选项卡。
5. 在 Authorization policies（授权策略）窗格中，选择 Create policy（创建策略），并从下拉列表中选择 Use policy generator（使用策略生成器）。
6. 在 Create statement（创建语句）窗格中，选择 Effect（效果）字段中的 Allow（允许）。（如果要创建策略以限制委派发件人，请改为选择 Deny拒绝。）
7. 在 Principals（主体）字段中，输入您的代理发件人与您共享的 AWS 账户 ID 或 IAM 用户 ARN，以授权他们代表您的账户为此身份发送电子邮件，然后选择 Add（添加）。（如果您希望授权多个委托发件人，请为每个代表发件人重复此步骤。）
8. 在 Actions（操作）字段中，选中要为委托发件人授权的每种发送类型的复选框。
9. （可选）如果希望向代理发件人权限添加限定语句，请展开 Specify conditions（指定条件）。
   a. 从 Operator（运算符）下拉菜单中选择运算符。
   b. 从 Key（类型）下拉菜单中选择类型。
   c. 根据您选择的键类型，在 Value（值）字段中输入其值。（如果想添加更多条件，请选择 Add new condition（添加新条件），然后为每个额外的步骤重复此步骤。）
10. 选择 Save statement（保存语句）。
11. （可选）如果希望向策略中添加更多语句，请展开 Create another statement（创建另一个语句），并重复步骤 6 - 10。
12. 选择 Next（下一步），则 Customize policy（自定义策略）屏幕中，Edit policy details（编辑策略详细信息）容器将具有您可以自己更改或自定义策略的 Name（名称）和 Policy document（策略文档）的字段。
13. 选择 Next（下一步），在 Review and apply（查看并应用）屏幕上，Overview（概述）容器将显示您为代理发件人授权的已验证身份以及此策略的名称。在 Policy document（策略文档）窗格将是您刚刚写的实际政策以及您添加的任何条件——查看策略，如果看起来正确，请选择 Apply policy（申请策略）。（如果您需要更改或更正某些内容，请选择 Previous（上一页）并在 Edit policy details（编辑策略详细信息）容器中操作。）您刚刚创建的策略将允许您的代理发件人代表您发送邮件。
14. （可选）如果您的委托发件人还希望使用他们拥有的 SNS 主题、在他们收到退回或投诉时接收反馈通知，或者在发送电子邮件时，您需要在此经过验证的身份中配置他们的 SNS 主题。（您的委托发件人需要与您分享他们的 SNS 主题 ARN。）选择 Notifications（通知）选项卡，然后选择 Feedback notifications（反馈通知）容器中的 Edit（编辑）。

a. 在 Configure SNS topics（配置 SNS 主题）窗格的任何反馈字段(“退回”、“投诉”或“发送”)中，选择 SNS topic you don’t own（您不拥有的 SNS 主题），并输入您的代理发件人拥有并与您共享的 SNS topic ARN（SNS 主题 ARN）。（只有您的委托发件人才会收到这些通知，因为他们拥有 SNS 主题——您作为身份所有者不会。）

b. （可选）如果您希望主题通知包括原始电子邮件中的标头，请选中 Include original email headers（包括原始电子邮件标头）框直接位于每种反馈类型的 SNS 主题名称下方。此选项仅在您已将 Amazon SNS 主题分配到关联的通知类型时可用。有关原始电子邮件标头的内容的信息，请参阅 通知内容中的 `mail` 对象。

c. 选择 Save changes（保存更改）。您对通知设置所做的更改可能需要几分钟才能生效。

d. （可选）由于您的委托发件人将收到有关退回和投诉的 Amazon SNS 主题通知，因此如果您不想收到有关此身份发送的反馈，则可以完全禁用电子邮件通知。要禁用退回和投诉的电子邮件反馈，请在 Notifications（通知）选项卡下的 Email Feedback Forwarding（电子邮件反馈转发）容器中，选择 Edit（编辑），取消选中 Enabled（已启用）框，然后选择 Save changes（保存更改）。现在，交付状态通知只会发送到委托发件人所拥有的 SNS 主题。

创建自定义发送授权策略

如果要创建自定义发送授权策略并将其附加到身份，您有以下选择：

- 使用 Amazon SES API – 在文本编辑器中创建策略，然后使用 Amazon Simple Email Service API 参考中所述的 `PutIdentityPolicy` API 将策略附加到身份。
- 使用 Amazon SES 控制台 – 在文本编辑器中创建策略，并在 Amazon SES 控制台中将其粘贴到自定义策略编辑器，将策略附加到身份。以下步骤介绍这种方法。

使用自定义策略编辑器创建自定义发送授权策略

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择身份。
3. 在 Verified identities（已验证身份）屏幕上的 Identities（身份）容器中，选择您希望授权代理发件人代表您发送的已验证身份。
4. 在上一步中选择的已验证身份的详细信息屏幕中，选择 Authorization（授权）选项卡。
5. 在 Authorization policies（授权策略）窗格中，选择 Create policy（创建策略），并从下拉列表中选择 Create custom policy（创建自定义策略）。

6. 在 Policy document（策略文档）窗格中，键入或粘贴 JSON 格式的策略文本。您还可以使用策略生成器快速创建策略的基本结构，然后在此处对其进行自定义。
7. 选择应用策略。（如果您需要修改自定义策略，只需在 Authorization（授权）选项卡，选择 Edit（编辑），然后在 Policy document（策略文档）窗格中更改，然后是窗格 Save changes（保存更改）。
8. （可选）如果您的委托发件人还希望使用他们拥有的 SNS 主题、在他们收到退回或投诉时接收反馈通知，或者在发送电子邮件时，您需要在此经过验证的身份中配置他们的 SNS 主题。（您的委托发件人需要与您分享他们的 SNS 主题 ARN。）选择 Notifications（通知）选项卡，然后选择 Feedback notifications（反馈通知）容器中的 Edit（编辑）。

   a. 在 Configure SNS topics（配置 SNS 主题）窗格的任何反馈字段(“退回”、“投诉”或“发送”)中，选择 SNS topic you don’t own（您不拥有的 SNS 主题），并输入您的代理发件人拥有并与您共享的 SNS topic ARN（SNS 主题 ARN）。（只有您的委托发件人才会收到这些通知，因为他们拥有 SNS 主题——您作为身份所有者不会。）
   b. （可选）如果您希望主题通知包括原始电子邮件中的标头，请选中 Include original email headers（包括原始电子邮件标头）框直接位于每种反馈类型的 SNS 主题名称下方。此选项仅在您已将 Amazon SNS 主题分配到关联的通知类型时可用。有关原始电子邮件标头的内容的信息，请参阅 通知内容中的 `mail` 对象。
   c. 选择 Save changes（保存更改）。您对通知设置所做的更改可能需要几分钟才能生效。
   d. （可选）由于您的委托发件人将收到有关退回和投诉的 Amazon SNS 主题通知，因此如果您不想收到有关此身份发送的反馈，则可以完全禁用电子邮件通知。要禁用退回和投诉的电子邮件反馈，请在 Notifications（通知）选项卡下的 Email Feedback Forwarding（电子邮件反馈转发）容器中，选择 Edit（编辑），取消选中 Enabled（已启用）框，然后选择 Save changes（保存更改）。现在，交付状态通知只会发送到委托发件人所拥有的 SNS 主题。

发送策略示例

您可以使用发送授权指定允许委托发件人代表您发送电子邮件的精细条件。

以下条件和示例介绍如何编写策略来控制发送的不同方面：

- 发送授权的特定条件
- 指定委托发件人
- 限制“发件人”地址
- 限制主体发送电子邮件的时间
- 限制电子邮件发送操作

- 限制电子邮件发件人的显示名称
- 使用多个语句

发送授权的特定条件

条件是语句中有关权限的任何限制。语句中指定条件的部分可能为各部分中最为详尽的部分。密钥是作为访问限制基础的具体特征，例如，请求的日期和时间。

您需使用条件和密钥一起明确说明限制。例如，如果需要限制委托发件人，使其不能在 2019 年 7 月 30 日后代表您对 Amazon SES 发出请求，则可以使用名为 `DateLessThan` 的条件。您使用的密钥名为 `aws:CurrentTime`，并将其值设置为 `2019-07-30T00:00:00Z`。

您可以使用 IAM 用户指南中可用密钥中列出的任何一个 AWS范围内的密钥，也可以使用以下特定于 SES 的密钥之一，这些密钥在发送授权策略时非常有用：

条件键	描述
`ses:Recipients`	限制收件人地址，这包括“收件人”、“抄送”和“密件抄送”地址。
`ses:FromAddress`	限制“发件人”地址。
`ses:FromDisplayName`	限制用作“发件人”显示名称的字符串的内容 (有时称为“易记发件人”)。例如，“John Doe <johndoe@example.com>”的显示名称为 John Doe。
`ses:FeedbackAddress`	限制“退回路径”地址，这是供退回邮件和投诉通过电子邮件反馈转发发送给您的地址。有关电子邮件反馈转发的信息，请参阅通过电子邮件接收 Amazon SES 通知。

您可以将 `StringEquals` 和 `StringLike` 条件与 Amazon SES 密钥一起使用。这些条件用于区分大小写的字符串匹配。对于 `StringLike`，值可以在字符串中的任何位置包括多字符匹配的通配符 (*) 或单字符匹配的通配符 (?)。例如，下列条件指定委托发件人只能从以 invoicing 开头并以 @example.com 结尾的“发件人”地址发送电子邮件：

```
"Condition": {
```

```
    "StringLike": {
      "ses:FromAddress": "invoicing*@example.com"
    }
}
```

您还可以使用 StringNotLike 条件来防止委托发件人从某些电子邮件地址发送电子邮件。例如，您可以通过在您的策略语句中包含以下条件，禁止从 admin@example.com 以及类似地址（例如 "admin"@example.com、admin+1@example.com 或 sender@admin.example.com）发送邮件：

```
"Condition": {
    "StringNotLike": {
      "ses:FromAddress": "*admin*example.com"
    }
 }
```

有关如何指定条件的更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。

指定委托发件人

委托人（即您向其授予权限的实体）可以是 AWS 账户、 AWS Identity and Access Management (IAM) 用户或 AWS 服务。

以下示例显示了一个简单的策略，该策略允许 AWS ID 123456789012 从经过验证的身份 example.com（归88888888888888所有）发送电子邮件。 AWS 账户 本政策中的*Condition*声明仅允许委托人（即 AWS ID 123456789012）从地址营销+发送电子邮件。 * @example .com，其中 * 是发件人想要在 m arketing + 之后添加的任何字符串。 。

JSON

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

以下示例策略向两个 IAM 用户授予从身份 example.com 发送电子邮件的权限。IAM 用户通过其 Amazon 资源名称（ARN）来指定。

JSON

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

以下示例策略向 Amazon Cognito 授予从身份 example.com 发送电子邮件的权限。

JSON

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

以下示例策略向 AWS 组织中的所有账户授予从身份 example.com 发送电子邮件的权限。 AWS 组织是使用 PrincipalOrgID 全局条件键指定的。

JSON

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

限制“发件人”地址

如果您使用的是已验证的域，则可能需要创建一个策略，以便只允许委托发件人从指定电子邮件地址发送电子邮件。要限制 “发件人” 地址，您需要在密钥上设置一个名为 ses: 的条件FromAddress。以下政策允许从身份 e xample.com 发送 AWS 账户 ID 123456789012，但只能从电子邮件地址 sender@example.com 发送。

JSON

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

限制主体发送电子邮件的时间

您还可以配置发件人授权策略，限制委托发件人只能在一天的特定时间范围或在特定的日期范围内发送电子邮件。例如，如果您计划在 2021 年 9 月发送电子邮件广告，则可以使用以下策略限制主体只能在此月发送电子邮件。

JSON

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

限制电子邮件发送操作

发件人可使用两种操作通过 Amazon SES 发送电子邮件：SendEmail 和 SendRawEmail，具体取决于发件人想要对电子邮件格式的控制程度。通过发送授权策略，您可以限制委托发件人使用两种操作中的一种。但是，许多身份所有者都会将电子邮件发送调用的具体方式交给委托发件人决定，也就是在策略中同时启用这两种操作。

> **Note**
>
> 如果您允许委托发件人经由 SMTP 接口访问 Amazon SES，则必须至少选择 SendRawEmail。

如果您的使用案例中需要限制操作，则可在发送授权策略中仅包括一种操作。以下示例向您说明如何限制仅使用 SendRawEmail 操作。

JSON

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

限制电子邮件发件人的显示名称

某些电子邮件客户端显示邮件发送者的易记名称 (如果电子邮件标头中提供) 而不是实际的“发件人”地址。例如，“John Doe <johndoe@example.com>”的显示名称为 John Doe。例如，您可能是从 user@example.com 发送电子邮件，但您希望收件人看到该电子邮件来自 Marketing 而不是来自 user@example.com。以下政策允许从 identity e xample.com 发送 AWS 账户 ID 123456789012，但前提是 “发件人” 地址的显示名称包含营销信息。

JSON

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

```
}
```

使用多个语句

发送授权策略可以包含多个语句。以下示例策略包含两个语句。第一项声明授权两 AWS 账户 人从 sender@example.com 发送，只要 “发件人” 地址和反馈地址都使用域名 e xam ple.com 即可。第二条语句授权 IAM 用户从 sender@example.com 发送电子邮件，条件是收件人的电子邮件地址使用 example.com 域。

JSON

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

向委托发件人提供有关 Amazon SES 发送授权的身份信息

在您创建发送授权策略并将其附加到您的身份后，可以向委托发件人提供身份的 Amazon Resource Name (ARN)。委托发件人在电子邮件发送操作或电子邮件的标头中将该 ARN 传递给 Amazon SES。要查找您身份的 ARN，请执行以下步骤。

查找身份的 ARN

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择已验证身份。
3. 在身份列表中，选择附加了发送授权策略的身份。
4. 在 Summary（摘要）容器中，第二列 Amazon Resource Name (ARN) 将包含身份的 ARN。该值类似于 arn:aws:ses:us-east-1:123456789012:identity/user@example.com。复制整个 ARN 并将它提供给您的委托发件人。

> **Important**
>
> 如果作为全球终端节点功能的一部分，在次要区域中复制了您正在授权的身份，请将区域参数（例如，）替换为星号us-east-1，如下*例所示。arn:aws:ses:*:123456789012:identity/user@example.com

Amazon SES 发送授权的委托发件人任务

作为委托发件人，您将代表不属于自己但已获得使用授权的身份发送电子邮件。即使您代表身份所有者发送，退回邮件和投诉也会计入您 AWS 账户的退回和投诉指标，您发送的消息数量也计入您的发送配额。如果发送身份拥有者的电子邮件时需要提出发送配额提升申请，该申请也由您负责提交。

作为委托发件人，您必须完成以下任务：

- 向身份拥有者提供信息
- 使用委托发件人通知
- 代表身份拥有者发送电子邮件

向 Amazon SES 发送授权的身份拥有者提供信息

作为委托发件人，您必须向身份所有者提供您的 AWS 账户 ID 或 IAM 用户的 Amazon 资源名称 (ARN)，因为您将代表身份所有者发送电子邮件。身份所有者需要您的账户信息，这样他就可以创建一个策略，授予您从其验证身份之一发送的权限。

如果您想使用自己的 SNS 主题，可以请求身份所有者配置反馈通知，以便发送到一个或多个 SNS 主题的退回、投诉或交付。执行此操作，您需要与身份所有者共享 SNS 主题 ARN，以便其可以在授权您发送的已验证身份中配置您的 SNS 主题。

以下步骤说明了如何查找您的账户信息以及要与您的身份所有者共享的 SNS 主题 ARNs 。

要查找您的 AWS 账户 ID

1. 登录到 a AWS Management Console t https://console.aws.amazon.com。
2. 在控制台的右上角，扩展您的账号，然后选择从下拉菜单中的 My Account（我的账户）。
3. 账户设置页面将打开并显示您的所有账户信息，包括您的 AWS 账户 ID。

要查找您的 IAM 用户 ARN

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Users（用户）。
3. 在用户列表中，选择用户名。Summary 部分将显示 IAM 用户 ARN。ARN 与以下示例类似：arn:aws:iam::123456789012:user/John。

要查找您的 SNS 主题 ARN

1. 在 v3/home 上打开亚马逊 SNS 控制台。https://console.aws.amazon.com/sns/
2. 在导航窗格中，选择 Topics（主题）。
3. 在主题列表中，SNS 主题显示 ARNs 在 AR N 列中。ARN 类似于以下示例：arn: aws: sns: us-east-1 :4444555566 66:。my-sns-topic

使用 Amazon SES 发送授权的身份拥有者通知

作为跨账户电子邮件的委托发件人，您代表您未拥有但已获得授权使用的身份发送电子邮件；但是，退回邮件和投诉仍然计入您的退回邮件和投诉指标，而不是身份拥有者的指标。

如果账户的邮件退回率或投诉率过高，则您的账户有被审核的风险，或其发送电子邮件的功能将被暂停。因此，设置通知并制定流程来监控它们十分重要。您还需要建立一套流程从邮件列表中删除已退回或遭投诉的地址。

因此，作为委托发件人，您可以将 Amazon SES 配置为在您代表您未拥有但经身份拥有者授权使用的任何身份发送的电子邮件发生退回邮件和投诉事件时发送通知。您还可以设置事件发布以将退回邮件和投诉通知发布到 Amazon SNS 或 Firehose。

> **Note**
>
> 如果将 Amazon SES 设置为使用 Amazon SNS 发送通知，则按标准 Amazon SNS 费率收取所接收通知的费用。有关更多信息，请参阅 Amazon SNS 定价页面。

创建新的委托发件人通知

您可以通过使用事件发布的配置集，或使用配置了您自己的 SNS 主题的经验证身份来设置委托发送通知。

下面给出了使用任何一种方法设置新的委托发送通知的过程：

- 通过配置集进行事件发布
- 针对您拥有的 SNS 主题的反馈通知

通过代表发送的配置集设置事件发布

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 按照 创建事件目标 所述的过程操作。
3. 在配置集中设置事件发布后，当您使用身份所有者授权您发送的经验证的身份以委托发件人身份发送电子邮件时，指定配置集的名称。请参阅 代表身份拥有者发送电子邮件。

为代表发送的 SNS 主题设置反馈通知

1. 在决定要用于反馈通知的 SNS 主题之后，请按照以下步骤操作查找您的 SNS 主题 ARN，然后复制完整的 ARN 并与身份所有者共享。
2. 要求身份所有者配置 SNS 主题，以获得有关他授权发送的共享身份的反馈通知。（您的身份所有者将需要遵循在授权策略过程中 configuring SNS topics（配置 SNS 主题）的过程）。

使用 Amazon SES 发送授权代表身份拥有者发送电子邮件

作为委托发件人，您发送电子邮件的方式与其他 Amazon SES 发件人相同，不同的是，您需要提供身份拥有者已授权您使用的身份的 Amazon Resource Name (ARN)。当您调用 Amazon SES 来发送电子邮件时，Amazon SES 会检查您指定的身份是否包含授权您发送电子邮件的策略。

有几种不同的方法可用于指定在发送电子邮件时的身份的 ARN。您使用的方法取决于您是使用 Amazon SES API 操作还是 Amazon SES SMTP 接口来发送电子邮件。

> **Important**
>
> - 要成功发送电子邮件，您必须连接到身份所有者验证身份的 AWS 区域中的 Amazon SES 终端节点。
> - 此外，必须先将身份所有者和委托发件人的 AWS 账户从沙箱中删除，然后两个账户才能向未经验证的地址发送电子邮件。有关更多信息，请参阅 请求生产访问权限（从 Amazon SES 沙盒中移出）。
> - 如果您被授权使用的身份作为全球终端节点功能的一部分在辅助区域中复制：
>   - 身份所有者应向您提供一个身份 ARN，其区域参数（例如，）替换为星号us-east-1，如下*例所示。arn:aws:ses:*:123456789012:identity/user@example.com
>   - 身份所有者应该已经在主要和次要区域为您创建了发送授权策略。

使用 Amazon SES API

与任何 Amazon SES 电子邮件发件人一样，如果您通过 Amazon SES API（直接通过 HTTPS 或间接通过 AWS SDK）访问 Amazon SES，则可以在三种电子邮件发送操作中进行选择：`SendEmail`、`SendTemplatedEmail` 和 `SendRawEmail`。Amazon Simple Email Service API 参考详细介绍了这些参数 APIs，但我们在此处提供了发送授权参数的概述。

SendRawEmail

如果您想要使用 `SendRawEmail` 以便可以控制电子邮件的格式，您可以通过以下两种方式之一指定委托授权身份：

- 向 **`SendRawEmail`** API 传递可选参数。下表中描述了必需参数：

参数	描述
`SourceArn`	与发送授权策略相关联的身份的 ARN 允许您使用在 `SendRawEmail` 的 `Source` 参数中指定的电子邮件地址发送电子邮件。 **Note** 如果仅指定 `SourceArn`，Amazon SES 会将“From”地址和“Return Path”地址设置为在 `SourceArn` 中指定的身份。
`FromArn`	与发送授权策略相关联的身份的 ARN 允许您在原始电子邮件的标头中指定特定的“发件人”地址。
`ReturnPathArn`	与发送授权策略相关联的身份的 ARN 允许您使用在 `SendRawEmail` 的 `ReturnPath` 参数中指定的电子邮件地址。

- 在电子邮件中包括 X 标头。X 标头是自定义标头，可以作为标准电子邮件标头的补充来使用（例如 From、Reply-To 或 Subject 标头）。Amazon SES 能识别三种 X 标头，您可以用它们来指定发送授权参数：

> **Important**
>
> 请不要在 DKIM 签名中包含这些 X 标头，因为它们将在发送电子邮件之前被 Amazon SES 删除。

X 标头	描述
X-SES-SOURCE-ARN	对应于 SourceArn 。
X-SES-FROM-ARN	对应于 FromArn。
X-SES-RETURN-PATH-ARN	对应于 ReturnPathArn 。

Amazon SES 会从电子邮件中删除所有 X 标头，然后再发送。如果电子邮件中包含 X 标头的多个实例，则 Amazon SES 仅使用第一个实例。

以下示例显示了一封包含发送授权 X 标头的电子邮件：

```
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 和 SendTemplatedEmail

如果您使用 SendEmail 或 SendTemplatedEmail 操作，则可以通过传入下面的可选参数来指定委托的授权身份。使用 SendEmail 或 SendTemplatedEmail 操作时，不能使用 X 标头方法。

参数	描述
SourceArn	与发送授权策略相关联的身份的 ARN 允许您使用在 SendEmail 或 SendTemplatedEmail 的 Source 参数中指定的电子邮件地址发送电子邮件。
ReturnPathArn	与发送授权策略相关联的身份的 ARN 允许您使用在 SendEmail 或 SendTemplatedEmail 的 ReturnPath 参数中指定的电子邮件地址。

以下示例显示如何使用 SendEmail 或 SendTemplatedEmail 操作和适用于 Python 的 SDK 来发送包含 SourceArn 和 ReturnPathArn 属性的电子邮件。

```
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'])
```

使用 Amazon SES SMTP 接口

使用 Amazon SES SMTP 接口进行委托发送时，必须在邮件中包含 X-SES-SOURCE-ARN、X-SES-FROM-ARN 和 X-SES-RETURN-PATH-ARN 标头。在 SMTP 会话中发出 DATA 命令后将传递这些标头。

使用模拟器在 Amazon SES 中发送测试电子邮件

我们建议使用 Amazon SES 控制台发送 SES 测试电子邮件。由于控制台要求您手动输入信息，因此您通常仅使用它来发送测试电子邮件。在开始使用 Amazon SES 之后，您很可能会使用 Amazon SES SMTP 接口或者 API 来发送电子邮件。但控制台对监控您的发送活动很有用。

以下主题介绍如何从控制台和通过手动发送电子邮件来使用邮箱模拟器：

- 从控制台使用邮箱模拟器
- 手动使用邮箱模拟器

从控制台使用邮箱模拟器

> **Important**
>
> - 在本教程中，您将从控制台向自己发送电子邮件，以便检查是否收到了该电子邮件。如需进一步试验或进行负载测试，请参阅手动使用邮箱模拟器。
> - 您发送到邮箱模拟器的电子邮件不会计入您的发送配额或您的退信率和投诉率，也不影响 Virtual Deliverability Manager 指标。

在按照这些步骤进行操作之前，请完成设置 Amazon Simple Email Service中的任务。

从 Amazon SES 控制台发送测试电子邮件

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的 “配置” 下，选择 “身份”。
3. 从 Identities（身份）表中，选择已验证的电子邮件身份（通过直接单击身份名称而不是选中其复选框）。如果您没有经过验证的电子邮件身份，请参阅 创建电子邮件地址身份。
4. 在所选电子邮件身份的详细信息页面上，选择 Send test email（发送测试电子邮件）。
5. 对于邮件详细信息，选择电子邮件格式。有以下两个选项：

   - Formatted (格式化) – 这是最简单的选项。如果您只是想将消息文本键入到 Body 文本框中，则选择此选项。当您发送电子邮件时，Amazon SES 会为您将文本转换为电子邮件格式。
   - Raw (原生) – 如果您要发送更复杂的邮件（例如包含 HTML 或附件的邮件），请选择此选项。由于此灵活性，您需要设置消息的格式，如使用 Amazon SES API v2 发送原始电子邮件中所述，然后将整个设置了格式的消息 (包括标头) 粘贴到 Body 文本框中。您可以使用以下示例 (包含 HTML) 来发送采用 Raw 电子邮件格式的测试电子邮件。将此消息整个复制并粘贴到 Body 文本框中。确保 `MIME-Version` 标头和 `Content-Type` 标头之间没有空白行；这两行之间的空白行将导致电子邮件的格式为纯文本而不是 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. 通过展开 Scenario（场景）列表框选择要测试的模拟电子邮件场景的类型。

   - 如果您选择 Custom（自定义）并且仍在 Amazon SES 沙盒中，请确保 Custom recipient（自定义收件人）字段中的地址是经过验证的电子邮件地址。有关更多信息，请参阅 创建电子邮件地址身份。

7. 根据需要填写其余字段。
8. 选择 Send Test Email（发送测试电子邮件）。
9. 登录您将电子邮件发送到的地址的电子邮件客户端。您将找到已发送的电子邮件。

手动使用邮箱模拟器

Amazon SES 包含一个邮箱模拟器，您可以用它来测试您的应用程序如何处理各种电子邮件发送场景。邮箱模拟器非常有用，例如，在您希望测试电子邮件发送应用程序而无需创建虚构电子邮件地址时，或者当您希望确定系统的最大吞吐量而不会影响每日发送配额时。

重要注意事项

在使用 Amazon SES 邮箱模拟器时，请注意以下功能和限制：

- 即使您的账户位于 Amazon SES 沙盒中，您也可以使用邮箱模拟器。
- 发送到邮箱模拟器的电子邮件限制为您的账户的最大发送速率，但不会影响您的每日发送配额。例如，如果授权您的账户每 24 小时周期发送 10000 封邮件，而您向邮箱模拟器发送了 100 封邮件，您最多仍可以向普通收件人发送 10000 封邮件而不会超过您的发送配额。
- 发送到邮箱模拟器的电子邮件不会影响您的电子邮件送达率或声誉指标。例如，如果您发送大量邮件到电子邮件模拟器的退信地址，它不会在声誉指标控制台页面上显示消息来警告您邮件退回率太高。
- 在计费方面，发送到 Amazon SES 邮箱模拟器的电子邮件与使用 Amazon SES 发送的任何其他电子邮件相同。换而言之，对于您发送给邮箱模拟器的邮件以及发送到普通收件人的邮件，我们将向您收取相同的费用。
- 邮箱模拟器支持加标签，利用此功能，您可以通过多种方式将电子邮件发送到同一邮箱模拟器地址，也可以查看应用程序如何处理可变信封退回路径 (VERP)。例如，您可以将电子邮件发送到 bounce

+label1@simulator.amazonses.com 和 bounce+label2@simulator.amazonses.com 以测试您的应用程序是否可以使用导致退回的电子邮件地址匹配退信。

- 如果您使用邮箱模拟器模拟来自同一个发送请求的多个退信，Amazon SES 会将退信回复合并成单个回复。

使用邮箱模拟器

要使用电子邮件模拟器，请在下表中找到场景，然后将电子邮件发送到对应的电子邮件地址。

> **Note**
>
> 向邮箱模拟器地址发送电子邮件时，必须使用软件开发工具包、Amazon SES 控制台、Amazon SES AWS SMTP 接口或亚马逊 SES API 通过 Amazon SES 发送。 AWS CLI邮箱模拟器不会回复接收自外部来源的电子邮件。

模拟的场景	电子邮件地址
成功送达 – 收件人的电子邮件提供商接受了电子邮件。如果您按照为 Amazon SES 设置事件通知中所述方式设置送达通知，那么 Amazon SES 会通过 Amazon Simple Notification Service（Amazon SNS）将送达通知发送给您。	success@simulator.amazonses.com
退信 – 收件人的电子邮件提供商拒绝您的电子邮件并返回 SMTP 550 5.1.1（“未知用户”）响应代码。Amazon SES 会生成一条退信通知，根据您的账户设置，通过电子邮件将其发送给您或者发送通知到 Amazon SNS 主题。邮箱模拟器电子邮件地址不会像一般的电子邮件地址一样在出现硬退信时被加入 Amazon SES 黑名单。您从邮箱模拟器收到的退信响应符合 RFC 3464。有关如何接收退信反馈的信息，请参阅为 Amazon SES 设置事件通知。	bounce@simulator.amazonses.com

模拟的场景	电子邮件地址
自动回复 – 收件人的电子邮件提供商接受您的电子邮件并将其传送到收件人的收件箱。电子邮件提供商发送自动回复 (例如“out of the office (外出，OOTO)”消息) 到电子邮件 Return-Path 标头中的地址，如果没有 Return-Path 标头，则发送到信封发件人 ("MAIL FROM") 地址。您从邮箱模拟器收到的自动回复符合 RFC 3834 标准。	ooto@simulator.amazonses.com
投诉 – 收件人的电子邮件提供商接受您的电子邮件并将其传送到收件人的收件箱。收件人在自己的邮件客户端中确定您的邮件是未经请求的，并单击 Mark as Spam (标记为垃圾邮件)。Amazon SES 随后通过电子邮件或通知 Amazon SNS 主题（具体取决于您的账户设置）将投诉通知转发给您。您从邮箱模拟器收到的投诉响应符合 RFC 5965。有关如何接收投诉反馈的信息，请参阅为 Amazon SES 设置事件通知。	complaint@simulator.amazonses.com
收件人地址在黑名单上 – Amazon SES 生成硬退信，就像是收件人的地址在全局黑名单上一样。	suppressionlist@simulator.amazonses.com

测试拒绝事件

您经由 Amazon SES 发送的每个邮件都会经过病毒扫描。如果您发送一个包含病毒的邮件，Amazon SES 会接受该邮件，检测病毒，然后拒绝整个邮件。当 Amazon SES 拒绝邮件时，它会停止处理邮件，并且不会尝试将邮件发送到收件人的邮件服务器。然后，它将生成一个拒绝事件。

Amazon SES 邮箱模拟器不包含用于测试拒绝事件的地址。但是，您可以使用欧洲计算机防病毒研究所 (EICAR) 测试文件来测试拒绝事件。此文件是一个以安全的方式测试反病毒软件的行业标准方法。要创建 EICAR 测试文件，请将以下文本粘贴到一个文件中：

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

将该文件另存为 `sample.txt`，将其附加到电子邮件，然后将电子邮件发送到经验证的地址。如果该电子邮件没有任何其他问题，Amazon SES 会接受该邮件，但接着会拒绝它，如同它包含真实病毒一样。

> **Note**
>
> 拒绝的电子邮件（包括通过使用上述过程发送的电子邮件）将被计入您的每日发送配额中。我们将向您发送的每封邮件收费，包括被拒绝的邮件。

要了解有关 EICAR 测试文件的更多信息，请参阅 Wikipedia 上的 EICAR 测试文件页面。

在 SES 中使用配置集

配置集是可以应用于已验证身份的规则组。已验证身份是您用于通过 Amazon SES 发送电子邮件的域、子域或电子邮件地址。当您对电子邮件应用配置集时，该配置集中的所有规则将应用至该电子邮件。

您可以使用配置集对您的电子邮件发送应用以下类型的规则，并可以包含其中一种或两种规则：

- 活动目的地 — 允许您发布电子邮件发送指标，包括您发送的每封电子邮件的发送次数、送达次数、打开次数、点击次数、退回次数以及向其他 AWS 产品投诉的次数。例如，您可以将电子邮件指标发送到 Amazon Data Firehose 目标，然后使用适用于 Apache Flink 的亚马逊托管服务对其进行分析。此外，您也可以将退信和投诉信息发送到 Amazon SNS，并在这些事件发生时立即收到通知。
- IP 池管理 – 如果您租赁专用 IP 地址在 Amazon SES 中使用，可以使用这些地址创建组（称为专用 IP 池）以用来发送特定类型的电子邮件。例如，您可以将这些专用 IP 池与配置集关联，并使用其中一个池发送营销通讯，并使用另一个池发送事务电子邮件。这样，您的事务性电子邮件的发件人信誉就会与营销电子邮件的信誉隔离。

要使用以下方法将配置集与经过验证的身份相关联，可通过以下方式完成：

- 在电子邮件的标头中包含对该配置集的引用。有关在电子邮件中指定配置集的更多信息，请参阅在您发送电子邮件时指定配置集。
- 指定要用作身份的现有配置集默认配置集，无论是在身份创建时，还是稍后在编辑经验证的身份时。请参阅了解默认配置集。

内容

在 SES 中创建配置集

您可以使用 SES 控制台、Amazon SES API v2 中的 `CreateConfigurationSet` 操作或 SES CLI v2 中的 `aws sesv2 create-configuration-set` 命令来创建新的配置集。本节介绍如何使用 SES 控制台和 Amazon SES CLI v2 创建配置集。

创建配置集 (控制台)

要使用 SES 控制台创建配置集，请执行以下步骤：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择配置集。
3. 选择创建集。
4. 一般细节-本节提供自定义配置集的选项：

   - 配置集名称 – 配置集的名称。名称最多可包含 64 个字母数字字符，仅包括字母、数字、连字符 (-) 和下划线 (_)。
   - 发送 IP 池 – 当您使用此配置集发送电子邮件时，将从指定池中的专用 IP 地址发送消息。从列表中选择一个 IP 池。

     > **Note**
     >
     > 默认 (ses-default-dedicated-pool) 包含尚未分配给任何其他地址池的专用 IP 地址。要了解有关管理 IP 池详情，请参阅分配 IP 池。

   - 跟踪选项
     - 使用自定义重定向域 – 选中该复选框，可使用自定义重定向域来处理使用此配置集发送的电子邮件的打开和单击跟踪。
     - 自定义重定向域-从选择经过验证域列表中选择一个经过验证的域作为您的自定义重定向域。您也可以在输入子域字段中输入一个子域。

       > **Note**
       >
       > 可以按如下方式指定自定义重新导向域：
       >
       > - 您必须先在要发送和跟踪的电子邮件中 AWS 区域 创建和验证自定义重定向域，并设置内容分发网络 (CDN)。相关解释，请参阅配置自定义域以处理打开和单击跟踪。
       > - 然后，要使用您的自定义重定向域进行打开和单击跟踪，您必须在本步骤中创建或编辑配置集时指明该域。
       > - 最后，在指定您的自定义重定向域后，查看 DNS 记录将出现在配置集的一般详细信息容器中。如果将其展开，则会看到包含正在使用的跟踪域的 CNAME

记录。 AWS 区域例如，如果您的自定义子域名为 marketing.example.com，并且是在 AWS 区域 us-east-1 中创建的，则展开查看 DNS 记录将显示一条包含以下值的 CNAME 记录：名称 = marketing.example.com 并且值 = r.us-east-1.awstrack.me。

您可以使用此信息直接来确认在设置 CDN（如配置自定义域以处理打开和单击跟踪中所述）时，是否已从表中选择了正确的跟踪域，也可以先执行此操作，然后使用此处的 CNAME 记录值来设置 CDN。

- HTTPS 策略 — 为自定义重定向域的打开和单击跟踪链接的协议选择 HTTPS 策略选项：
  - 可选 – （默认行为）打开跟踪链接将使用 HTTP 进行包装。单击跟踪链接将使用链接的原始协议进行包装。
  - 必选 – 打开和单击跟踪链接都将使用 HTTPS 进行包装。
  - 打开跟踪必选 – 打开跟踪链接将使用 HTTPS 进行包装。单击跟踪链接将使用链接的原始协议进行包装。
- 高级交付选项 – 选择左侧的箭头可展开高级交付选项部分。
  - 传输层安全性协议（TLS）– 若要求 SES 与接收邮件服务器建立安全连接，并使用 TLS 协议发送电子邮件，请选中必选复选框。

> **Note**
>
> SES 支持 TLS 1.2，但建议使用 TLS 1.3。要了解更多信息，请参阅SES 中的基础设施安全性。

  - 最长传送持续时间 – 要指定 SES 尝试通过此配置集传送电子邮件的时间限制，请输入一个以秒为单位的值，范围从 300 到 50400 不等。

> **Note**
>
> 设置自定义的最大送达限制（小于 SES 默认值 14 小时）可能非常有用，例如对时间敏感的电子邮件（例如包含 a 的电子邮件 one-time-password）、交易电子邮件以及您要确保不在非工作时间送达的电子邮件。
>
> > **Tip**
> >
> > - 要将分钟转换为秒，请乘以 60，例如 7 分钟 * 60 = 420 秒。

- 要将小时转换为秒，请乘以 3600，例如，2 小时 * 3600 = 7200 秒。

5. 信誉选项-本节介绍如何设置信誉指标：

   - 信誉指标-用于跟踪使用此配置集发送 CloudWatch 的电子邮件的退回和投诉指标。（需支付额外费用，请参阅每个指标的价格 CloudWatch。）
     - 已启用 – 选中此复选框可为配置集启用声誉指标。

6. 禁止列表选项 — 本节提供了定义自定义禁止的决策集，首先是使用此配置集来覆盖账户级别禁止的选项。configuration set-level suppression logic map（配置集级别抑制逻辑映射）将帮助您了解覆盖组合的影响。这些多层次的覆盖选择可以组合起来实现三种不同级别的抑制：

   a. 使用账户级别抑制：不要覆盖您的账户级别抑制，也不要实施任何配置集级别的抑制 - 基本上，使用此配置集发送的任何电子邮件都将使用您的账户级别抑制。要实现此目的，应按照以下步骤进行：

      - 在 Suppression list settings（抑制列表设置）中，取消选中 Override account level settings（覆盖账户级别设置）的复选框。

   b. 请勿使用任何抑制：在不启用任何配置集级别抑制的情况下覆盖您的账户级别抑制 - 这意味着使用此配置集发送的任何电子邮件都不会使用任何账户级别的抑制；换句话说，所有抑制都将被取消。要实现此目的，应按照以下步骤进行：

      i. 在抑制列表设置中，勾选覆盖账户级别设置复选框。

      ii. 在抑制列表设置中，取消勾选 Enabled (已启用)复选框。

   c. 使用配置集级别抑制：使用此配置集中定义的自定义黑名单设置覆盖您的账户级别抑制 - 这意味着使用此配置集发送的任何电子邮件将仅使用自己的隐藏设置并忽略任何账户级别的抑制设置。要实现此目的，应按照以下步骤进行：

      i. 在黑名单设置中，勾选覆盖账户级别设置复选框。

      ii. 在黑名单中，勾选已启用。

      iii. 在指定原因…中，选择要使用此配置集的抑制原因之一。

7. 虚拟可交付性管理器选项 — 只有在启用了虚拟交付管理器功能时，才会显示此部分。在这里，您可以通过覆盖账户级别的虚拟交付管理器设置中的定义方式，来定义此配置集将如何使用互动跟踪和优化的共享交付的自定义设置：

   a. 要针对此配置集禁用互动跟踪和优化共享送达，请执行以下操作：

i. 选中 Override account level settings（覆盖账户级设置）复选框。

ii. 确保针对互动跟踪和优化共享送达取消选中 Enabled（启用），然后选择 Save changes（保存更改）。

b. 要针对此配置集启用或禁用“互动跟踪”和/或“优化共享送达”，请执行以下操作：

i. 选中 Override account level settings（覆盖账户级设置）复选框。

ii. 针对互动跟踪和/或优化共享送达，选中或取消选中 Enabled（启用），然后选择 Save changes（保存更改）。

c. 要恢复到 Virtual Deliverability Manager 账户级设置，以针对此配置集启用互动跟踪和优化共享送达，请执行以下操作：

- 取消选中 Override account level settings（覆盖账户级设置）复选框，然后选择 Save changes（保存更改）。

8. 存档选项-本节提供了存档从此配置集发送的电子邮件的选项：

a. 选中 Enabled (启用) 复选框。

b. 在 “存档” 字段内单击，然后从列表中选择一个档案，然后选择 “保存更改”，或者选择 “创建存档” 并继续执行其余步骤。

c. 在存档名称字段中输入唯一的名称。

d. （可选）在保留期字段中选择一个保留期，以覆盖默认的 180 天保留期。

e. （可选）您可以通过在 KMS 密钥 ARN 字段中输入自己的 AWS KMS 密钥或选择创建 KMS 密 AWS 钥来加密您的档案。

f. 选择创建存档。

9. 标签 — 在本节中，您可以选择向配置集添加一个或多个标签：

a. 选择添加新标签。

b. 输入标签键。

c. 输入标签值（可选）。

要删除您输入的标签，请为该标签选择删除。您最多可输入 50 个标签。

10. 选择创建集可创建配置集。

现在已创建配置集，您可以选择为配置集定义事件目标，以支持根据您为事件目标指定的事件类型触发的事件发布。配置集可以具有已定义多个事件类型的多个事件目标。请参阅创建 Amazon SES 事件目标。

创建一个配置集 (AWS CLI)

可以使用 JSON 文件作为 AWS CLI中 `aws sesv2 create-configuration-set` 命令的输入来创建配置集。

1. 创建 CLI 输入 JSON 文件

   使用您常用的文件编辑工具创建包含以下键，以及对您的环境有效的值的 JSON 文件，或使用 SES API v2 `aws sesv2 create-configuration-set` 命令及未指定值的 `--generate-cli-skeleton` 选项，将示例 JSON 结构打印到标准输出。

   此示例使用名为 `create-configuration-set.json` 的文件：

```
{
    "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"]
    },
    "ArchivingOptions": {
        "ArchiveArn": "arn:aws:ses:us-east-1:123456789012:mailmanager-
archive/MyArchiveID"
    }
}
```

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，Linux 则使用正斜杠 (/)。

2. 使用创建的文件作为输入，运行以下命令。

```
aws sesv2 create-configuration-set --cli-input-json file://create-configuration-
set.json
```

> **Note**
>
> 要查看此命令的 AWS CLI 参考资料，请参阅create-configuration-set。

在 Amazon SES 中管理配置集

创建完配置集后，您可以使用 Amazon SES 控制台、Amazon SES API v2 和 Amazon SES CLI v2 通过查看、更新和删除选项进行管理。还可以将配置集作为每次从身份发送电子邮件时应用的默认配置集分配给已验证的身份。

本节中的主题：

- 了解默认配置集
- 创建 Amazon SES 事件目标
- 在 Amazon SES 中分配 IP 池
- 配置自定义域以处理打开和单击跟踪

查看、编辑和删除配置集（控制台）

访问现有配置集的详细信息页面

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择配置集。
3. 从配置集列表中选择一个名称，在 “概述” 选项卡中打开其详细信息页面，您可以在其中查看、编辑或禁用所选选项。对于活动目标选项，也可以通过选择其选项卡来完成同样的操作。有关每个选项及其字段的更多信息，请参阅中的相关部分创建配置集 (控制台)。
4. 在每个配置集的详细信息页面的顶部，可以从 “概述” 或 “事件目标” 选项卡中看到，其中有以下选项：
   - 删除 – 此按钮将删除您的配置集。
   - 禁用发送 – 此按钮将停止从配置集发送电子邮件。

列出配置集 (AWS CLI)

您可以使用中的list-configuration-sets命令生成当前区域中 AWS CLI 与您的账户关联的所有配置集的列表，如下所示：

```
aws sesv2 list-configuration-sets
```

获取配置集详细信息 (AWS CLI)

您可以使用中的get-configuration-set命令 AWS CLI 来获取特定配置集的详细信息，如下所示：

```
aws sesv2 get-configuration-set --configuration-set-name name
```

删除配置集 (AWS CLI)

您可以使用中的delete-configuration-set命令删除特定的配置集，如下所示： AWS CLI

```
aws sesv2 delete-configuration-set --configuration-set-name name
```

停止从配置集 (AWS CLI) 发送电子邮件

您可以使用中的put-configuration-set-sending-options命令停止 AWS CLI 发送来自特定配置集的电子邮件，如下所示：

```
aws sesv2 put-configuration-set-sending-options --configuration-set-name name --no-sending-enabled
```

要重新开启发送，请使用 `--sending-enabled` 选项运行相同的命令，如下所示：

```
aws sesv2 put-configuration-set-sending-options --configuration-set-name name --sending-enabled
```

了解默认配置集

本节介绍了将配置集分配为默认配置集以供经验证的身份使用的概念，以帮助了解其优势和使用案例。

默认配置集会自动将其规则应用于您从与该配置集关联的电子邮件身份发送的所有邮件。您可以在创建身份期间或事实之后将默认配置集应用于电子邮件地址和域身份，作为现有身份的编辑功能。

默认配置集注意事项

- 必须先创建配置集，然后才能将其与身份关联。
- 只有在验证身份之后，才会应用默认配置集。
- 一个电子邮件身份每次只能与一个配置集相关联。然而，您可以将同一个配置集应用于多个身份。
- 电子邮件地址级别的默认配置集会覆盖域级别的默认配置集。例如，与 joe@example.com 关联的默认配置集会覆盖 example.com 的域的默认配置集。
- 域级别的默认配置集应用于该域的所有电子邮件地址（除非您验证该域的特定地址）。
- 如果您删除被指定为某个身份的默认配置集的配置集，然后尝试通过该身份发送电子邮件，那么您对 Amazon SES 的调用将失败，并且显示“bad request”错误。

- 无法将默认配置集分配给委托发件人正在使用的已验证身份。
- 如何指定用作身份默认配置集的现有配置集实际上是经过验证的身份的函数，因此身份工作流程中会相应地给出说明：
  - 在身份创建期间指定默认配置集— 按照位于 在 Amazon SES 中创建和验证身份 章节可选步骤 6 中提供的说明操作域标识默认配置集或者电子邮件身份默认配置集。
  - 为现有身份指定默认配置集—遵照 使用控制台编辑身份 中的步骤，以及针对第 5 步的详细信息：
    a. 选择 Configuration set 选项卡。
    b. 选择默认配置集容器中的编辑。
    c. 选择列表框，然后选择要用作默认配置集的现有配置集。
    d. 继续完成 使用控制台编辑身份 中的剩余步骤。

> **Note**
>
> 如果您分配为默认配置集的配置集启用了信誉指标，则使用默认配置集发送的任何邮件都将产生额外费用，请参阅每个指标的价格 CloudWatch。

创建 Amazon SES 事件目标

事件目标允许您将以下外发电子邮件跟踪操作发布到其他 AWS 服务以进行监控：

- 发送
- 呈现失败
- 拒绝
- 已传送数
- 硬退信数
- 投诉
- 送达延迟
- 订阅
- 打开
- 点击

要了解有关设置事件发布的更多信息，请参阅the section called “使用事件发布监控电子邮件发送”。

创建事件目标

创建配置集后，您可以选择为配置集定义事件目标，以支持根据您为事件目标指定的事件类型触发的事件发布。配置集可以具有已定义多个事件类型的多个事件目标。

如果尚未创建配置集，请参阅 the section called “创建配置集”。

以下步骤说明如何创建事件目标或将事件目标添加到配置集。

要使用 Amazon SES 控制台创建或添加事件目标，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择配置集。
3. 从 Name（名称）列中选择配置集的名称以访问其详细信息。
4. 选择 Event destinations（事件目标）选项卡。
5. 选择添加目标。
6. 选择事件类型

   电子邮件发送事件是与您的发送活动相关的指标，您可以使用 Amazon SES 进行衡量。在此步骤中，请选择希望 Amazon SES 发布到事件目标的电子邮件发送事件类型。

   要了解有关事件类型详情，请参阅监控您的 Amazon SES 发送活动。

   a. 选择要发布的事件类型

      - 发送和送达 – 要选择要发布的事件类型，请选中其各自的复选框，或选择全选以发布所有事件类型。

        事件类型

        - 发送 – 发送请求成功，Amazon SES 将尝试将邮件发送到收件人的邮件服务器。
        - 呈现失败 – 由于模板呈现问题，未发送电子邮件。当模板数据丢失或模板参数与数据不匹配时，可能会发生此事件类型。（此事件类型仅在您使用 `SendTemplatedEmail` 或 `SendBulkTemplatedEmail` API 操作发送电子邮件时发生。）
        - 拒绝 – Amazon SES 已接受电子邮件，但确定它包含病毒，并且未尝试将其发送到收件人的邮件服务器。
        - 送达 – Amazon SES 已将电子邮件成功送达至收件人的邮件服务器。

- 硬退信 – 收件人的邮件服务器永久拒绝了电子邮件。（只有当 Amazon SES 重试一段时间后仍无法发送邮件时才包括软退信。）
- 投诉 – 电子邮件已成功送达收件人的邮件服务器，但收件人将其标记为垃圾邮件。
- 送达延迟 – 无法将电子邮件传送给收件人的邮件服务器，因为临时出现问题。例如，当收件人的收件箱已满，或者当接收电子邮件服务器遇到临时问题时，可能会发生传送延迟。（Amazon Pinpoint 不支持此事件类型。）
- 订阅 – 电子邮件已成功发送，但收件人通过单击电子邮件页眉中的 `List-Unsubscribe` 或页脚中的中的 `Unsubscribe` 链接更新了订阅首选项。（Amazon Pinpoint 不支持此事件类型。）

- 打开和单击跟踪 – 要测量订阅用户参与度，请选择一个或两个复选框以跟踪打开次数和单击次数。
  - 打开 – 收件人已收到邮件并在其电子邮件客户端中打开了邮件。
  - 单击 – 收件人单击了电子邮件中包含的一个或多个链接。

> **Note**
>
> 此处定义的打开和单击事件发布或任何其他配置集不会影响 Virtual Deliverability Manager 控制面板的互动跟踪选项；这些选项是通过 Virtual Deliverability Manager 的账户设置或配置集覆盖项定义的。例如，如果您通过 Virtual Deliverability Manager 禁用了互动跟踪，则它不会禁用您在 SES 事件目标中设置的打开和单击事件发布。

- 配置集重定向域— 如果在创建配置集时分配了自定义重定向域的名称，则此字段将显示并预填充自定义重定向域的名称。

> **Note**
>
> 您可以在该域下的打开和单击跟踪配置集中更新自定义重新导向域 - 请参阅创建配置集第 4 步中的跟踪选项。有关配置自定义打开和单击域的更多信息，请参阅配置自定义域以处理打开和单击跟踪。

b. 选择下一步以继续。

7. 指定目标

事件目标是一种可以向其发布电子邮件发送事件的 AWS 服务。选择合适的目标取决于要捕获的详细程度以及接收数据的方式。

a. 目标选项

- 目标类型 — 当您选择要向其发布事件的 AWS 服务旁边的单选按钮时，将出现一个详细信息面板，其中包含与该服务对应的字段。选择以下链接将提供有关服务详细信息面板的说明：
  - Amazon CloudWatch（需支付额外费用，请参阅每个指标的价格 CloudWatch。）
  - Amazon Data Firehose
  - Amazon EventBridge
  - Amazon Pinpoint [不支持 Delivery delays（送达延迟）或 Subscriptions（订阅）事件类型。]
  - Amazon SNS

  要了解有关使用事件发布模型监控电子邮件操作详情，请参阅 使用 Amazon SES 事件发布监控电子邮件发送。
- 名称 – 输入此配置集的目标名称。名称可以包含字母、数字、破折号和连字符。
- 事件发布 – 若要为此目标启用事件发布，请选中已启用复选框。

b. 选择下一步以继续。

8. 查看

   如果您确信输入正确，请选择添加目标，以添加事件目标。

您还可以使用 Amazon SES 控制台、Amazon SES API v2 或 SES CLI v2 创建事件目标。

要使用 Amazon SES API 创建事件目标，请执行以下操作：

- 有关使用 Amazon SES API 创建事件目标，请参阅 `CreateConfigurationSetEventDestination`。

编辑、禁用/启用或删除事件目标

执行以下步骤，使用 Amazon SES 控制台编辑、禁用/启用或删除事件目标：

要使用 Amazon SES 控制台编辑、禁用/启用或删除事件目标，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。

2. 在导航窗格中的配置下，选择配置集。
3. 从 Name（名称）列中选择配置集的名称以访问其详细信息。
4. 选择配置集的 Event destinations（事件目标）选项卡。
5. 在 Name（名称）列下，选择事件目标的名称。
6.
   - 编辑 – 在相应面板上，为要编辑的一组字段选择 Edit（编辑）按钮，做出所需更改，然后选择 Save changes（保存更改）。
   - 禁用或启用 – 选择右上角标记为 Disable（禁用）或 Enable（启用）的按钮。
   - 删除 – 选择右上角的 Delete（删除）按钮。

您还可以使用 Amazon SES 控制台、Amazon SES API v2 或 Amazon SES CLI v2 编辑、禁用/启用或删除事件目标。

要使用 Amazon SES API 编辑、禁用/启用或删除事件目标，请执行以下操作：

1. 有关使用 disabling/enabling SES API 的事件目的地，请参阅`UpdateConfigurationSetEventDestination`。
2. 有关使用 Amazon SES API 删除事件目标，请参阅`DeleteConfigurationSetEventDestination`。

在 Amazon SES 中分配 IP 池

您可以使用 IP 池创建用于发送特定类型的电子邮件的专用 IP 地址组。您还可以使用由所有 Amazon SES 客户共享的 IP 地址池。

将 IP 池分配给配置集时，可以从以下选项中进行选择：

- 特定的专用 IP 池 – 当您选择一个现有专用 IP 池时，使用配置集的电子邮件将只能使用属于该池的专用 IP 地址进行发送。有关如何创建以下 IP 池的过程：
  - 创建新的标准 IP 池，请参阅 为专用 IPs（标准）创建标准专用 IP 池。
  - 创建新的托管式 IP 池，请参阅 创建托管 IP 池以启用专用 IPs（托管）。
- ses-default-dedicated-pool— 此地址池包含您账户中尚未属于 IP 池的所有专用 IP 地址。如果您使用不与池关联的配置集发送电子邮件，或者在根本未指定配置集的情况下发送电子邮件，则电子邮件将从该默认池中的一个地址发送。此池由 SES 自动管理，无法编辑。
- ses-shared-pool— 该地址池包含大量 IP 地址，这些地址由所有 Amazon SES 客户共享。此选项在您需要发送与您的常规发送行为不一致的电子邮件时可能很有用。

向配置集分配 IP 池

本节介绍了使用 Amazon SES 控制台在配置集中分配和修改 IP 池的过程。

- 使用控制台向配置集分配 IP 池...
  - 创建新配置集时 – 请参阅 创建配置集 的步骤 4 中的发送 IP 池
  - 修改现有配置集时 – 选择所选配置集的一般详细信息面板中的编辑按钮，然后按照 创建配置集 的步骤 4 中的发送 IP 池的说明进行操作

配置自定义域以处理打开和单击跟踪

当您使用事件发布来捕获打开和单击事件时，Amazon SES 将对您发送的电子邮件进行细微更改。为了捕获打开事件，SES 在通过 SES 发送的每封电子邮件中添加 1 像素 x 1 像素的透明 GIF 图像，其中包括每封电子邮件的唯一文件名，并托管在 SES 运营的服务器上；当图像被下载时，SES 可以准确地指明哪封邮件已由谁打开。

默认情况下，此像素将插入到电子邮件底部；不过，一些电子邮件提供商的应用程序会在电子邮件超出特定大小时截断电子邮件预览，并且可能会提供用于查看邮件的其余部分的链接。在此场景中，SES 像素跟踪图像不会加载，并且会摆脱您试图跟踪的打开率。要解决此问题，您可以选择将像素置于电子邮件的开头或其他任何位置，方法是将 `{{ses:openTracker}}` 占位符插入电子邮件正文中。在 SES 收到带占位符的邮件后，它将替换为打开跟踪像素图像。

> **Important**
>
> - SES 将在发送时移除任何超过一个的 `{{ses:openTracker}}` 占位符。
> - 如果在电子邮件模板中使用 `{{ses:openTracker}}` 占位符，请仅添加一个，因为多个占位符会导致返回 `400 BadRequestException` 错误代码。

为了捕获链接单击事件，SES 会将电子邮件中的链接替换为指向由 SES 运营的服务器的链接。这会立即将收件人重定向到其预期目标。向该服务器发出的请求的标头（包括 Cookie）总大小不得超过 8192 字节，否则将返回 `400 BadRequestException` 错误代码。

您还可以选择使用自己的域（而非 SES 拥有和运营的域），来为收件人打造更一致的体验，这意味着将删除所有 SES 指标。您可以配置多个自定义域以处理打开和单击跟踪事件。这些自定义域与配置集关联。当您使用某个配置集发送电子邮件时，如果该配置集被配置为使用自定义域，则电子邮件中的打开和单击链接将自动使用该配置集中指定的自定义域。

本节包含一些过程，用于指示如何在您拥有的服务器上设置子域，以自动将用户重定向到由 SES 运营的打开和单击跟踪服务器。设置这些域涉及到三个步骤。首先，配置子域本身，再设置一个配置集来使用自定义域，然后设置其事件目标以发布打开和单击事件。本主题包含完成所有这些步骤的过程。

但是，如果您只希望在不设置自定义域的情况下启用打开或单击跟踪，则可以直接为配置集定义事件目标以支持针对指定的事件类型（包括打开和单击事件）触发的事件发布。配置集可以具有已定义多个事件类型的多个事件目标。请参阅创建 Amazon SES 事件目标。

第 1 部分：设置用于处理打开和单击链接重定向的域

设置重定向域的具体过程因您的 Web 托管提供商 (和您的内容传输网络，如果您使用了 HTTPS 服务器) 而异。以下各节中的过程提供了一般性指导，而不是具体步骤。

选项 1：配置 HTTP 域

如果打算使用 HTTP 域处理打开和单击链接 (与 HTTPS 域相对)，则配置子域的过程仅涉及几个步骤。

> **Note**
>
> 如果您设置了使用 HTTP 协议的自定义域，并且发送了包含使用 HTTPS 协议的链接的电子邮件，则您的客户在单击您的电子邮件中的链接时可能会看到一条警告消息。如果您计划发送包含使用 HTTPS 协议的链接的电子邮件，则应使用 HTTPS 域来处理单击跟踪事件。

设置用于处理打开和单击链接的 HTTP 子域

1. 创建用于打开和单击跟踪链接的子域。SES 建议该子域专门用于处理这些链接，并 AWS 区域 为您要跟踪的每封电子邮件创建一个子域名。
2. 验证用于 SES 的子域。有关更多信息，请参阅 创建域身份。
3. 在子域的 DNS 设置中添加新的 CNAME 记录，以将请求重定向到 SES 跟踪域。您重定向到的地址必须与您的自定义子域 AWS 区域 相同。

   - 使用中的跟踪域名表 AWS 一般参考 ，选择与您的自定义域名位于同一区域的跟踪域。

   > **Note**
   >
   > 您对子域的 DNS 记录所做的更改可能需要几分钟才能生效，具体取决于您的 Web 托管提供商。您的 Web 托管提供商或 IT 组织可能提供有关这些延迟的其他信息。

选项 2：配置 HTTPS 域

您还可以使用 HTTPS 域来跟踪链接打开和链接单击次数。要设置用于跟踪打开和链接单击次数的 HTTPS 域，除了设置 HTTP 域所需的步骤之外，还必须执行一些额外步骤。

设置处理打开和单击链接的 HTTPS 子域

1. 创建用于打开和单击跟踪链接的子域。SES 建议该子域专门用于处理这些链接，并 AWS 区域 为您要跟踪的每封电子邮件创建一个子域名。
2. 验证用于 SES 的子域。有关更多信息，请参阅 创建域身份。
3. 使用内容分发网络 (CDN)（例如 Amazon）创建新账户 CloudFront，请参阅基本 CloudFront 分发入门。
4. 将 CDN 配置为作为 SES 跟踪域的源，例如 `r.us-east-1.awstrack.me`。CDN 必须指向与您的自定义域名位于同一区域的 AWS 跟踪域。CDN 必须将请求者提供的 `Host` 标头传递给源，有关更多信息，请参阅此 AWS re:Post 文章以了解更多信息。

   - 使用中的跟踪域名表 AWS 一般参考 ，选择与您的自定义域名位于同一区域的跟踪域。
5. 如果你使用 Route 53 来管理你的域名和 CloudFront你的 CDN 的 DNS 配置，请在 Route 53 中创建一个引用你的 CloudFront分配的别名记录（例如 d111111abcdef8.cloud front.net）。有关更多信息，请参阅《Amazon Route 53 开发人员指南》中的使用 Amazon Route 53 控制台创建记录。

   否则，在您的子域的 DNS 配置中，添加一个 CNAME 记录以引用您的 CDN 的地址。
6. 从可信证书颁发机构获取 SSL 证书。该证书应涵盖在步骤 1 中创建的子域以及在步骤 3-5 中配置的 CDN。将证书上传到 CDN。
7. 您可以使用以下 curl 命令来验证新创建的自定义域是否使用了正确的区域和 HTTPS 协议。在以下示例中，除了您的域名之外，其他所有内容均为字面意思：

   ```
   curl --head https://custom.domain.com/favicon.ico
   ```

   返回的响应如以下示例所示：

   ```
   (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
```

此代码包含以下属性：

- `x-amz-ses-region` 标头值是收到请求的 SES 区域。
- `x-amz-ses-request-protocol` 标头值是 CDN 和 SES 之间的请求中标头所使用的协议。

如果您的设置正确，则该区域应反映您创建域所在的区域，协议应为 HTTPS。

第 2 部分：通过配置集指定您的自定义重定向域和 HTTPS 策略

在将您的域配置为处理打开和单击跟踪重定向后，您必须在配置集中指定自定义域和 HTTPS 策略。

当您使用某个配置集发送电子邮件时，如果该配置集被配置为使用自定义重定向域，则电子邮件中的打开和单击链接将自动使用该配置集中指定的自定义域和 HTTPS 策略。

您可以使用 SES 控制台或 `CreateConfigurationSet` v2 API 操作来完成此过程。

使用控制台指定自定义重定向域和 HTTPS 策略

- 创建或编辑配置集时，使用创建配置集步骤 4 中的跟踪选项，来指定您的自定义重定向域和 HTTPS 策略选项。

要指定自定义重定向域和 HTTPS 策略，请使用 AWS CLI

您可以在 SES API v2 中使用 `CreateConfigurationSet` 操作，并使用 `TrackingOptions` 属性来指定您的自定义重定向域和 HTTPS 策略。您可以从调用此操作， AWS CLI 如以下示例所示。

- 在要发送和跟踪电子邮件 AWS 区域 的地点中创建配置集：

```
aws sesv2 create-configuration-set --cli-input-json file://create.json
```

- 在此示例中，输入文件使用 `TrackingOptions` 属性的参数，`CustomRedirectDomain` 指定用于跟踪打开和单击链接的自定义域，`HttpsPolicy` 指定一个 HTTPS 策略选项：

```
{
    "ConfigurationSetName": "my-config-set",
    "TrackingOptions": {
        "CustomRedirectDomain": "marketing.example.com",
```

```
        "HttpsPolicy": "REQUIRE"
    },
    "SendingOptions": {
        "SendingEnabled": true
    }
}
```

对于 HttpsPolicy 参数，可以指定以下值来设置自定义重定向域的打开和单击跟踪链接的协议：

- OPTIONAL – (默认行为) 打开跟踪链接将使用 HTTP 进行包装。单击跟踪链接将使用链接的原始协议进行包装。
- REQUIRE – 打开和单击跟踪链接都将使用 HTTPS 进行包装。
- REQUIRE_OPEN_ONLY – 打开跟踪链接将使用 HTTPS 进行包装。单击跟踪链接将使用链接的原始协议进行包装。

第 3 部分：通过配置集指定打开和单击事件类型

在上一步的配置集中指定自定义域和 HTTPS 策略后，您必须通过配置集指定要在事件目标中跟踪的 open and/or click 事件类型。

您可以使用 SES 控制台或 CreateConfigurationSetEventDestination v2 API 操作来完成此过程。

要使用控制台选择打开 and/or 点击事件类型

- 创建或修改事件目标时，使用the section called “创建事件目标”步骤 6 中的打开和单击跟踪来指定事件类型。

在您发送电子邮件时指定配置集

要在发送电子邮件时使用某个配置集，必须在电子邮件的标头中传递该配置集的名称。所有 Amazon SES 电子邮件发送方法 (包括AWS CLIAWS SDKs、和 Amazon SES SMTP 接口) 都允许您传递在发送的电子邮件标题中设置的配置。

如果您使用的是 SMTP 接口或 SendRawEmailAPI 操作，则可以通过将以下标头包含在您的电子邮件中 (将 *ConfigSet* 替换为您要使用的配置集的名称) 来指定一个配置集：

```
X-SES-CONFIGURATION-SET: ConfigSet
```

本指南包括使用 AWS SDKs 和 Amazon SES SMTP 接口发送电子邮件的代码示例。其中每个示例都包含一个指定配置集的方法。要查看发送包含配置集参考的电子邮件的 step-by-step过程，请参阅以下内容：

- 使用软件开发工具包通过 Amazon AWS SES 发送电子邮件
- 使用 Amazon SES SMTP 接口发送电子邮件

查看和导出声誉指标

Amazon SES 会自动将有关您整个账户的总体退回率和投诉率的信息导出到亚马逊 CloudWatch。您可以使用这些指标在中创建警报 CloudWatch，或者使用 Lambda 函数自动暂停电子邮件发送。

您也可以将单个配置集的声誉指标导出到 CloudWatch。在配置集级别导出声誉数据让您可以更好地控制发件人声誉。

本节包括使用 Amazon SES API 将各个配置集 CloudWatch 的信誉数据导出到的过程。

启用声誉指标的导出

要开始导出配置集声誉指标，请使用 `UpdateConfigurationSetReputationMetricsEnabled` API 操作。要访问 Amazon SES API，我们建议使用 AWS CLI 或其中一个 AWS SDKs。

此过程假设您的计算机上已安装 AWS CLI 且配置正确。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

启用配置集声誉指标的导出

- 在命令行处，键入以下命令：

```
aws ses update-configuration-set-reputation-metrics-enabled --configuration-set-
name ConfigSet --enabled
```

  将前面的命令替换*ConfigSet*为要开始导出信誉指标的配置集的名称。

禁用声誉指标的导出

也可使用 `UpdateConfigurationSetReputationMetricsEnabled` API 操作禁用配置集声誉指标的导出。

禁用配置集声誉指标的导出

- 在命令行处，键入以下命令：

```
aws ses update-configuration-set-reputation-metrics-enabled --configuration-set-
name ConfigSet --no-enabled
```

将前面的命令替换*ConfigSet*为要禁止导出信誉指标的配置集的名称。

在 Amazon SES 中使用全球终端节点

Amazon SES 全球终端节点是一项增强电子邮件发送操作连续性和可靠性的功能。本章将指导您了解全球终端节点的概念、设置和使用，帮助您利用多区域发送 (MRS) 为您的电子邮件工作负载实现更高的可用性和更好的灾难恢复能力。

什么是全球终端节点？

全局终端节点是允许您在两个之间分配 SES 出站工作负载的资源 AWS 区域。配置完成后，SES 会自动在选定的主区域与次要区域之间拆分您的发送流量。如果任一区域出现损失，SES 将自动将流量从受影响区域转移出去，以保持发送操作的连续性。

使用全球终端节点的主要好处包括：

- 提高了电子邮件发送的连续性
- 区域间自动故障转移
- 简化的多区域配置

全球端点的工作原理

设置全球终端节点时，您可以选择主区域（创建终端节点的地方）和次要区域。然后，SES 会创建一个多区域终端节点 (MREP)，作为您的电子邮件发送请求的入口点。

全球终端节点设置过程会同步关键工件并将限制从您的主要区域发送到辅助区域。这样可以确保两个区域都具有等效的经过验证的身份、配置集和经批准的发送限制，足以满足所有预期的容量。

在全局终端节点准备就绪且在 SendEmail API 调用中指定其终端节点 ID 后，SES 会自动在主区域和次要区域之间均匀地路由您的出站流量。如果任一区域受损，则流量将从该区域转移到另一个区域，直到损伤得到解决。

设置全局终端节点

主题

先决条件

在创建全局终端节点之前，您首先需要授予 SES 权限，以便在您的账户中创建服务相关角色 (SLRs)。这些角色支持创建、使用和监控全球端点所需的基本服务功能和资源访问权限。这可以通过实施以下策略来实现：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "ses.amazonaws.com"
        }
      }
    }
  ]
}
```

创建全局终端节点

要创建新的全局终端节点，请执行以下操作：

1. 打开 SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中，选择全局终端节点。
3. 选择创建全局端点并在名称字段中输入名称。
4. 从下拉菜单中选择一个次要区域。（您的主要区域默认为您登录控制台时使用的区域。）
5. （可选）向您的全局终端节点添加一个或多个标签。
6. 查看配置并选择创建全局终端节点。

创建过程可能需要几秒钟。完成后，您的全球终端节点的状态将更改为 “就绪”。

使用 AWS CLI：

```
aws sesv2 create-multi-region-endpoint --primary-region us-west-2 --secondary-
region us-east-1 --endpoint-name MyGlobalEndpoint
```

在上述示例中：

- *us-west-2*替换为您的全球终端节点的主区域。
- *us-east-1*替换为全球终端节点的次要区域。
- *MyGlobalEndpoint*替换为友好名称以提供您的全球终端节点。

全局终端节点状态

全局端点可以有以下状态：

- 正在@@ 创建-正在配置资源
- 就绪-资源已准备就绪，可以使用
- 失败 — 资源配置失败
- 正在@@ 删除-正在按要求删除资源

准备次要区域

既然您已经创建了全局终端节点，那么在使用全球终端节点发送电子邮件之前，您现在必须确保您的电子邮件发送配置（包括其所有组件（身份、配置集、电子邮件模板和发送限制）在主要和次要区域中保持一致。这种协调对于避免潜在问题以及确保电子邮件的正确投递和跟踪至关重要。

控制台中的区域复制功能通过自动复制资源和将账户级别设置从主区域复制到辅助区域来为您提供帮助，这将快速帮助您确保两个区域的配置相同。

根据资源依赖关系，复制资源的顺序很重要。为避免冲突，请按照以下主题顺序进行操作：

主题

- 复制配置集
- 复制经过验证的域名身份
- 复制生产限制

复制配置集

您可以从主区域中选择多个要复制的配置集及其设置，并将其设置复制到辅助区域。

“复制配置集” 功能允许您：

- 一次将多个配置集复制到辅助区域。
- 检查主区域与辅助区域中的配置集之间的差异。

要复制配置集，请执行以下操作：

1. 在全局终端节点页面上，从名称列中选择要复制的全局终端节点。
2. 在 “复制配置集” 卡片中，展开 “配置集操作”，然后选择 “复制”。
3. 选择最多 10 个配置集，然后选择 “确认”。
4. 如果状态不成功，请选择查看报告以确定问题。
5. （可选）对于以前复制的配置集，您可以通过在重复最后三个步骤的同时选择检查差异来检查主区域与次要区域之间的差异。

> **Note**
>
> - 如果您复制的配置集包含事件目标、信誉选项、存档选项，或者电子邮件模板中引用了这些设置，则需要在辅助区域中手动配置这些设置。
> - 如果您在主区域的配置集中为已发送（出站）电子邮件启用了存档，则必须使用在辅助区域创建的同名存档，在辅助区域的配置集中手动启用已发送（出站）电子邮件的存档。

复制经过验证的域名身份

为确保全球终端节点配置有效运行，需要在主区域和次要区域验证您的发送域身份。SES 确定性 Easy DKIM (DEED) 用来简化此过程。

Deterministic Easy DKIM (DEED) 是一项功能，它 AWS 区域 基于配置了 Easy DKIM 的父域在所有域中生成一致的 DKIM 令牌。这种一致性允许 SES 在主区域验证域名后自动验证辅助区域中的域，而无需额外更新 DNS 记录。因此，您必须确保要复制的域身份（即父域身份）已使用 Easy DKIM 进行配置。

“复制已验证的域名身份” 功能允许您：

- 一次将多个域名身份复制到辅助区域。
- 使用 Deterministic Easy DKIM (DEED) 自动验证它们。
- 检查主区域与次要区域中的身份之间的差异。

要从 SES 控制台复制身份，请执行以下操作：

1. 在全局终端节点页面上，从名称列中选择要复制的全局终端节点。
2. 在 “复制已验证的域身份” 卡片中，展开 “身份操作”，然后选择 “复制”。
3. 最多选择 10 个身份，然后选择 “确认”。
4. 如果状态不成功，请选择查看报告以确定问题。
5. （可选）对于以前重复的身份，您可以通过选择检查差异来检查主区域与次要区域之间的差异，同时重复最后三个步骤。

> **Note**
>
> - 使用 BYODKIM 验证或自签名的域名身份需要在辅助区域中手动创建，因为 DEED 不适用于这种情况。
> - 使用邮件发件人属性、策略或反馈转发和通知的域身份需要在辅助区域手动配置这些功能。

复制生产限制

SES 会检查区域之间的发送限制是否一致，并允许您在需要时请求提高次要区域的限制。

“重复生产限制” 功能允许您：

- 检查主要地区和次要区域之间的产量限制是否一致。
- 如有必要，可以申请提高次要区域的限制。

要复制生产限制，请执行以下操作：

1. 在全局终端节点页面上，从名称列中选择要复制的全局终端节点。
2. 在 “重复生产限制” 卡片中，如果状态显示发送限制未对齐，请展开 “发送限制操作”。

3. 选择管理次要区域的发送限制。
4. S ervice Q uotas 页面将在次要区域打开，您可以在其中请求提高“发送配额”和“发送速率”，以匹配主要区域的值。

> **Tip**
>
> 建议您申请两个地区都有资格获得的最大配额。虽然在正常操作条件下，电子邮件流量在两个区域之间分布，但在故障转移事件期间，全部电子邮件流量将发送到一个区域，其限制应足以处理全部容量负载。

5. （可选）您也可以通过选择管理主区域的发送限制来请求提高主要地区的产量，同时重复前两个步骤。

> **Important**
>
> 至关重要的是，这两个区域都必须具有您打算用来发送电子邮件的同等经过验证的身份和配置集，以及匹配的发送限制，以确保全球终端节点的正常运行。任何差异都可能导致交付失败、故障转移可靠性降低和指标缺失。

使用全局终端节点

主题

与您的应用程序集成

在应用程序中使用全局终端节点需要获取其终端节点 ID。

要检索全球终端节点的终端节点 ID，请执行以下操作：

1. 在 SES 控制台中，转到“全局终端节点”页面，然后从“名称”列中选择要使用的全局终端节点。
2. 在全局终端节点详细信息页面上，选择终端节点 ID 下的复制图标。

使用 AWS CLI：

```
aws sesv2 get-multi-region-endpoint --endpoint-name MyGlobalEndpoint --region us-west-2
```

在上述示例中：

- *MyGlobalEndpoint*替换为您在创建全局端点时为其提供的友好名称。
- *us-west-2*替换为您创建全球终端节点的主要区域。
- API 响应将包含您的终端节点 ID 的值，例如"EndpointId": "abcdef12.g3h"。

获得全局终端节点的终端节点 ID 后，您可以更新SendEmail或 SendBulkEmailAPI 调用以包含endpoint-id参数的终端节点 ID 值。以下是如何使用以下方法在 SendEmail API 调用中指定终端节点 ID 的示例 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*替换为您通过控制台或 API 获得的实际终端节点 ID。

监控和指标

全球终端节点功能提供了主区域和次要区域的电子邮件发送量的统一视图。您可以通过 SES 控制台全局终端节点详细信息页面上的跨区域指标选项卡访问这些指标。

要访问两个地区的发送指标，请执行以下操作：

1. 在 SES 控制台中，转到全局终端节点页面，从 “名称” 列中选择要查看其指标的全局终端节点。
2. 选择全球终端节点详细信息页面上的跨区域指标选项卡，然后输入最多 31 天的日期范围。将显示给定日期范围内的两个地区的指标。

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*替换为您实际的终端节点 ID。

最佳实践和注意事项

遵循这些最佳实践和注意事项有助于确保跨多个 AWS 区域 全球端点的有效利用、监控和成本优化，从而提高电子邮件发送功能的可用性和可靠性。

- 定期在区域之间同步对工件（例如配置集、已验证身份）所做的任何更改，以保持发送的完整性。
- 监控跨区域指标，确保流量分布均衡，并发现任何潜在问题。
- 请注意，虽然全球终端节点提高了可用性，但它们不会改变 SES Outbound 区域可用性的物理状态。
- 请注意，启动时，全球终端节点不支持 SMTP 或 VPC 终端节点访问。
- 如果使用 AWS 地址转换网关，请考虑可能产生的出口费用。
- 请注意，在调用启用了 MREP 的遥远区域时，API 延迟可能会略有增加。

定价

虽然确切的定价细节可能会发生变化，但对于同等数量的邮件，预计全球端点的价格将高于单一区域发送的价格。尽管有所增加，但与其他电子邮件服务提供商相比，总体成本预计仍将保持竞争力。

要了解更多 up-to-date定价信息，请参阅 Amazon SES 定价页面。

Amazon SES 专用 IP 地址

当您创建新的 Amazon SES 账户时，原定设置情况下，将从与其他 SES 用户共享的 IP 地址发送您的电子邮件。您还可以使用保留专供您使用的专用 IP 地址，只需支付额外费用即可租用它们。这让您可以完全控制发件人声誉，并使您能够隔离电子邮件程序中不同部分的声誉。Amazon SES 提供两种预调配和管理专用 IP 地址的方法：

- 标准 — 指您手动设置和管理的专用 IP 地址，包括手动预热和扩展它们以及手动将它们移入和移出 IP 池的选项。（这些地址以前在 SES 中被称为专用 IP 地址。）
- 托管式 — 指由 SES 代表您自动设置的专用 IP 地址，可为开始使用由 SES 管理的专用 IP 地址提供快捷简便的方法；它们会自动为每个 ISP 单独预热，并根据您的发送量自动扩展，以帮助确保根据您发送电子邮件的方式，以最佳方式使用您的专用 IP 地址。

在决定是使用共享 IP 地址还是上面定义的两种类型的专用 IP 地址时，请选择能为您发送的电子邮件的类型、数量和模式带来最大优势的地址。为了帮助您做出决定，下表总结了这些优势。在 Benefit（优势）列中选择一个项目以了解更多信息。

优势	共享 IP 地址	专用 IP 地址（标准）	专用 IP 地址（托管式）
准备好立即使用	是	否	否
需要额外设置	否	是	是
IP 地址和信誉与其他 SES 客户隔离	否	是	是
容量会随着流量的增加自动增加	否	否	是
适合采用连续、可预测发送模式的客户	支持	是	是
适合采用不可预测发送模式的客户	是	否	是
适合高容量发件人	支持	是	是

优势	共享 IP 地址	专用 IP 地址（标准）	专用 IP 地址（托管式）
适合低容量发件人	是	否	否
每月额外费用	否	是	是
完全掌控发件人信誉	否	是	是
通过电子邮件类型、收件人或其他因素隔离声誉	否	是	是
提供永不更改的已知 IP 地址	否	是	否

> **Important**
>
> 如果您不打算定期且可预测地发送大量电子邮件，建议您使用共享 IP 地址。如果您想在发送模式非常不规则的情况下使用专用 IP 地址，则使用专用 IPs （托管）是更好的选择。

易于设置

共享 IP 地址 — 您无需执行任何其他配置。一旦您确认电子邮件地址并且移出沙盒，您的 SES 账户就会准备好发送电子邮件。

专用 IP 地址（标准）— 您必须通过 Su AWS ppor t Center 提交请求，并可选择配置专用 IP 池。

专用 IP 地址（托管式）— 无需提交对于专用 IP 地址的请求。当您选择加入并进行一次性演练以创建托管式专用池时，将自动分配这些地址。

声誉管理

IP 地址声誉很大程度上取决于历史发送模式和数量。电子邮件发送量在长时间内较稳定的 IP 地址通常声誉较好。

共享 IP 地址 — 这些地址在多个 SES 客户间共享，它们共同发送大量电子邮件，并且 AWS 妥善管理出站流量，以最大限度地提高共享 IP 地址的声誉。

专用 IP 地址（标准）- 在预热后，您的 IP 地址将从 SES 共享池隔离出来，您通过发送稳定且可预测数量的电子邮件来维护自己的发件人声誉。

> **Note**
>
> 有关专 IPs 用（标准）的智能网络数据服务 (SNDS) 数据的信息，请参阅专用 SNDS 指标 IPs。

专用 IP 地址（托管）— 在您的新 IP 地址预热后 IPs，这些地址将与 SES 共享池隔离，您可以维护自己的发件人信誉。此外，还有一个额外的好处，就是可以跟踪每个 ISP 的声誉，并据此优化安排传出发送。因此，您仍能保持发件人声誉。同时，与手动配置的专用 IP 地址上的同等工作负载相比，这种自动化有助于提高整体送达率并降低退回率。

发送模式的可预测性

相较于之前没有发送历史记录但突然开始发出大量电子邮件的 IP 地址，具有稳定发送电子邮件历史记录的 IP 地址拥有更好的信誉。

共享 IP 地址 — 适用于不遵循可预测模式的电子邮件发送模式。使用共享 IP 地址，您可以根据需求情况增加或减少电子邮件发送模式。

专用 IP 地址（标准）— 您必须通过每天逐渐增加电子邮件发送量来给地址预热。预热新 IP 地址的过程请参阅 预热专用 IP 地址（标准） 中的说明。预热您的专用 IP 地址后，您必须保持稳定的发送模式。

专用 IP 地址（托管式）- 使用自适应预热策略（与 SES 共享池相结合），自动为托管式池中的每个 IP 预热您的专用 IP 地址。自适应预热策略考虑了实际发送模式，以单独优化每个 ISP 的预热。托管式 IP 池会根据使用情况和对 ISP 特定策略的考虑，自动按 ISP 进行扩展。

出站电子邮件量

共享 IP 地址 — 最适合发送少量电子邮件的客户。

专用 IP 地址（标准）| 专用 IP 地址（托管式）— 两者都适合发送大量电子邮件的客户。大多数人 ISPs 只有在收到来自给定 IP 地址的大量邮件时才会跟踪该地址的信誉。对于每个您想要对其培养信誉的 ISP，您应每月至少一次在 24 小时内发送数百封电子邮件。在某些情况下，这两种类型的专用 IP 地址也可能适用于较少量的电子邮件。例如，如果您向精心挑选的一小群收件人发送电子邮件，当他们

的邮件服务器使用特定 IP 地址的列表而不是 IP 地址声誉来决定接受或拒绝电子邮件时，这类地址可以顺利发挥作用。

额外费用

共享 IP 地址 — 包含在标准 SES 定价中。

专用 IP 地址（标准）— 对于您租用的每个 IP 地址，将按月收取额外的费用。有关定价的信息，请参阅 SES 定价页面。

专用 IP 地址（托管）— 仅收取标准月费（无论 IPs 所需金额多少）和每封邮件的使用费。有关定价的信息，请参阅 SES 定价页面。

完全掌控发件人信誉

共享 IP 地址 — 您的发件人声誉由 SES 控制。

专用 IP 地址（标准）| 专用 IP 地址（托管式）— 您的发件人声誉完全由您控制。您的 SES 账户是唯一一个能够从这些地址发送电子邮件的账户。因此，发件人声誉取决于您的电子邮件发送行为。此外，专用 IPs （托管）通过使用性能最高的 IP 地址来主动监控用于发送电子邮件的出站 IP 地址，从而提高向收件人发送电子邮件的能力。使用其他服务（例如亚马逊 CloudWatch 指标和 Amazon SES 中的内置控制面板）可以显示利用率数据。

隔离发件人信誉的能力

共享 IP 地址 — 您的发件人声誉是在账户级别设置的，无法隔离。

专用 IP 地址（标准）| 专用 IP 地址（托管式）— 您可以通过创建专用 IP 池（可用于发送特定类型电子邮件的专用 IP 地址组），隔离电子邮件程序中不同组件的发件人声誉。例如，您可以创建一个专用 IP 地址池来发送营销电子邮件，并使用另一个池发送事务性邮件。

已知的不变 IP 地址

共享 IP 地址 — 您不知道 SES 用来发送邮件的 IP 地址，它们可能随时变化。

专用 IP 地址（标准）-您可以在 SES 控制台的 “专用 IPs” 页面中找到发送邮件的地址的值。这是因为专用 IP 地址是静态的。

专用 IP 地址（托管式）— SES 将根据您的发送模式自动配置专用 IP 地址的最佳数量。虽然 SES 负责管理 IP 池的自动扩展，但您可以通过 SES 控制台或 API 查看当前分配给您的账户的所有专用 IP 地址。您的资金池 IPs 中的数量将继续根据您的发送需求动态增加或减少。

Amazon SES 中的专用 IP 地址（标准）

专用 IP 地址（标准）是您在 SES 中手动设置和管理的专用 IP 地址。它们与使用 SES 功能“the section called “专用 IP 地址（托管式）””自动设置和管理的地址不同。除了允许您使用专用 IP 地址完全控制发送信誉外，专用 IPs（标准）还允许您全面管理您的专用 IP 地址 IPs，包括预热、向外扩展和 IP 池管理。

专用 IPs（标准）和专用 IPs（托管）均指您在 SES 中租用以获得额外定价的专用 IP 地址，但在实施和管理方式上有所不同。虽然两者都有共同的优点，但根据您的电子邮件发送类型，它们又有各自的独特优势，如 专用 IP 地址 中所述。

本节的主题说明了如何在 SES 中手动设置和管理专用 IPs（标准）。

主题

- 请求和释放专用 IP 地址（标准）
- 预热专用 IP 地址（标准）
- 为专用 IPs（标准）创建标准专用 IP 池

请求和释放专用 IP 地址（标准）

要使用专用 IP 地址（标准），必须先请求这些地址。当您不再需要它们时，则必须释放它们。通过中心申请并放弃专用 IPs（标准）。AWS 支持对于您为 Amazon SES 租用的每个标准专用 IP 地址，我们将按月向您的账户收取额外的费用。使用专用 IPs（标准）时没有最低承诺。

有关专用 IPs（标准）相关费用的更多信息，请参阅 Amazon SES 定价。

有关当前已推出 Amazon SES 的所有区域的列表，请参阅《Amazon Web Services 一般参考》中的 AWS 区域 和端点。要详细了解每个可用区中可用区域的数量 AWS 区域，请参阅AWS 全球基础设施。

申请专用 IPs（标准）

通过在 Support Center 中创建服务配额增加案例，您可以根据需要申请任意数量的 AWS 专用 IPs（标准）。

申请专用 IPs（标准）

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，选择专用 IPs。
3. 请执行以下操作之一：

   a. 如果您的账户 IPs 中没有专属账户：

      - 将显示专用 IPs入职页面。在专用 IPs（标准）概览面板中，选择申请专用 IPs。

        将在 AWS 支持控制台中打开 Create case（创建案例）页面。

   b. 如果您的账户 IPs 中已有专用账户：

      i. 在 “专用” IPs 页面上选择 “标准 IP 池” 选项卡。

      ii. 在标准版概述面板中，选择请求或放弃标准版专用。 IPs

        将在 AWS 支持控制台中打开 Create case（创建案例）页面。
4. 在 Create case（创建案例）下，选择页面顶部的 Service limit increase（服务限制提高）卡片。
5. 在 Case details（案例详细信息）下，填写以下部分：

   - 对于 Limit Type（限制类型），保留 SES Service Limits（SES 服务限制）。
   - 对于 Mail Type（邮件类型），选择您计划使用专用 IP 地址发送的电子邮件的类型。如果多个值适用，请选择适用于您计划发送的大部分电子邮件的选项。
   - 对于 Website URL（网站 URL），输入您的网站的 URL。提供该信息将帮助我们更好地了解您打算发送的内容类型。
   - 对于 Describe, in detail, how you will only send to recipients who have specifically requested your mail（详细描述您将如何仅向专门请求您发送邮件的收件人发送邮件），提供与您的使用案例一致的响应。
   - 对于 Describe, in detail, the process that you will follow when you receive bounce and complaint notifications（详细描述当您收到退信和投诉通知时将遵循的流程），提供与您的使用案例一致的响应。
   - 在 “您是否会遵守 AWS 服务条款和 AUP” 中，选择适用于您的用例的选项。
6. 在请求下，填写以下部分：

   - 对于区域，请选择您的请求适用的区域。 AWS 区域
   - 对于 Limit（限制），保持 Desired Dedicated IP（所需的专用 IP）。

- 对于 New limit value（新限制值），输入您需要用于实施使用案例的专用 IP 地址的数量。

> **Note**
>
> 如果您想请求专用 IP 地址以用于其他请求 AWS 区域，请选择添加其他请求，然后填写额外请求的 “区域”、“限制” 和 “新限制值” 字段 AWS 区域。对每个 AWS 区域 要在中使用专用 IP 地址的用户重复此过程。

7. 在案例描述下，对于使用案例描述，指出您希望申请专用 IP 地址。如果您要申请特定数量的专用 IP 地址，请注明。如果您未指定专用 IP 地址的数量，我们将提供满足您在上一步中指定的发送速率要求所需的专用 IP 地址数。

   接下来，说明您计划如何通过 Amazon SES 使用专用 IP 地址发送电子邮件。请提供为什么您希望使用专用 IP 地址不是共享 IP 地址的原因。此信息有助于我们更好地了解您的使用案例。
8. 在 Contact options（联系选项）下，对于 Preferred contact language（首选联系语言），请选择您希望以 English（英语）还是 Japanese（日语）接收有关此案例的通信。
9. 完成后，选择 Submit（提交）。

在提交表单后，我们将评估您的请求。如果我们对您的请求授权，则会在支持中心内回复您的案例，以确认您的新专用 IP 地址与您的账户关联。

释放标准专用 IP 地址

如果您使用的是专用 IP 地址，并且不再希望它们与您的账户关联，则以下过程说明如何通过在 Supp AWS ort Center 中创建案例来放弃这些地址。

> **Important**
>
> 释放专用 IP 地址的过程无法撤消。如果您在某个月份中间释放了专用 IP 地址，我们将根据该月已过的天数，按比例分摊该月的专用 IP 使用费。

放弃专用 IPs（标准）

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，选择专用 IPs。

3. 在 “专用” IPs 页面上选择 “标准 IP 池” 选项卡。
4. 在标准版概述面板中，选择请求或放弃标准版专用。 IPs
5. 在 Case details（案例详细信息）下，对于 Limit type（限制类型），保留 SES Service Limits（SES 服务限制）。

> **Note**
>
> 本节中剩余的方框不适用于放弃专用。 IPs将其留空。

6. 在 Requests（请求）下，填写以下部分：

   - 对于区域，请选择您的放弃请求所 AWS 区域 适用的。

   > **Note**
   >
   > 每个专用 IP 地址都是唯一的 AWS 区域，因此选择与专用 IP 地址关联的非常重要。 AWS 区域

   - 对于 Limit（限制），保持 Desired Dedicated IP（所需的专用 IP）。
   - 对于 New limit value（新限制值），输入任意数字。您在此处输入的数字并不重要，您可以指定要在下一步 IPs 中放弃的专用数量。

   > **Note**
   >
   > 一个专用 IP 地址只能用于一个 AWS 区域。如果您希望释放在其他 AWS 区域中使用的专用 IP 地址，请选择 Add another request（添加其他请求）。然后填写其他 AWS 区域的 Region（区域）、Limit（限制）和 New limit value（新限制值）字段。对于您要释放的每个专用 IP 地址，重复此过程。

7. 在 Case Description（案例描述）下，对于 Use case description（使用情形描述），说明您希望释放现有专用 IP 地址。如果您当前租用了多个专用 IP 地址，请提供您要释放的专用 IP 地址数。
8. 在 Contact options（联系选项）下，对于 Preferred contact language（首选联系语言），请选择您希望以 English（英语）还是 Japanese（日语）接收有关此案例的通信。
9. 完成后，选择 Submit（提交）。

我们收到您的请求之后，将会发送消息，要求您确认希望释放专用 IP 地址。确认您要释放 IP 地址之后，我们将这些地址从您的账户中移除。

预热专用 IP 地址（标准）

在决定是接受还是拒绝电子邮件时，电子邮件服务提供商会考虑发送电子邮件的 IP 地址的信誉。有助于提高 IP 地址的信誉的因素之一是该地址是否具有发送高质量电子邮件的历史记录。电子邮件提供商从只有少量或者没有历史记录的新 IP 地址接受邮件的可能性更低。发送自只有少量或者没有历史记录的 IP 地址的电子邮件可能会进入收件人的垃圾邮件文件夹，也可能被全部阻止。

当您从新的专用 IP 地址开始发送电子邮件时，您可以逐渐增加从该地址发送的电子邮件数量，直至使用到其全部容量。此过程称为预热 IP 地址。

预热某个 IP 地址所需的时间量因电子邮件提供商而异。对于一些电子邮件提供商，您可以在两周左右建立良好声誉，而另一些提供商则可能需要六个星期。在预热新的专用 IP 地址时，您应将电子邮件发送到您的最活跃用户以确保低投诉率。如果您收到大量阻止或限制通知，则还应仔细检查您的退回邮件并减少发送的电子邮件数。有关监控退回邮件的信息，请参阅监控您的 Amazon SES 发送活动。

专用 IPs（标准）自动预热

当您请求专用 IP 地址（标准）时，Amazon SES 将自动预热它们，以改善您发送的电子邮件的送达情况。自动 IP 地址预热功能在原定设置情况下处于启用状态。SES 会 IPs 根据预定义的预热 IPs 计划，逐渐增加您通过专属设备发送的电子邮件数量，从而自动预热您的专用。这种逐步增长有助于您在互联网服务提供商中 IPs 建立良好的声誉（ISPs）。

自动预热过程中发生的步骤取决于您是否已有专用 IP 地址。

- 当您首次申请专用 IPs（标准）时，SES 会在您的专用 IP 地址和与其他 SES 客户共享的一组地址之间分发您的电子邮件。随着时间推移，SES 会逐步增加从您的专用 IP 地址发送的邮件数。
- 如果您已经有专用 IP 地址，SES 会在您现有的专用 IPs（已预热）和新的专用 IPs（未预热）之间分配您的电子邮件发送。随着时间推移，SES 会逐步增加从您的新专用 IP 地址发送的邮件数。

> **Note**
>
> IP 自动预热是一个需要一定时间的过程。预热百分比在 45 天内稳步增加，与您的发送量无关。

在预热了专用 IP 地址之后，您应向要维护良好信誉的每个提供商每天发送大约 1000 封电子邮件。您应在用于 SES 的各个专用 IP 地址上执行此任务。

应避免在预热过程完成后立即发送大量电子邮件。而是应该缓慢地增加您发送的电子邮件数，直至达到目标数量。如果电子邮件提供商发现从某个 IP 地址发送的电子邮件数突然大量增长，他们可能会阻止或限制来自该地址的电子邮件的传送。

在专用 IPs（标准）上禁用自动预热过程

在您购买新的标准专用 IP 地址时，Amazon SES 自动为您预热它们，因为原定设置情况下为您的账户启用自动 IP 地址预热功能。如果您偏好自行预热专用 IP 地址，您可以在账户级别为所有 IP 地址禁用自动预热功能。

如果您禁用自动预热功能，则随后租用的任何专用设备都 IPs将添加到您的帐户中，预热状态为 “完成”，这使得它们无需预热即可使用，这意味着在使用它们进行常规发送之前，您有责任确保 IPs 它们已正确预热。在您禁用自动预热功能时当前处于预热状态的任何 IPs 内容都不会受到影响。

> **Important**
>
> 如果禁用自动预热功能，您就需要负责预热自己的专用 IP 地址。如果您从未经预热的地址发送电子邮件，则可能会遇到糟糕的传送率。

禁用（或重新启用）账户中所有专用 IPs（标准）的自动预热功能

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，选择专用 IPs。
3. 在 “专用” IPs 页面上选择 “标准 IP 池” 选项卡。
4. 在 Standard overview（标准概览）面板中选择 Disable auto warm-up（禁用自动预热）以禁用自动预热，或选择 Enable auto warm-up（启用自动预热）以重新启用自动预热。

手动预热专用 IPs（标准）

您可以通过编辑其预热百分比来手动增加或减少专用 IPs（标准）当前发送音量，提早结束其预热过程，将其当前发送音量设置为 0%，然后重新启动预热过程。

手动预热专用 IPs（标准）

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，选择专用 IPs。
3. 在 “专用” IPs 页面上选择 “标准 IP 池” 选项卡。
4. 在 “全标准” 专用 IPs面板中，选择一个 IP 地址并选择 “编辑预热”，然后选择以下选项之一：

   a. 编辑百分比 — 在 Warm-up percentage（预热百分比）字段中输入一个值，通过编辑您的 IP 的预热百分比，然后点击 Save changes（保存更改），以增加或减少其当前发送量。

      Warm-up status（预热状态）列将显示 `In progress`，而 Warm-up percentage（预热百分比）列将显示您输入的值。

   b. 标记为已完成 — 阅读 Mark warm-up as Complete?（将预热标记为完成？）对话框，以确认您了解提前结束自动预热过程所带来的问题，然后选择 Mark as Complete（标记为完成）。

      Warm-up status（预热状态）列将显示 `Complete`，而 Warm-up percentage（预热百分比）列将显示 `100%`。

   c. 重置百分比 - 阅读重置预热百分比？对话框，以确认您正在将 IP 的当前发送量设置为 1%，并且必须重新启动自动预热过程或手动设置预热百分比，然后选择重置。

      Warm-up status（预热状态）列将显示 `In progress`，而 Warm-up percentage（预热百分比）列将显示 `1%`。

为专用 IPs（标准）创建标准专用 IP 池

如果您购买了要在 Amazon SES 中使用的多个专用 IP 地址（标准），您可以为这些地址创建组，称为专用 IP 池。将专用 IPs（标准）组合到一个池中可以更轻松地对其进行管理。常见的情景是创建一个池来发送营销宣传材料，并创建另一个池来发送事务电子邮件。这样，您的事务性电子邮件的发件人信誉就会与营销电子邮件的信誉隔离。在这种情况下，如果营销活动产生大量投诉，事务电子邮件的传送不会受影响。

本节包含用于创建专用 IP 池的步骤。

> **Note**
>
> 您还可以使用由所有 SES 客户所共享的 IP 地址池来创建配置集。共享的 IP 池适用于您需要发送与您的常规发送行为不一致的电子邮件的情况。有关使用共享 IP 池与配置集的信息，请参阅 在 Amazon SES 中分配 IP 池。

使用 SES 控制台为专用 IPs（标准）创建专用 IP 池

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，选择专用 IPs。

   > **Note**
   >
   > 如果您的账户中目前没有任何专用 IPs（标准），则会显示专用 IPs入门页面，让您可以购买专用 IPs（标准）。有关更多信息，请参阅the section called “申请专用 IPs（标准）”。

3. 在 “专用” IPs 页面上选择 “标准 IP 池” 选项卡。
4. 在 All Dedicated IP (standard) pools [所有专用 IP（标准）池] 面板中，选择 Create Standard IP pool（创建标准 IP 池）。

   Create IP Pool（创建 IP 池）页面将打开。
5. 在 Pool details（池详细信息）面板中，

   a. 在 Scaling mode（扩展模式）字段中选择 Standard (self managed) [标准（自行管理）]。

   b. 在 IP pool name（IP 池名称）字段中输入 IP 池的名称。

      > **Note**
      >
      > IP 池名称必须具有唯一性，并且不能与账户中的托管式 IP 池名称重复。

   c. （可选）如果您有想要添加到此 IP 池的现有标准专用 IP 地址，请从 Dedicated IP addresses（专用 IP 地址）字段的下拉列表中选择这些地址。

> **Note**
>
> 如果选择已与 IP 池关联的 IP 地址，则该地址现在将仅与此 IP 池关联。

6. （可选）您可以通过从 Configuration sets（配置集）字段的下拉列表中选择一个配置集，将此 IP 池与该配置集相关联。

> **Note**
>
> - 如果您选择已与 IP 池关联的配置集，则该配置集现在将仅与此 IP 池相关联。
> - 要在创建此 IP 池后添加或删除关联的配置集，请编辑配置集的 Sending IP pool（发送 IP 池）参数。
> - 如果尚未创建任何配置集，请参阅 配置集。

7. （可选）您可以通过包含标签键和该键的可选值，向此 IP 池添加一个或多个标签。

   a. 选择添加新标签，然后输入键。您还可以为标签添加可选值。

   b. 要添加标签，请选择保存更改。

   最多可以添加 50 个标签。您可以选择删除以删除任何标签。

8. 选择 Create pool（创建池）。

> **Note**
>
> 创建标准 IP 池后，可选择将其转换为托管式 IP 池。请参阅 创建托管式 IP 池。

Amazon SES 的专用 IP 地址（托管式）

专用 IP 地址（托管式）是 Amazon SES 的一项功能，它代表您自动设置和管理专用 IP 地址，从而为开始使用由 SES 管理的专用 IP 地址提供一种快速简便的方法。这有助于确保您的专用 IP 地址以高效、最佳的方式用于发送电子邮件。

要在您的账户中启用专用 IPs（托管）功能，您只需创建一个托管 IP 池，剩下的就交给 SES。SES 将根据您的发送模式确定 IPs 您需要多少专属发送模式，为您创建这些模式，然后根据您的发送要求管理它们的扩展方式。

启用后，您可以在电子邮件发送中使用专用 IPs （托管）功能，方法是将托管 IP 池与配置集相关联，然后在发送电子邮件时指定该配置集。也可以使用原定设置配置集将配置集应用于发送身份。

专用 IPs （托管）的优势和功能

您使用专用 IPs （托管）创建的专用 IP 地址可以自动执行管理任务，以帮助确保您的专用 IP 地址的使用方式最适合您发送电子邮件的方式：

- 轻松上手 — 要开始使用专用 IPs （托管），您可以直接从 SES 控制台创建托管 IP 池。专用 IP 地址会自动分配到池中。您无需通过 Su AWS pport Center 提交请求案例，即可开始使用托管 IP 池进行发送。
- 每个 ISP 自动扩展 - 您无需手动监控或扩展您的专用 IP 池，因为托管式 IP 池会根据使用情况自动扩展。它还考虑了特定于 ISP 的策略。例如，如果 SES 检测到 ISP 支持较低的每日发送限额，则池会进行扩展，以便通过更多 IP 地址更好地向该 ISP 分配流量。
- 智能预热 — 专用 IPs （托管）开始 ISPs 根据其容量向其发送邮件。也就是说，根据它们目前的预热程度。它们会自动单独跟踪每个 ISP 的预热程度。此外，专用 IPs （托管）功能以有效的每日费率提供有关您的声誉的信息，其中最重要 ISPs 的是Amazon CloudWatch 指标和内置控制面板。
  - 每个 ISP 预热 — SES 分别跟踪每个 ISP 的托管式 IP 池中每个 IP 的声誉。例如，如果您一直将所有流量发送到 Gmail，则只有 Gmail 的 IP 地址被视为已预热，其他 ISPs的 IP 地址会被视为已预热。如果您通过增加发送到 Hotmail 的电子邮件来改变流量模式，则 SES 会缓慢增加 Hotmail 的流量，因为 IP 地址尚未预热。
  - 自适应预热和共享池转换 - 预热调整是自适应的，并考虑了实际的发送模式。当发送给 ISP 的量下降时，该 ISP 的预热百分比也会下降。在预热的早期阶段，任何基于当前预热级别而言过多的发送都将通过与其他 Amazon SES 用户共享的 IP 地址（SES 共享池）发送。在预热的后期阶段，任何过多的发送都会主动减慢速度，稍后再试。

> **Important**
>
> 虽然专用 IPs （托管）会自动预热您的专用 IP 地址，但该自动过程的一部分是与 SES 共享 IP 池交互式协作。
>
> - 如果你的发送速率对你的新专用服务器来说过于激进 IPs ，SES 会自动将你发送的部分内容溢出到 SES 共享 IP 池中，以保护你的新专用 IPs服务器的声誉。
> - 即使在您的新专用 IPs 设备完全预热之后，也不能保证您的所有发送都将在 100% 的时间内通过它们。例如，如果您的发送速率突然上升，并且专用 IPs （托管）决定必须分配额外的专用 IP 地址，则它将启动预热过程，包括使用共享池。同样，如果您的发送速

率突然降至非常低，则您的所有发送都可能切换到 SES 共享 IP 池，请参阅the section called “预热的重要性”。

- 自动请求和放弃专用 IP 地址 — 使用专用（标准）时，您无需像使用专用（标准）时那样通过 AWS 支持中心请求或放弃托管专用 IPs IP 地址。直接通过 SES 控制台、CLI 或 API 使用专用 IPs（托管）进行登录时，系统会自动为您分配专用 IP 地址，并根据您发送的消息量收取费用。当您删除由专用 IPs（托管）创建的 IP 池或选择退出专用 IPs（托管）的 IP 池时，您分配的 IP 地址将自动放弃并立即停止收费。
  - 获取您的第一个专用 IP 地址 — 当您的发送量在几天内达到数百封电子邮件时，专用 IPs（托管）功能将自动分配您的第一个专用 IP 地址。这样可以确保您的发件 IP 可以建立发送信誉并提高送达率。（如果您预计发送量不会达到这个水平，则应使用共享 IP 地址。请参阅专用 IP 地址 中的比较表，查看最适合您的电子邮件发送方式的 IP 地址类型。）

为什么正确的 IP 预热很重要

为确保您的电子邮件将通过您的专用 IP 地址发送，它必须在收件人 ISP 中享有良好的声誉。 ISPs 只会接受来自他们无法识别的 IP 的少量电子邮件。当您首次获得分配的 IP 时，该 IP 是新 IP，接收 ISP 无法识别，因为它没有与之相关的声誉。为了建立 IP 声誉，它必须逐步与接收 ISP 建立信任 - 这种逐步建立信任的过程被称为预热。专用 IPs（托管）分配 IP 后，它会立即启动智能预热过程。

借助每个 ISP 的预热和专用 IPs（托管）的自适应预热功能，通过确保电子邮件的传送，可以在整个预热周期中保持业务连续性。预热阶段完成后，任何多余的容量都将排队并仅通过专用 IP 池发送。但是，如果您有一个专用 IP 地址，并且您的发送量低于维护 IP 信誉所需的最低音量，则专用 IPs（托管）可能会删除您的专用 IP，您的发送将通过 SES 共享 IP 池路由。

> **Note**
>
> 如果您发送少量电子邮件（几天内每天发送少于几百封），则通过 SES 共享 IP 池发送会更有益。查看中的比较表，查看专用 IPs（托管）是否适合您的邮件发送方式专用 IP 地址。

了解您和 SES 在使用专用 IPs（托管）时的共同责任

虽然专用 IP 地址（托管）为专用 IP 管理、扩展和预热提供了许多自动化功能，但需要了解这种自动化的程度和 SES 的职责。假设 “托管” 意味着 SES 可以完全处理 IP 信誉和列表问题的各个方面，这是不正确的。为了澄清这些误解，我们需要强调的是，虽然该服务可以自动执行扩展和预热等技术方面，但您仍有责任维护发送声誉并管理任何与声誉相关的问题，例如被列入信誉屏蔽列表 (RBL)。

以下内容 FAQs 阐明了您与SES之间的责任共担模式，并解决了对服务范围的常见误解。这些常见问题解答强调，尽管 “托管” 方面指的是技术基础设施管理，但您仍必须积极监控和维护您的发送声誉，保持较低的退回率，并自己处理大多数 RBL 下架请求：

- the section called “托管 DIPS FAQs”

创建托管 IP 池以启用专用 IPs（托管）

要启用专用 IPs（托管），请先创建一个托管 IP 池。创建托管池后，该功能会根据您的发送模式确定 IPs 您需要多少专用池，并将根据您的要求动态扩展这些存储池。

要使用托管池发送电子邮件，必须将托管池与配置集相关联，然后在发送电子邮件时指定该配置集。也可以使用原定设置配置集将配置集应用于发送身份。

创建托管式 IP 池的方法有两种：

- 创建新的池。
- 将现有池从标准池转换为托管池。

以下过程为两种方法都提供了说明。

使用 SES 控制台创建或转换为托管式 IP 池

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，选择专用 IPs。
3. 根据您是要创建新的托管式 IP 池还是要将标准专用 IP 池转换为托管池，请按照相应的说明进行操作：

   Create new pool

   创建新的托管式 IP 池

   1. 请执行以下操作之一：

      a. 如果您的账户 IPs 中没有专属账户：

      - 将显示专用 IPs入职页面。在专用 IPs（托管）概览面板中，选择启用专用 IPs。

        Create IP Pool（创建 IP 池）页面将打开。

b. 如果您的账户 IPs 中已有专用账户：

i. 在 “专用” IPs 页面上选择 “托管 IP 池” 选项卡。

ii. 在 All Dedicated IP (managed) pools [所有专用（托管式）IP 池] 面板中，选择 Create Managed IP pool（创建托管式 IP 池）。

Create IP Pool（创建 IP 池）页面将打开。

2. 在 Pool details（池详细信息）面板中，

a. 在 Scaling mode（扩展模式）字段中，选择 Managed (auto managed) [托管式（自动管理）]。

b. 在 IP pool name（IP 池名称）字段中输入托管式池的名称。

> **Note**
>
> - IP 池名称必须具有唯一性。它不能与您账户中的标准专用 IP 池名称重复。
> - 您账户中的每个 AWS 区域 的专用 IP 池不能超过 50 个，包括托管式 IP 池和标准 IP 池。

3. （可选）您可以通过从 Configuration sets（配置集）字段的下拉列表中选择一个配置集，将此托管式 IP 池与该配置集相关联。

> **Note**
>
> - 如果您选择已与某个 IP 池关联的配置集，则该配置集将与该托管式池相关联，而不再与以前的池相关联。
> - 要在创建此托管式池后添加或删除关联的配置集，请在 General details（常规详细信息）面板中编辑配置集的 Sending IP pool（发送 IP 池）参数。
> - 如果尚未创建任何配置集，请参阅 配置集。

4. （可选）您可以通过包含标签键和该键的可选值，向 IP 池添加一个或多个标签：

a. 选择添加新标签，然后输入键。您还可以为标签添加可选值。您最多可以添加 50 个标签，如果出错，请选择 Remove（删除）。

b. 要添加标签，请选择 Save changes（保存更改）。

创建池后，您可以通过选择托管池并选择 Edit（编辑），来添加、删除或编辑标签。

5. 选择 Create pool（创建池）。

> **Note**
>
> - 创建托管式 IP 池后，无法将其转换为标准 IP 池。
> - 使用专用 IPs（托管）时，每个 AWS 区域 账户中的发送身份（域名和电子邮件地址，任意组合）不能超过 10,000 个。

Convert standard to managed

将标准专用 IP 池转换为托管池

1. 在 “专用” IPs 页面上选择 “标准 IP 池” 选项卡。
2. 在所有专用 IP（标准）池面板中，选中要将其从标准池转换为托管池的专用 IP 池的复选框。
3. 选择转换为托管池 – 阅读转换为托管式 IP 池对话框中的内容，确认您了解将标准专用 IP 池转换为托管池的条件。

> **Note**
>
> 在将专用 IP 池从标准池转换为托管池之前，请注意以下几点：
>
> 1. 您当前所有的专用 IPs（标准）资源池都将移至托管池。
> 2. 如果您目前为发送量租用了太多的专用 IPs（标准），则专用 IPs（托管）将删除多余的内容 IPs。
> 3. 如果您的任何专用 IPs（标准）应用程序属于其他应用程序的允许名单，则不应将其转移到托管池，因为如果它们变得多余，它们将被删除，请参阅第 2 点。
> 4. 将不再按 IP 向您收费，而是根据您使用托管池的发送量向您收费。请参阅 Amazon SES 定价。

4. 如果您同意陈述的条件，请选择确认，此时会出现一条横幅，确认您的标准专用 IP 池已转换为托管池。

> **Note**
>
> 现在，您在转换之前与标准池关联的任何配置集或标签都将与托管池相关联，为使用该配置集发送的任何电子邮件提供无缝过渡。

可以使用事件发布功能来跟踪托管池的发送性能。有关更多信息，请参阅 the section called “使用事件发布监控电子邮件发送”。

在 Amazon SES 控制台中查看托管式 IP 池的发送情况和容量

对于您创建的托管式 IP 池，SES 控制台通过使用显示发送指标以及 ISP 利用率和容量的卡片和时间序列图表，为您提供了一种简单的方法来观察它们用于电子邮件发送的情况。

使用 SES 控制台查看托管式 IP 池的发送情况和容量

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，选择专用 IPs。
3. 在 “专用” IPs 页面上选择 “托管 IP 池” 选项卡。
4. 根据您是要在 Amazon SES 控制台还是 Amazon 控制台中查看发送和容量指标，请按照相应的说明进行操作： CloudWatch

   Amazon SES console

   在 Amazon SES 控制台中查看发送和容量指标

   1. 在所有专用 IP（托管）池表中，选择 IP 池列中列出的托管式 IP 池的名称，以查看其详细信息。

      所选 IP 池的详细信息页面将打开，并显示以下卡片和时间序列图表：

      a. 卡片：

      - 发送状态 — IPs 通过显示以下两种状态之一来指示您的发送量和频率是否足以使用专用：
        - 发送量不足 – 您的发送量太低。
        - 通过专用池发送 IPs-您的托管池中 IPs正在使用一个或多个专用池。

- 托管专用 IP 发送量-过去 7 天内通过托管池 IPs 中的专用地址发送的电子邮件量。
- 托管专用 IP 发送百分比-过去 7 天内通过托管池 IPs 中的专用专用 IP 发送的电子邮件的百分比。

b. 图表：

- 发送量-与共享电子邮件相比，在过去 7 天内通过托管专用 IPs 发送的电子邮件量 IPs。
- 发送量百分比-在过去 7 天内通过托管专用 IPs 服务器发送的电子邮件与共享电子邮件的百分比 IPs。
- ISP 容量 — 显示根据最广泛使用的前 10 个托管池 IPs 中的专用电子邮件发送量，ISPs 以及它们在发送期间的可用容量：
  - ISP 发送量（红色条形）– 您在过去 24 小时内通过所选 ISP 发送的电子邮件数量。
  - ISP 容量（蓝线）– 所选 ISP 在过去 24 小时内的可用容量。

2. 要筛选特定 ISP 的 ISP 容量图，请选择 ISP 列表框并选择一个 ISP，图表将使用所选 ISP 的指标进行更新。（如果您不筛选 ISP，则原定设置情况下会显示 Gmail）。

Amazon CloudWatch console

在 Amazon CloudWatch 控制台中查看发送和容量指标

- 在所有专用 IP（托管）池表中，选择CloudWatch <pool_name>指标列中的查看CloudWatch指标链接以查看其详细信息。

  所选 IP 池的页面将在 CloudWatch 控制台中打开，显示以下指标：

  - 发送-通过托管专用 IPs 和共享方式发送的电子邮件量 IPs。
  - ApproximateDedicatedSendingPercentage— 表示通过专用 IP 传输的流量的大致百分比。
  - SentLast24 小时-您在过去 24 小时内通过所选 ISP 发送的电子邮件量。（SES 控制台中标为 ISP 发送量。）
  - 可用24 HourSend-所选 ISP 在过去 24 小时内的可用容量。（SES 控制台中标为 ISP 容量。）

删除托管 IP 池并选择退出专用 IPs （托管）

当您删除托管式 IP 池时，其所有分配的 IP 地址都将自动释放。如果您只有一个托管 IP 池并将其删除，或者您删除了最后剩下的托管 IP 池，则您将选择退出专用 IPs （托管）功能，费用将立即停止。

要使用 SES 控制台删除托管式 IP 池，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，选择专用 IPs。
3. 在 “专用” IPs 页面上选择 “托管 IP 池” 选项卡。
4. 在 All Dedicated IP (managed) pools [所有专用 IP（托管式）池] 表中，选择要删除的托管池的 IP pool（IP 池）名称旁边的单选按钮，然后选择 Delete（删除）。
5. 在弹出模式中，您将有机会通过选择 Delete（删除）来确认您的选择，或者选择 Cancel（取消）以保留您的托管池。

> **Note**
>
> 如果您只有一个托管池，或者您要删除最后一个托管池，则弹出式模式会提醒您，删除剩余的托管池后，您将选择退出专用 IPs （托管）功能，并且将不再为此付费。在选择 Delete（删除）之前，需要在确认字段中输入 *Disable*。

通过 Amazon SES 使用您自己的 IP 地址发送电子邮件

Amazon SES 包括一个名为自带 IP 地址（BYOIP）的功能，让您可以通过 Amazon SES 使用自己的 IP 地址发送电子邮件。如果您已经使用了某个 IP 地址范围来发送电子邮件，您可以请求将您的 IP 范围设为可通过 Amazon SES 发送电子邮件。

> **Note**
>
> BYOIP 仅适用于您手动配置的专用 IP 地址，不能与专用（托管）一起使用。 IPs

在一些情况下 BYOIP 很有用，例如，当您使用内部电子邮件发送系统建立了良好的 IP 声誉但希望迁移到 Amazon SES 时。通过 BYOIP，您可以立即开始使用 Amazon SES 发送电子邮件，而无需重新建立 IP 地址的声誉。

要求

要使用 BYOIP，您的 IP 地址范围必须满足以下要求：

- 必须在区域 Internet 注册表 (RIR) 中注册该地址范围，例如 American Registry for Internet Numbers (ARIN)、Réseaux IP Européens Network Coordination Centre (RIPE NCC) 或 Asia-Pacific Network Information Centre (APNIC)。该地址范围必须由企业或机构实体注册，而不能由个人注册。
- 您还必须通过提交签名的授权消息，证明您拥有该地址范围。
- 该 IP 地址范围中的地址必须具有干净的历史记录。我们可能会调查 IP 地址范围的声誉，并保留在其中包含的 IP 地址具有不良声誉或与恶意行为关联的情况下拒绝此 IP 地址范围的权利。
- IP 地址范围不能包括为 BYOIP 引入另一个 AWS 服务 IP 地址范围，例如 Amazon。 EC2

注意事项

在请求将 IP 范围转移到 Amazon SES 之前，您应该考虑以下几个因素：

- 您可指定的最具体的地址范围为 /24。换句话说，如果您将 IP 范围 203.0.113.0/24 传输到您的 Amazon SES 账户，您就可以从共计 256 个地址（范围从 203.0.113.0 到 203.0.113.255）发送邮件。您必须传输整个范围，Amazon SES 目前不允许传输单个 IP 地址。
- 如果您为特定 IP 地址范围使用 BYOIP，则只能从单个 AWS 区域访问该范围。
- 您可以将每个区域的 5 个地址范围添加到您的 AWS 账户中。
- 如果您使用自己的 IP 地址，则无法使用共享 Amazon SES IP 地址池中的地址。如果您需要使用这些共享 IP 地址，可以在其他地址中使用 Amazon SES AWS 区域，也可以创建一个新的 IP 地址 AWS 账户。
- 您用于 BYOIP 的每个 IP 地址每个月会产生费用。有关更多信息，请参阅 Amazon SES 定价。

通过 Amazon SES 使用自带 IP 地址

为了防止我们的系统被用于发送未经请求或恶意的内容，我们必须仔细考虑每个 BYOIP 请求。

如果你想在 Amazon SES 上使用自己的 IP 范围，请将以下信息发送到 ses-byoip-request@amazon .com：

- 您的 AWS 账户 ID。
- 你想 AWS 区域 在其中使用 IP 范围的，比如 ap-south-1。
- 您的使用案例的描述。

- 您要用于 Amazon SES 的 IP 范围。
- 该范围注册到的 Internet 注册表的名称。

我们将在工作时间的 48 个小时内回复您的请求。在与您沟通的过程中，我们可能会要求您提供其他信息，包括证明您拥有 IP 范围的文件。

Virtual Deliverability Manager for Amazon SES

可送达性（即确保您的电子邮件到达收件人的收件箱而不是垃圾邮件或垃圾文件夹）是成功的电子邮件策略的核心要素。

Virtual Deliverability Manager 是 Amazon SES 的一项功能，通过提供对发送和传递数据的洞察信息，并就如何解决对送达成功率和声誉产生负面影响的问题提供相关建议，来帮助您提高电子邮件送达率，比如提高收件箱到达率和电子邮件转换率。

为什么收件箱到达率和发件人声誉很重要

收件箱到达率是电子邮件转换（收件人在打开电子邮件后执行操作时）的关键因素 — 未收到您的邮件的客户将无法看到该邮件，更不用说与他们互动了。

在客户体验层面，发送声誉对收件箱到达率的影响最大，它决定了不想要的邮件是否会到达收件人的邮箱，或者所需邮件在有机会到达收件人邮箱之前是否会被路由到垃圾邮件文件夹或被屏蔽。

Virtual Deliverability Manager 如何帮助提高送达率和声誉

借助控制面板，Virtual Deliverability Manager 可帮助您提高送达率和声誉，控制面板能够提供账户电子邮件程序的概览视图和详细视图，可以帮助您专注于任何有问题的领域，并通过 Advisor 提供解决方案，以修复对电子邮件送达率和声誉产生不利影响的基础架构问题。

- 控制面板 – 提供对送达数据的洞察信息，重点关注账户、ISP、发送身份和配置集级别。这可以帮助您快速查看有问题的领域和趋势，并在这些问题演变为临时拒绝（延期）或阻止等更大的送达问题之前解决它们。这些洞察信息还将通过计算理想的时间和日期来帮助您提高发件人的声誉，从而提高电子邮件营销活动的客户参与度和转化率。
- Advisor – 通过标记对您的电子邮件送达率和声誉产生负面影响的配置问题，提供可提高电子邮件送达率的建议。它将推荐相关解决方案，以解决发送域、IP 空间和身份验证记录基础架构中的特定问题，例如当 SPF、DMARC 或 DKIM 记录不存在时，或者 DKIM 密钥长度太短时。

开始使用 Virtual Deliverability Manager

要开始使用 Virtual Deliverability Manager，Amazon SES 控制台中的入门向导将引导您完成为账户启用 Virtual Deliverability Manager 的步骤。请参阅the section called “入门”。

主题

- Virtual Deliverability Manager advisor
- Virtual Deliverability Manager 设置

开始使用 Virtual Deliverability Manager

要开始在您的账户中使用 Virtual Deliverability Manager，必须使用 Amazon SES 控制台中的注册向导将其启用，您可以在其中设置互动跟踪和优化共享送达。Virtual Deliverability Manager 使用互动跟踪和优化共享送达功能来监控您的发送情况，并帮助您提高送达率和声誉。

- 互动跟踪 – 借助该功能，您可以通过使用封装链接中的跟踪像素，通过打开和点击事件监控收件人互动行为。触发后，跟踪像素会提供消息打开时的时间戳，并指明收件人点击了哪些链接。启用此功能会更改您的 URLs 和链接，使其包含 Amazon SES 参与度跟踪包装。
- 优化共享送达 - 自动选择要在发送电子邮件时使用的最佳 IP，从而提高送达至目标电子邮件收件人的邮件的端点送达能力。这不适用于专用 IP 地址。

虽然在注册向导中，互动跟踪和优化共享送达功能在默认情况下均处于启用状态，但您可以选择将其关闭。强烈建议您保持启用这两项功能，以充分利用虚拟可交付性管理器。

使用 Amazon SES 控制台开始体验 Virtual Deliverability Manager

以下过程演示了如何使用 Amazon SES 控制台开始体验 Virtual Deliverability Manager。

要使用 Amazon SES 控制台开始体验 Virtual Deliverability Manager，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，选择 Virtual Deliverability Manager。
3. 在 Virtual Deliverability Manager overview（Virtual Deliverability Manager 概述）页面上选择任意 Get started with Virtual Deliverability Manager（开始使用 Virtual Deliverability Manager）按钮。
4. 在 Select Engagement tracking（选择互动跟踪）页面上，接受默认设置或选择 Turn off engagement tracking（关闭互动跟踪），然后选择 Next（下一步）。

> **Note**
>
> 开启互动跟踪功能会更改您的 URLs 和链接，使其包含 Amazon SES 参与度跟踪包装器。

5. 在 Select Optimized shared delivery（选择优化共享送达）页面上，接受默认设置或选择 Turn off optimized shared delivery（关闭优化共享送达），然后选择 Next（下一步）。

   > **Important**
   >
   > 优化共享送达可能会导致预先延迟发送电子邮件，以保护您的发送声誉。要确保必须立即发送某个关键工作负载，建议您不要启用此设置。相反，应使用配置集进行发送，并且仅在您可以承受延迟的情况下为相关配置集启用优化共享送达。

6. 可在 Review and enable（查看和启用）页面上查看您的互动跟踪和优化共享送达功能选择情况。如果要返回并进行更改，请选择 Previous（上一步）；否则，请选择 Enable Virtual Deliverability Manager（启用 Virtual Deliverability Manager）。

   此时将打开 Virtual Deliverability Manager settings（Virtual Deliverability Manager 设置）页面。Subscription overview（订阅概览）面板显示 Virtual Deliverability Manager 的状态，Additional settings（其他设置）面板显示 Engagement tracking（互动跟踪）和 Optimized shared delivery（优化共享送达）设置的状态。

为您的账户启用 Virtual Deliverability Manager 后，您可以通过覆盖自定义设置在 Virtual Deliverability Manager 中的定义方式，来定义这些自定义设置，以确定配置集将如何使用互动跟踪和优化共享送达。这使您可以灵活地为特定的电子邮件营销活动量身定制电子邮件发送方式。例如，您可以为营销电子邮件启用互动跟踪和优化共享送达功能，并对事务性电子邮件禁用它们。创建或编辑配置集时，请参阅 Virtual Deliverability Manager 选项。

开始使用虚拟可交付性管理器使用 AWS CLI

以下示例演示了如何使用 AWS CLI开始体验 Virtual Deliverability Manager。

要开始使用虚拟可交付性管理器，请使用 AWS CLI

您可以使用 Amazon SES API v2 中的 `PutAccountVdmAttributes` 操作开始使用 Virtual Deliverability Manager。您可以从调用此操作 AWS CLI，如以下示例所示。

- 在您的账户中启用 Virtual Deliverability Manager：

```
aws --region us-east-1 sesv2 put-account-vdm-attributes --vdm-attributes
 VdmEnabled=ENABLED
```

- 使用输入文件启用互动跟踪和优化共享送达：

```
aws --region us-east-1 sesv2 put-account-vdm-attributes --cli-input-json file://
attributes.json
```

输入文件如下所示：

```
{
    "VdmAttributes": {
        "VdmEnabled": "ENABLED",
        "DashboardAttributes": {
            "EngagementMetrics": "ENABLED"
        },
        "GuardianAttributes": {
            "OptimizedSharedDelivery": "ENABLED"
        }
    }
}
```

通过从 Amazon SES API v2 参考中的 VdmAttributes 数据类型进行链接，可以查找参数值和相关数据类型。

> **Note**
>
> 开启互动跟踪功能会更改您的 URLs 和链接，使其包含 Amazon SES 参与度跟踪包装器。

> **Important**
>
> 优化共享送达可能会导致预先延迟发送电子邮件，以保护您的发送声誉。要确保必须立即发送某个关键工作负载，建议您不要启用此设置。相反，应使用配置集进行发送，并且仅在您可以承受延迟的情况下为相关配置集启用优化共享送达。

- 要验证结果，请执行以下操作：

```
aws --region us-east-1 sesv2 get-account
```

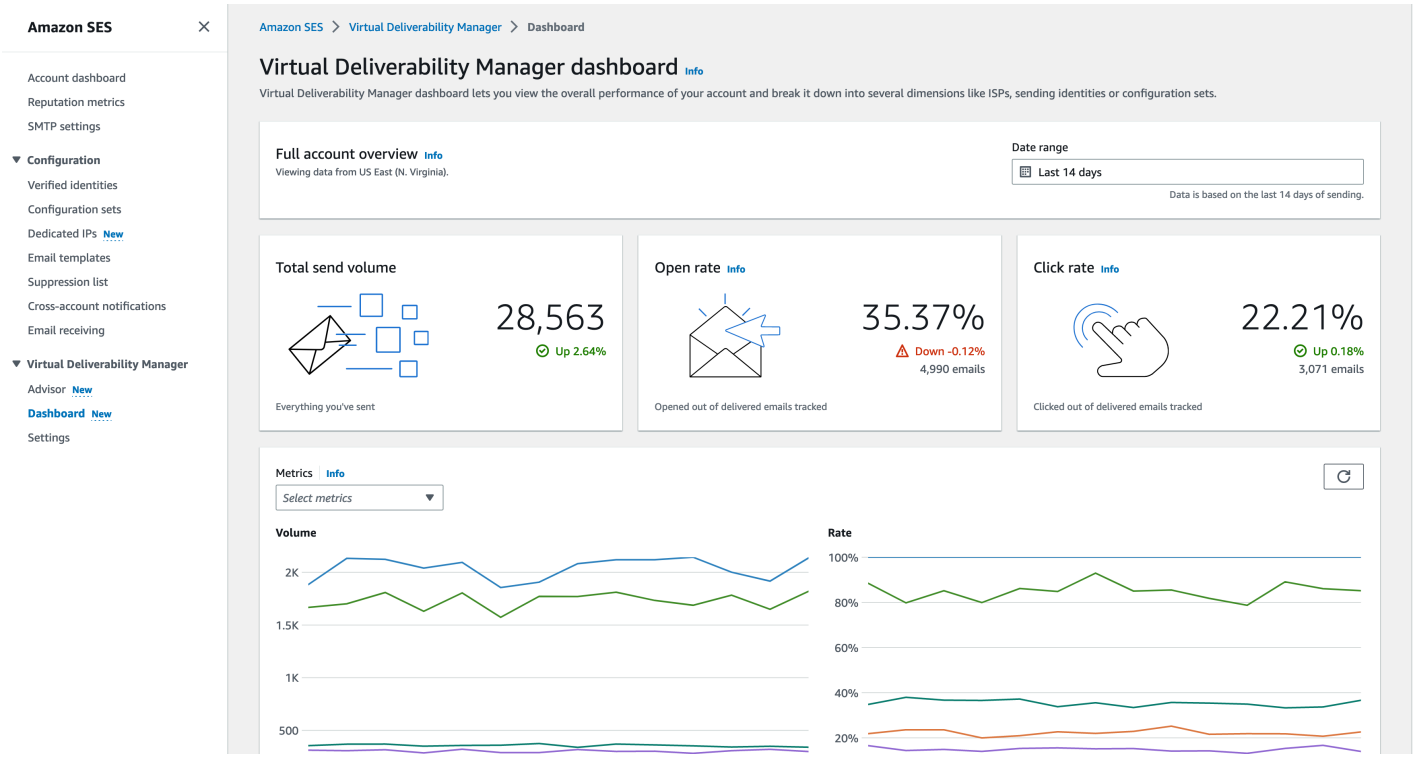
- 要通过覆盖 Virtual Deliverability Manager 中的定义方式，为配置集如何使用互动跟踪和优化的共享交付来定义自定义设置，请参阅中的 AWS CLI 示例。the section called “设置”

Virtual Deliverability Manager 控制面板

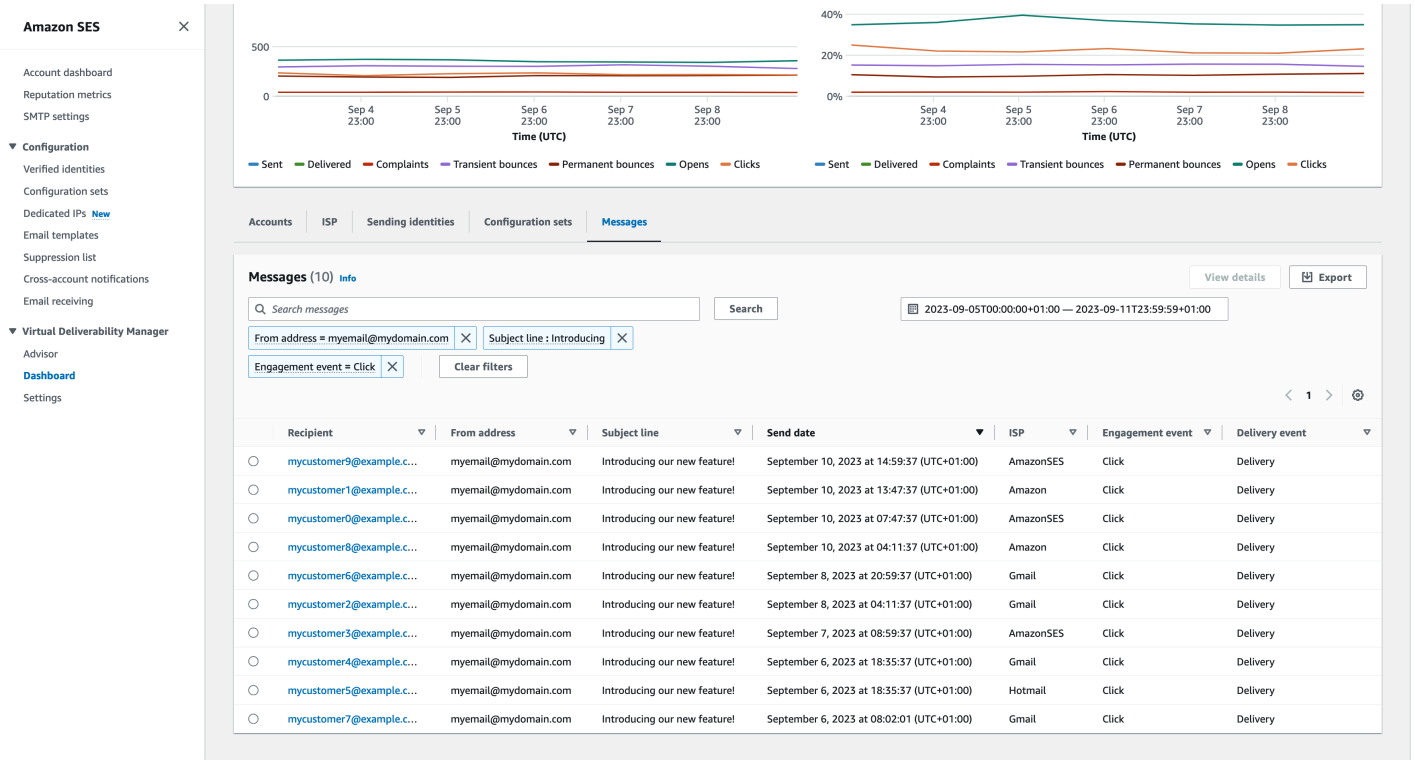
控制面板提供您账户可交付性计划的高级视图，例如易于阅读的卡片和通过open/click and delivery rates and bounce/complaint统计数据显示可交付性和声誉的时间序列图。控制面板还提供更详细的视图，使您能够在出现与特定 ISP、发送身份或与电子邮件营销活动相关的配置集有关的问题时，深入查看更详细的具体表格数据。

能够从较高的总体水平查看信息，同时还能够了解具体的细节，这可以让您专注于送达问题，而不必整体查看电子邮件程序。这种洞察力还使您能够在趋势和可能的问题演变为更大的送达问题（例如延期或阻止）之前解决这些问题。

虚拟可交付性管理器控制面板中的账户概览，其中显示了卡片和时间序列图表。



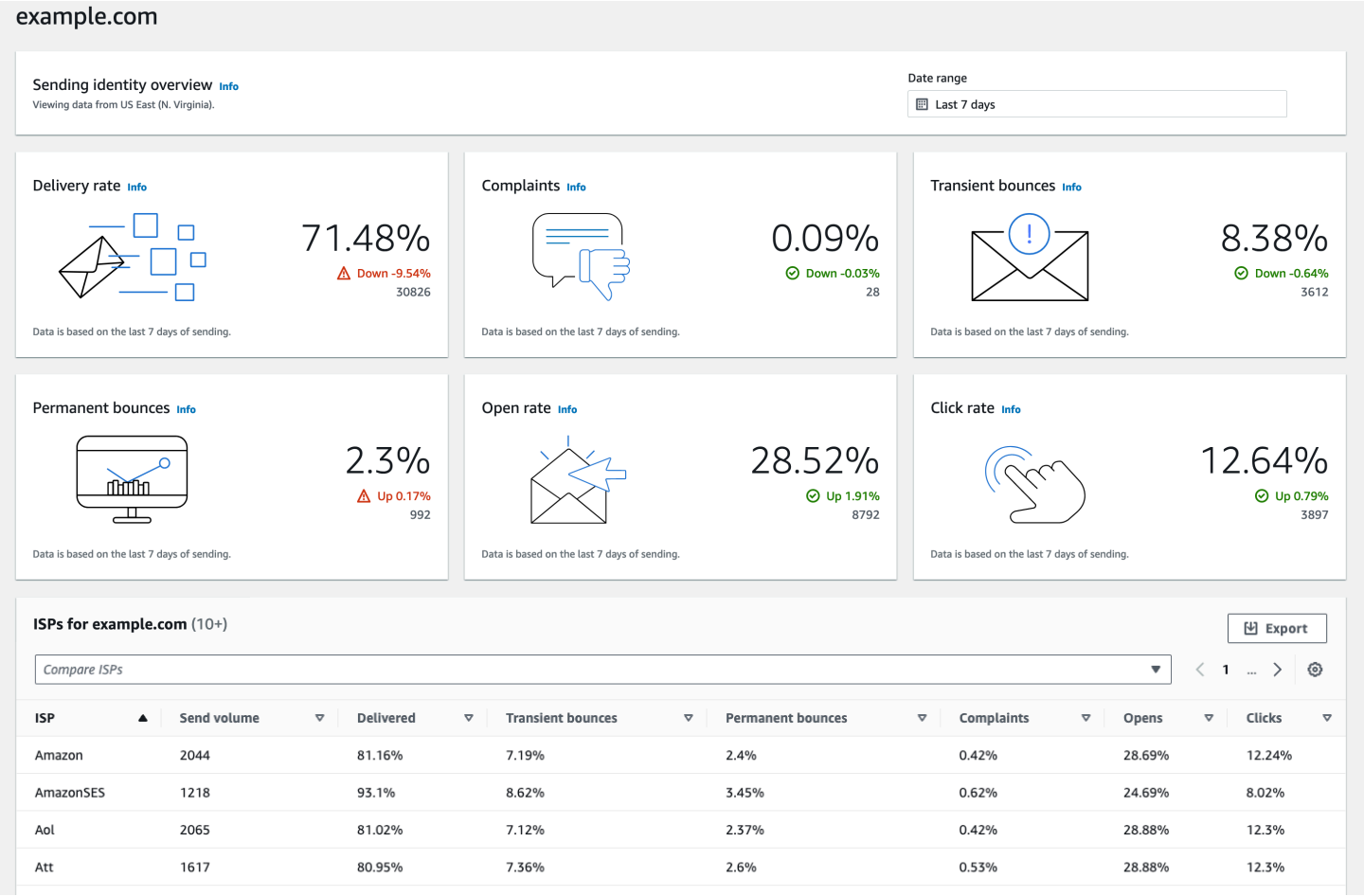
在虚拟可交付性管理器控制面板中选择的邮件表，显示与日期范围和筛选条件匹配的已发送邮件。



控制面板提供的精细数据可以帮助您提高发件人声誉并计算理想的时间和日期，从而提高电子邮件程序的参与度和转化率，使您能够深入了解特定数据集：

- ISP 数据 – 当您遇到针对特定 ISP 或邮箱提供商的送达问题时很有用—不要尝试调整整个账户，这可能不起作用，您可以将注意力集中在有问题的端点上，并遵循其最佳实践来提高发件人对该 ISP 的声誉以及恢复良好的收件箱送达率，使邮件送达至收件人。了解您的 ISP 分布情况也很重要，因为您可能会向某个 ISP 或邮箱提供商发送比其他 ISP 或邮箱提供商更多的邮件。您需要确保流量始终由最终收件人传送和参与，以对您的电子邮件转换产生积极影响。
- 发送身份和配置集数据 – 有助于您识别导致整体账户送达问题的发送身份和配置集。您可以专门关注这些问题，调整配置，并减少使用特定身份发送邮件，直到问题得到解决。例如，发送身份导致将邮件意外发送到黑名单，从而导致所有流量都通过该身份。该身份与配置集相关联，进而导致送达问题。在这种情况下，能够识别发送身份或配置集很有用，这样您就可以专注于具体地纠正这个问题，而不是梳理整个账户来尝试找出送达问题的根本原因。

显示在虚拟可交付性管理器控制面板中的有关选定发送身份（example.com）的向下钻取数据 – 卡片显示了送达率和声誉指标。该表显示了 ISPs 所有发送身份向其发送的邮件，以及输入日期范围内每个 ISP 的公制费率。



在 Amazon SES 控制台中使用 Virtual Deliverability Manager 控制面板

以下过程演示了如何在 Amazon SES 控制台中使用 Virtual Deliverability Manager 控制面板来查看您的整体送达率和声誉统计数据，并深入了解有问题的领域。

要使用 Virtual Deliverability Manager 控制面板大致了解和详细查看您的账户的送达指标数据，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，在 Virtual Deliverability Manager 下选择 Dashboard（控制面板）。

> **Note**
>
> - 如果您尚未为您的账户启用 Virtual Deliverability Manager，将看不到 Dashboard（控制面板）。有关更多信息，请参阅 the section called “入门”。

- 控制面板指标以接近实时的方式显示。
- 控制面板消息在发送后几分钟内即可显示。

3. 在完整账户概述面板中，选择用于卡片、时间序列图表和向下钻取表中所有指标的日期范围。

   - 在日期范围字段中，选择相对范围（默认）或绝对范围。
     - Relative range（相对范围）- 选择与所需天数相对应的单选按钮。
       - 自定义范围 - 输入以天（最多 60 天）、周（最多 8 周）或月（最多 2 个月）为单位的范围。
     - 绝对范围 - 您选择的第一个日期将是开始日期，第二个日期将是结束日期，总共不超过 60 天。要指定某一天，请同时选择该日期作为开始和结束日期。

   > **Note**
   >
   > 以下内容适用于控制面板中的所有日期范围：
   >
   > - 所有日期和时间均为世界标准时间。
   > - 对于Relative range（相对范围）日期，最后一天以其 UTC 午夜时间戳结束。例如，如果您选择了 Last 7 days（过去 7 天），则第七天将是昨天，在午夜结束。
   > - 如果日期范围大于 30 天，则账户统计数据表中的差异百分比列和卡片中的变化百分比将没有值（用破折号 - 表示）。

4. 卡片、时间序列图表和所有向下钻取表、账户统计信息、ISP、发送身份和配置集均显示根据输入的日期范围计算得出的指标总计，并使用如何计算控制面板指标中所述的指标数学运算。

   - 要创建您当前在 ISP、发送身份或配置集表中查看的数据的本地 `.csv` 文件，请选择其导出按钮。

5. 指标窗格中显示了时间序列图形，这些图形显示您输入的日期范围内的量和比率进展情况。将鼠标悬停在图形中的日期间隔上，将显示基于每日聚合的确切量计数或比率百分比。您可以使用选择指标下拉列表筛选想要查看的指标。

6. 选择 Accounts（账户）选项卡以显示 Accounts statistics（账户统计数据）表。

   - 此表概述了您的送达率和声誉指标，显示了针对根据输入的日期范围计算得出的已发送、送达、投诉、临时和永久退信以及打开和点击次数的总发送量、比率百分比和差异百分比。

> **Note**
>
> 如果日期范围大于 30 天，则差异百分比列将没有值（用破折号 - 表示）。

7. 选择 ISP 选项卡以显示 ISP 表。

   - 此表显示了您向其发送电子邮件的每个 ISP 的发送量、送达、临时和永久退信、投诉、打开和点击次数的相关指标，这些指标是根据输入的日期范围计算得出的。
   - 要筛选特定内容 ISPs， ISPs请在比较搜索框中，为要包括的每个 ISP 选择相应的复选框。
   - 要创建您当前在此表中查看的数据的本地 `.csv` 文件，请选择其导出按钮。

8. 选择 Sending identities（发送身份）选项卡以显示 Sending identities（发送身份）表。

   - 此表显示您使用的每个发送身份的发送量、送达、临时和永久退信、投诉、打开和点击次数的相关指标，这些指标是根据输入的日期范围计算得出的。
   - 要筛选特定发送身份，请在比较身份搜索框中，选中与要包括的每个身份对应的复选框。
   - 要深入了解特定发送身份，请在 Sending identity（发送身份）列中选择其名称。
     - 将显示卡片，其中显示根据输入的日期范围计算得出的所选发送身份的送达率、投诉率、临时和永久退信、打开和点击率。
     - 时间序列图表将刷新，显示根据输入的日期范围计算出的所选发送身份的所有指标。
     - 将显示一个 ISP 表，其中列出了向其 ISPs 发送邮件的所有发送身份，并给出了根据输入的日期范围计算得出的每个 ISP 的指标。
   - 要创建您当前在此表中查看的数据的本地 `.csv` 文件，请选择其导出按钮。

9. 选择 Configuration sets（配置集）选项卡以显示 Configuration sets（配置集）表。

   - 此表显示了根据输入的日期范围计算得出的用于发送邮件的每个配置集的发送量、送达、临时和永久退信、投诉、打开和点击次数的相关指标。
   - 要筛选特定配置集，请在比较配置集搜索框中，选中与要包括的每个配置集对应的复选框。
   - 要深入了解特定配置集，请在 Configuration set（配置集）列中选择其名称。
     - 将显示卡片，其中显示根据输入的日期范围计算得出的所选配置集的送达率、投诉率、临时和永久退信、打开和点击率。
     - 时间序列图表将刷新，显示根据输入的日期范围计算出的选定配置集的所有指标。
     - 将显示一个 ISP 表，其中列出了用于向 ISPs 其发送邮件的所有配置集，并给出了根据输入的日期范围计算得出的每个 ISP 的指标。
   - 要创建您当前在此表中查看的数据的本地 `.csv` 文件，请选择其导出按钮。

10. 选择邮件选项卡以显示邮件表。

    这是一个交互式表，为您提供一种搜索和查找已发送邮件的方法。对于每封邮件，您可以跟踪其当前的传递和参与状态、事件历史记录，并查看邮箱提供商返回的回复。以下几点涵盖了您可以搜索特定邮件的方法：

    - 在日期范围选择器中进行选择，您可以筛选过去 30 天内发送的邮件。如果您未选择日期范围，则搜索将原定设置为最近 7 天，包括您所在时区的当天。
    - 在搜索邮件字段中，您可以根据收件人、发件人地址、主题行、ISP、参与事件、传递事件和邮件 ID 进行筛选 -- 以下属性适用：
      - 根据筛选条件类型，您可以输入区分大小写的文本字符串，也可以从列表中选择一个值。
      - 参与事件仅限单个值，主题行最多可以有两个值，而所有其他筛选条件对于每次搜索最多可以有五个值。根据邮件 ID 筛选将排除您可能已选择的任何其他筛选条件，包括日期范围。
      - 邮件 ID 列在默认情况下处于隐藏状态，但可以通过选择齿轮图标以自定义查看邮件表的方式来显示该列。
    - 选择筛选条件和日期范围后，选择搜索，表中将填充符合搜索条件的邮件。该表最多可以加载 100 封邮件。如果您的搜索返回的邮件超过 100 封，则表中的 100 封邮件是返回的邮件总数的随机样本。
    - 选择邮件的单选按钮，然后选择查看详细信息，将生成邮件信息侧边栏，其中包含邮件的完整事件历史记录（最新的事件历史记录位于顶部），以及邮箱提供商返回的任何回复或诊断代码的详细信息。
    - 要创建您当前在此表中查看的数据的本地 `.csv` 文件，请选择其导出按钮。

使用 AWS CLI访问 Virtual Deliverability Manager 指标数据

以下示例演示了如何使用 AWS CLI访问 Virtual Deliverability Manager 指标数据。这与在控制台的 Virtual Deliverability Manager 控制面板中使用的数据相同。

要访问您的送达率指标数据，请使用 AWS CLI

您可以使用 Amazon SES API v2 中的 `BatchGetMetricData` 操作来访问送达指标数据。您可以从 AWS CLI 调用此操作，如以下示例所示。

- 访问送达指标数据：

```
aws --region us-east-1 sesv2 batch-get-metric-data --cli-input-json file://sends.json
```

- 输入文件如下所示：

```
{
 "Queries": [
   {
     "Id": "Retrieve-Account-Sends",
     "Namespace": "VDM",
     "Metric": "SEND",
     "StartDate": "2022-11-04T00:00:00",
     "EndDate": "2022-11-05T00:00:00"
    }
 ]
}
```

通过从 Amazon SES API v2 参考中的 BatchGetMetricDataQuery 数据类型进行链接，可查找有关参数值和相关数据类型的更多信息。

使用筛选和导出您的送达率指标数据 AWS CLI

此示例向您展示如何通过 AWS CLI使用 CreateExportJob 操作来筛选送达率指标数据，并将其导出到 .csv 或 .json 文件。这与在虚拟可交付性管理器控制面板的 ISP、发送身份和配置集表中使用的数据相同。

要筛选您的送达率指标数据并将其导出到.csv 或.json 文件，请使用 AWS CLI

您可以在 Amazon SES API v2 中使用 CreateExportJob 操作以及 MetricsDataSource 数据类型来筛选指标数据并将其导出到 .csv 或 .json 文件。您可以从调用此操作 AWS CLI ，如以下示例所示。

- 使用输入文件筛选和导出送达率指标数据：

```
aws --region us-east-1 sesv2 create-export-job --cli-input-json file://metric-export-input.json
```

- 在此示例中，输入文件使用MetricsDataSource参数筛选 ISPs 您向其发送的所有邮件，显示给定日期范围内的成功投递率，以及为输出文件指定的.csv 格式：

```
{
    "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"
    }
}
```

在 Amazon SES API v2 参考中，通过 MetricsDataSource 中作为类型 ExportDataSource 的对象，可以找到有关参数值和相关数据类型的更多信息。

查找您发送的消息、其送达和参与状态，并使用导出结果 AWS CLI

这些示例向您展示了如何使用 AWS CLI通过 CreateExportJob 操作搜索和查找您发送的特定邮件，查看其当前的送达和参与状态，以及将搜索结果导出到 .csv 或 .json 文件。这与在虚拟可交付性管理器控制面板的邮件表中使用的数据相同。

要查找已发送的消息、其发送和参与状态，并将结果导出到.csv 或.json 文件，请使用 AWS CLI

您可以在 Amazon SES API v2 中使用 CreateExportJob 操作和 MessageInsightsDataSource 数据类型来应用筛选条件，以便查找您发送的特定邮件、查看其送达和参与状态，并将结果导出到 .csv 或 .json 文件。您可以从调用此操作 AWS CLI ，如以下示例所示。

> **Note**
>
> 如果您筛选后的搜索返回的邮件超过 10,000 封，则 API 结果集中的 10,000 封邮件是返回的邮件总数的随机样本。

- 使用输入文件查找已发送的邮件、查看其当前状态并导出结果：

```
aws --region us-east-1 sesv2 create-export-job --cli-input-json file://message-
insights-export-input.json
```

- 在本例中，输入文件使用 MessageInsightsDataSource 参数，根据等于“促销于今晚结束”的主题进行筛选，并为输出文件指定 .csv 格式：

```
{
    "ExportDataSource": {
        "MessageInsightsDataSource": {
            "StartDate": "2023-07-01T00:00:00",
            "EndDate": "2023-07-10T00:00:00",
            "Include": {
                "Subject": [
                    "Sale Ends Tonight!"
                ]
            }
        }
    },
    "ExportDestination": {
        "DataFormat": "CSV"
    }
}
```

- 在此示例中，输入文件使用MessageInsightsDataSource参数筛选以 “Hello” 开头的主题，该主题与 FromEmailAddress 包含 “信息” 一起发送到以 “@example .com” 结尾的目的地，并为输出文件指定.json 格式：

```
{
    "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"
    }
}
```

- 在此示例中，输入文件使用MessageInsightsDataSource参数筛选以 “Hello” 开头的主题，排除以 “noreply@example.com” 为 FromEmailAddress输出文件指定.csv 格式的结果：

```
{
    "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"
    }
}
```

- 在此示例中，输入文件使用MessageInsightsDataSource参数筛选以 “Hello” 开头的主题、向以 “@example .com” 结尾的目的地发送的 FromEmailAddress 包含 “信息”、使用 Gmail 作为 ISP、最后一次投递事件为 “DELIVERY”、“OPEN” 或 “CLICK” 的最后一个参与事件，以及为输出文件指定的.json 格式：

```
{
    "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"
    }
}
```

- 在此示例中，输入文件使用MessageInsightsDataSource参数筛选以“@example1 .com”、“@example2 .com” 或 “@example3 .com” 结尾的目的地，排除 LastDeliveryEvent 等于 “发送” 或 “送达” 且为输出文件指定.csv 格式的邮件：

```
{
    "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"
    }
}
```

在 Amazon SES API v2 参考中，通过 MessageInsightsDataSource 中作为类型 ExportDataSource 的对象，可以找到有关参数值和相关数据类型的更多信息。

使用 AWS CLI管理导出任务

这些示例向您展示了如何使用 AWS CLI，通过列出导出任务、获取有关导出任务的信息以及取消它们来管理这些任务。

要列出您的导出任务，请使用 AWS CLI

您可以使用 Amazon SES API v2 中的 ListExportJobs 操作来列出导出任务。您可以从调用此操作， AWS CLI 如以下示例所示。

- 列出您的导出任务：

```
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
```

- 输入文件如下所示：

```
{
  "NextToken": "",
  "PageSize": 0,
  "ExportSourceType": "METRICS_DATA",
  "JobStatus": "CREATED"
}
```

有关 ListExportJobs 操作的参数值的更多信息，请参阅 Amazon SES API v2 参考。

要获取有关您的导出任务的信息，请使用 AWS CLI

您可以使用 Amazon SES API v2 中的 GetExportJob 操作来以获取有关导出任务的信息。您可以从调用此操作， AWS CLI 如以下示例所示。

- 获取有关导出任务的信息：

```
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
```

- 输入文件如下所示：

```
{
    "JobId": "e2220d6b-dce5-45f2-bf60-3287a465b732"
}
```

有关 GetExportJob 操作的参数值的更多信息，请参阅 Amazon SES API v2 参考。

要取消您的导出任务，请使用 AWS CLI

您可以使用 Amazon SES API v2 中的 CancelExportJob 操作来取消导出任务。您可以从调用此操作， AWS CLI 如以下示例所示。

- 取消导出任务：

```
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
```

- 输入文件如下所示：

```
{
    "JobId": "e2220d6b-dce5-45f2-bf60-3287a465b732"
}
```

有关 CancelExportJob 操作的参数值的更多信息，请参阅 Amazon SES API v2 参考。

使用查看消息的完整事件历史记录和 ISP 的回复 AWS CLI

以下示例向您展示了如何使用 AWS CLI查看邮件完整事件历史记录的详细信息以及邮箱提供商返回的任何回复或诊断代码。这与在虚拟可交付性管理器控制面板的邮件表中选择邮件的单选按钮后，在邮件信息侧边栏中使用的数据相同。

要查看消息的事件历史记录和 ISP 的回复，请使用 AWS CLI

您可以使用 Amazon SES API v2 中的 GetMessageInsights 操作查看已发送邮件的详细信息。您可以从调用此操作， AWS CLI 如以下示例所示。

- 查看通过 message-id 标识的已发送电子邮件的邮件详细信息：

```
aws --region us-east-1 sesv2 get-message-insights --message-id
 01000100001000dd-2a19190d-99d4-0000-9f00-deb5bbf2bfbe-000001
```

有关 GetMessageInsights 操作的参数值的更多信息，请参阅 Amazon SES API v2 参考。

如何计算 Virtual Deliverability Manager 控制面板指标

虚拟可交付性管理器控制面板中显示的所有指标比率卡片和向下钻取表都会计算在完整账户概览面板中输入的日期范围内的指标。

控制面板中显示的指标比率百分比按表中的说明进行计算。最后四列代表用于推导显示指标的基本数学的限定符。例如，打开率的计算方法是，打开总次数除以开启互动跟踪功能时送达的 HTML 邮件的总数。它们不反映您在未进行互动跟踪的情况下发送的任何邮件，也不采用 HTML 编码。

比率百分比	它是如何计算的	启用互动跟踪和 HTML	并且至少有 1 个跟踪链接	用 SES F ISPs BL 配送到	位于账户级黑名单中时则排除
打开率	打开总次数/送达总次数	✗			
点击率	点击总次数/送达总次数	✗	✗		
投诉率	投诉总次数/送达总次数			✗	✗
送达率	送达总次数/发送总次数				
临时退信率	临时退信总次数/发送总次数				✗
永久退信率	永久退信总次数/发送总次数				✗
发送总量	未显示比率百分比（您发送的所有邮件；始终为 100%）				

如何计算所有指标的差异率和总量：

- 差异百分比 - 在给定日期范围内，指标总数与先前指标总数的差异。例如，如果最近 7 天是指定的日期范围，则为最近 7 天的指标比率 - 前 7 天的指标比率。
  - 总发送量的差异百分比的计算方法有所不同。例如，（最近 7 天的发送量 - 前 7 天的发送量）/ 前 7 天的发送量。
- 量 – 每个指标的总计数。

> **Note**
>
> - 向下钻取表中的 Delivered（送达）列显示直接送达量，但未使用送达限定符来计算打开率、点击率和投诉率。

- Virtual Deliverability Manager 仅跟踪具有一个收件人的电子邮件中的指标，具有多个收件人的电子邮件不计入任何 Virtual Deliverability Manager 控制面板指标。
  - 在这些情况下，您的虚拟送达率管理器指标计数将低于您的Amazon CloudWatch 指标计数，因为 CloudWatch 指标包括有多个收件人的电子邮件。
- 发送到 SES 邮箱模拟器的电子邮件不计入任何 Virtual Deliverability Manager 控制面板指标。
- 通过代理发件人的账户发送（以前称为跨账户发送）的电子邮件不计入任何虚拟可交付性管理器控制面板指标。

Important

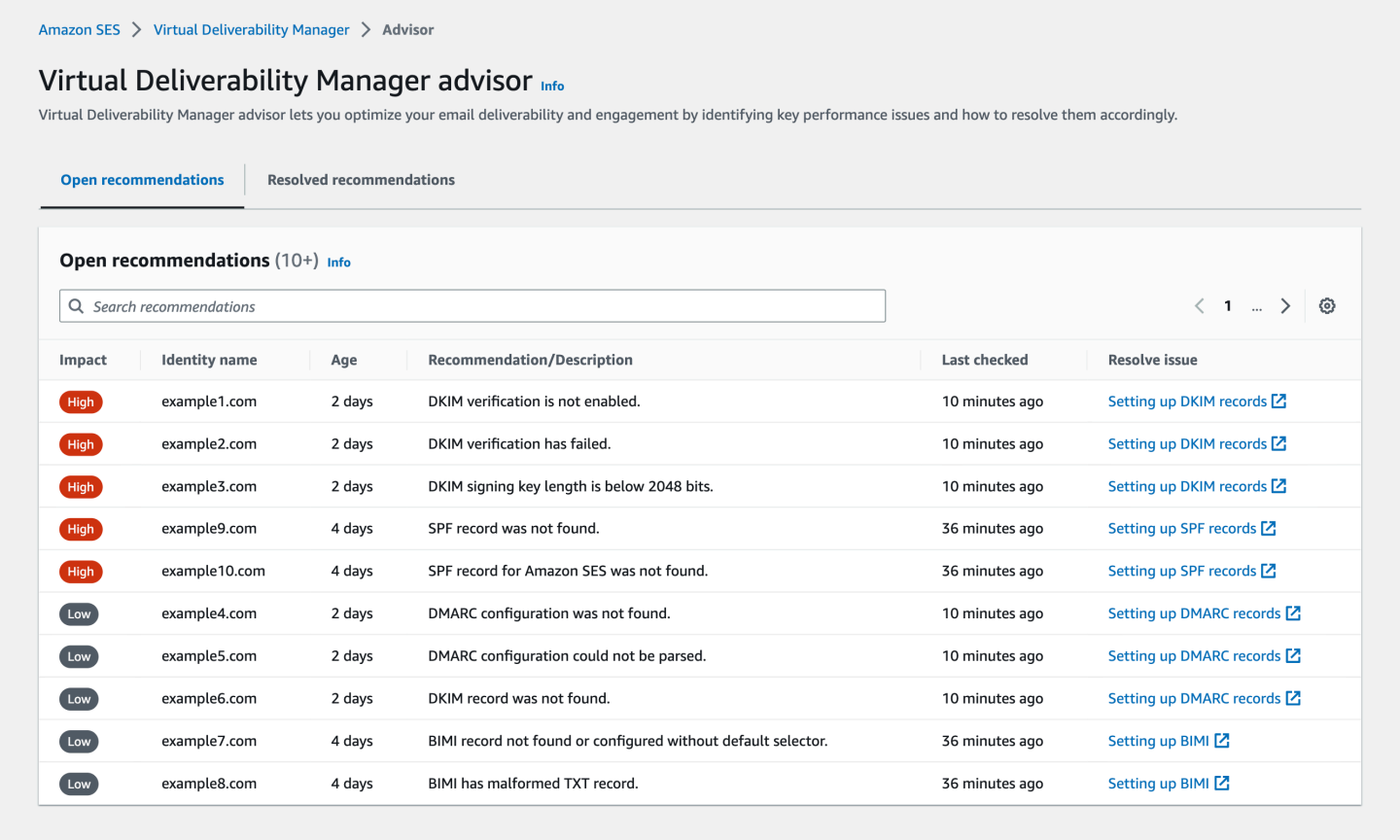
Apple Mail的隐私保护及其对参与率的影响：由于苹果从我起在苹果设备上实施了邮件隐私保护（MPP）功能OS15，随着Apple Mail应用程序启动时MPP触发器的打开，参与人数变得夸大了，不一定是在收件人打开使用and/or clicks a message. This causes engagement data to look much higher than it typically would be and this is something email marketers will have to take into account when reviewing engagement. There are several other ways of identifying engagement, such as web activity, app/portal情况以及使用来自非苹果设备的代理数据来建立汇总指标的时候。需要关注的重要事项是互动趋势，因为这可以表明您的电子邮件发送是否存在问题。有关更多信息，请参阅 Apple Mail 隐私保护。

Virtual Deliverability Manager advisor

Virtual Deliverability Manager advisor 通过在账户和发送身份级别识别对您的电子邮件送达率和声誉产生不利影响的关键性能和基础架构问题，来帮助优化您的电子邮件送达率和参与度。它通过提供有关如何解决已发现问题的具体指导来提供解决方案。

Advisor 的基础架构建议列在待处理建议表中。这些建议确定了标准电子邮件身份验证问题，例如 SPF、DKIM、DMARC 或 BIMI 记录不存在或者配置有问题（如格式错误或者密钥长度太短）。这些建议按 Impact（影响）严重程度、发送域的 Identity name（身份名称）和警报 Age（时长）进行分类。在搜索栏中，列表框提供了按影响级别、基础架构类别或发送身份名称进行筛选的选项。Last checked（上次检查）列会显示建议上次更新的相对时间，例如“刚才更新”或“15 分钟前”。最后一列 Resolve issue（解决问题）提供了指向《Amazon SES 开发人员指南》中相关部分的链接，其中包含有关如何解决已发现问题的指导。

待处理建议显示在 Virtual Deliverability Manager advisor 中，按影响级别排序。



如果您没有收到任何持续的 advisor 通知，则会显示一条消息，表明您没有任何待处理的建议。建议您定期查看 advisor。或者，您可以将这些顾问通知事件与 Amazon 集成 EventBridge ，以构建可扩展的事件驱动型应用程序，如中所述。使用监控 EventBridge

您还可以从 Virtual Deliverability Manager advisor 页面访问 Resolved recommendations（已处理建议）表，该表中列出了您通过实施 advisor 指导而解决的基础架构问题。列出的已处理建议的初始状态描述了得到解决之前的问题。已处理的建议将在 30 天后过期。

虚拟可交付性管理器 Advisor 的检查内容

在上一节中，我们讨论了虚拟可交付性管理器 Advisor 会对您的发送域进行检查，以确定您是否配置了经过安全身份验证的基础设施，从而确保您能够保持较高的电子邮件送达率并维护良好的发件人声誉。在激活虚拟可交付性管理器 Advisor 之前，我们认为让您确切了解 Advisor 的检查内容以及在这些检查中寻找什么将对您有所帮助。

您可以使用此表作为参考，来检查您的发送域配置，并更正不符合本表所列标准的任何元素，以免它们成为 Advisor 必须提醒您注意的问题。

检查的类型	Advisor 消息	Advisor 提醒您的原因	了解更多
投诉率检查	*ISP_name*互联网服务提供商有*high/med/low* 投诉率。	该身份已超出此 ISP 的投诉建议阈值。	监控发件人声誉
DKIM 配置	未启用 DKIM 验证。	未按身份启用 DKIM。	SES 中的 Easy DKIM
DKIM 密钥强度	DKIM 签名密钥长度低于 2048 位。	DKIM 签名密钥长度未使用至少 2048 位。	SES 中的 Easy DKIM
DKIM DNS 记录验证	DKIM 验证失败。	在查找并尝试验证密钥后，DKIM CNAME 记录被确定为无效。	与您的 DNS 提供商一起验证 DKIM 域身份
DMARC 配置	未找到 DMARC 配置。	缺少 DMARC TXT 记录。	对域设置 DMARC 策略
DMARC DNS 记录格式检查	无法解析 DMARC 配置。	在 DMARC TXT 记录中发现无效格式。	对域设置 DMARC 策略
DMARC 的 DKIM 配置	找不到 DKIM 记录。	未找到符合 DMARC 要求的 DKIM 记录。	通过 DKIM 遵守 DMARC
DMARC 的 DKIM 配置	DKIM 记录不一致。	DKIM 签名中指定的域与发件人地址中的域不一致（匹配）。	通过 DKIM 遵守 DMARC
SPF 配置	找不到 SPF 记录。	缺少自定义 MAIL FROM 域的 SPF TXT 记录。	配置自定义 MAIL FROM 域
已配置 SPF“include”	未找到 Amazon SES 的 SPF 记录。	SPF TXT 记录中缺少 `include:amazonses.com` 。	配置自定义 MAIL FROM 域
已配置 SPF 执行	缺少 SPF all 限定符。	SPF TXT 记录中缺少 `~all`。	配置自定义 MAIL FROM 域

检查的类型	Advisor 消息	Advisor 提醒您的原因	了解更多
SPF 执行验证	发现一个 SPF 配置问题。	尝试在 72 小时内检测所需的 SPF MX 记录失败。	自定义 MAIL FROM 域设置状态
BIMI 已配置	未找到 BIMI 记录或未使用默认选择器配置。	BIMI TXT 记录缺失或缺少选择器属性。	设置 BIMI
BIMI 格式验证	BIMI 的 TXT 记录格式错误。	在检查版本、证书 URL 和徽标 URL 的存在性和有效格式后，确定 BIMI TXT 记录配置错误。	设置 BIMI

在 Amazon SES 控制台中使用 Virtual Deliverability Manager advisor

以下过程演示了如何在 Amazon SES 控制台中使用 Virtual Deliverability Manager advisor 来解决已发现的送达问题。

要使用 Virtual Deliverability Manager advisor 来解决送达和声誉问题，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，在 Virtual Deliverability Manager 下，选择 Advisor。

   > **Note**
   >
   > 如果您尚未为账户启用 Virtual Deliverability Manager ，则无法看到 Advisor。有关更多信息，请参阅 the section called “入门”。

3. 默认情况下，将显示 Open recommendations（待处理的建议）表。建议按 Impact（影响）（高/低）、Identity name（身份名称）（发送域）、（警报的）Age（时长）和 Recommendation/Description（建议/描述）（已发现的问题）进行分类。在搜索栏中，按 Impact（影响）级别、基础架构问题的 Category（类别）或发送域的 Identity name（身份名称）进行筛选。

4. 要修复 Recommendation/Description（建议/描述）列中描述的问题，请选择该行的 Resolve issue（解决问题）列中的链接，然后实施建议的解决方案。

> **Note**
>
> 实施解决方案后，已解决的问题最多可能需要六个小时即可得到反映。您可以在 Resolved recommendations（已处理建议）选项卡上查看已解决的问题。

使用 AWS CLI访问 Virtual Deliverability Manager 建议

以下示例演示了如何使用 AWS CLI访问 Virtual Deliverability Manager 建议。

要访问您的虚拟可交付性管理器推荐，请使用 AWS CLI

您可以使用 Amazon SES API v2 中的 `ListRecommendations` 操作来列出您的送达建议。您可以从 AWS CLI调用此操作，如以下示例所示。

- 列出建议以查看送达问题：

```
aws --region us-east-1 sesv2 list-recommendations
```

- 应用筛选器，检索针对您拥有的特定域提供的建议：

```
aws --region us-east-1 sesv2 list-recommendations --cli-input-json file://list-
recommendations.json
```

- 输入文件如下所示：

```
{
  "PageSize":100,
  "Filter":{
    "RESOURCE_ARN": "arn:aws:ses:us-east-1:123456789012:identity/example.com"
   }
}
```

Virtual Deliverability Manager 设置

您可以随时查看或更改账户中的 Virtual Deliverability Manager 设置。您可以启用或禁用虚拟可交付性管理器，也可以通过 Amazon SES 控制台或 Amazon SES 控制台在虚拟交付管理器账户级别指定参与跟踪和优化共享交付的开启或关闭模式 AWS CLI

Virtual Deliverability Manager 选项还在配置集级别提供，这使您可以定义自定义设置，通过覆盖自定义设置在 Virtual Deliverability Manager 中的定义方式，来确定配置集将如何使用互动跟踪和优化共享送达功能。这使您可以灵活地为特定的电子邮件营销活动量身定制电子邮件发送方式。例如，您可以为营销电子邮件启用互动跟踪和优化共享送达功能，并对事务性电子邮件禁用它们。

使用 Amazon SES 控制台更改您的 Virtual Deliverability Manager 账户设置

以下过程演示了如何使用 Amazon SES 控制台来更改您的 Virtual Deliverability Manager 账户设置。

要使用 Amazon SES 控制台来更改 Virtual Deliverability Manager 账户设置，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航窗格中，在 Virtual Deliverability Manager 下，选择 Settings（设置）。

   此时将打开 Virtual Deliverability Manager settings（Virtual Deliverability Manager 设置）页面。Subscription overview（订阅概览）面板显示 Virtual Deliverability Manager 的状态，Additional settings（其他设置）面板显示 Engagement tracking（互动跟踪）和 Optimized shared delivery（优化共享送达）设置的状态。
3. 要更改 Engagement tracking（互动跟踪）或 Optimized shared delivery（优化共享送达）设置，请执行以下操作：

   a. 在 Additional setting（其他设置）面板中，选择 Edit（编辑）。
   b. 选择相应的单选按钮以打开或关闭任一功能，然后选择 Submit settings（提交设置）。

      Virtual Deliverability Manager settings（Virtual Deliverability Manager 设置）页面在 Additional settings（其他设置）面板中显示您所做更改的摘要。

      > **Note**
      >
      > 您在此处或在 Virtual Deliverability Manager 的配置集覆盖项中定义的 Engagement tracking（互动跟踪）选项，控制是否在 Virtual Deliverability Manager 控制面板中报告打开和单击事件；它们不会影响发布打开和单击事件的事件目标配置。例如，如果

> 您在此处禁用了互动跟踪，则不会禁用您在 SES 事件目标中设置的打开和单击事件发布。

4. （可选）要定义自定义设置以确定配置集将如何使用互动跟踪和优化共享送达，可覆盖自定义设置在 Virtual Deliverability Manager 中的定义方式；可在创建或编辑配置集时参考 Virtual Deliverability Manager 选项。
5. 要禁用 Virtual Deliverability Manager，请执行以下操作：
   a. 在 Subscription overview（订阅概览）面板中，选择 Disable Virtual Deliverability Manager（禁用 Virtual Deliverability Manager）。
   b. 在 Disable Virtual Deliverability Manager?（禁用 Virtual Deliverability Manager?）弹出窗口中，在确认字段中输入 *Disable*，然后选择 Disable Virtual Deliverability Manager（禁用 Virtual Deliverability Manager）。
   c. 此时将显示一条横幅，确认您已禁用 Virtual Deliverability Manager。
6. 要重新启用 Virtual Deliverability Manager，请参阅 the section called “入门”。

使用 AWS CLI更改 Virtual Deliverability Manager 账户设置

您可以使用 AWS CLI更改您的 Virtual Deliverability Manager 账户设置。

要更改您的虚拟送达率管理器账户设置，请使用 AWS CLI

您可以使用 Amazon SES API v2 中的 `PutAccountVdmAttributes` 和 `PutConfigurationSetVdmOptions` 操作来更改 Virtual Deliverability Manager 设置。您可以从调用此操作 AWS CLI，如以下示例所示。

- 使用输入文件启用或禁用互动跟踪和/或优化共享送达：

```
aws --region us-east-1 sesv2 put-account-vdm-attributes --cli-input-json file://
attributes.json
```

在此示例中，互动跟踪已 ENABLED，优化共享送达已 DISABLED，输入文件如下所示：

```
{
    "VdmAttributes": {
        "VdmEnabled": "ENABLED",
        "DashboardAttributes": {
            "EngagementMetrics": "ENABLED"
```

```
        },
        "GuardianAttributes": {
            "OptimizedSharedDelivery": "DISABLED"
        }
    }
}
```

通过从 Amazon SES API v2 参考中的 VdmAttributes 数据类型进行链接，您可以查找有关参数值和相关数据类型的更多信息。

- 通过覆盖自定义设置在 Virtual Deliverability Manager 中的定义方式，可以定义自定义设置，以确定配置集将如何使用互动跟踪和优化共享送达：

```
aws --region us-east-1 sesv2 put-configuration-set-vdm-options --cli-input-json
 file://config-set.json
```

在此示例中，名为 example 的配置集同时启用了互动跟踪和优化共享送达，输入文件如下所示：

```
{
    "ConfigurationSetName": "example",
    "VdmOptions": {
        "DashboardOptions": {
            "EngagementMetrics": "ENABLED"
        },
        "GuardianOptions": {
            "OptimizedSharedDelivery": "ENABLED"
        }
    }
}
```

有关参数值和相关数据类型的更多信息，请参阅 Amazon SES API v2 参考中的 VdmOptions 数据类型。

- 要验证结果，请执行以下操作：

```
aws –region us-east-1 sesv2 get-configuration-set –configuration-set-name example
```

- 如果没有在配置集级别指定 DashboardOptions 或 GuardianOptions 选项，将导致您的 Virtual Deliverability Manager 账户级设置应用于通过该配置集发送的流量。

Amazon SES 中的 Mail Manager

Mail Manager 是一组 Amazon SES 电子邮件网关功能，旨在协助您增强组织的电子邮件基础设施、简化电子邮件工作流程管理和精简电子邮件合规性控制。它可以与您现有的基础设施集成，能够连接不同的业务应用程序，并自动处理入站电子邮件。Mail Manager 还通过高效管理您的电子邮件流量，并借助其电子邮件存档功能增强合规性，在维护电子邮件系统健康方面发挥着第一道防线的作用。

除了当前的 Amazon SES 功能外，Mail Manager 还包括以下支持入站流量的功能：

- 入站端点 – 一个关键的基础设施组件，它利用您可以配置的筛选策略和规则，来确定哪些电子邮件可以允许进入您的组织，哪些应该被拒绝。
- 流量策略和规则集 – 使电子邮件管理员能够使用高度可自定义的策略和规则来定义和执行用于管理入站电子邮件流量的规则，这些策略和规则可以根据您定义的一系列丰富条件和例外情况对电子邮件进行排序、分类、划分优先级和执行操作。这种智能筛选功能与自动化工作流程相结合，有助于精简电子邮件管理、提高效率并确保遵守组织的电子邮件策略。
- SMTP 中继 – 通过连接内部电子邮件系统，根据您在规则中定义的标准将电子邮件流量重定向到其他 SMTP 服务器，并利用自动转发功能简化电子邮件管理。能够在多个服务器和网关之间分配流量，使您的组织即使在混合环境中，也能够有效地管理大量电子邮件流量。
- 电子邮件存档 – 通过将数据存储在持久且安全的长期存储中，来保存和保护您的电子邮件，并为您提供一种快速搜索和存档电子邮件的方法。它提供全天候的企业级存档服务，而不会增加邮箱服务器的存储需求。
- 电子邮件插件 – 由 SES 批准的提供商提供的一系列专业安全工具，可用于管理传入您的入口端点的电子邮件，并根据安全结果提供路由选项。这些工具是经过认证的安全情报和执行解决方案，可以随时集成到您的电子邮件工作流程中，并且可以直接从 Mail Manager 控制台激活。

Mail Manager 入门

要开始使用 Mail Manager，Amazon SES 控制台中的入门向导将引导您完成为账户启用 Mail Manager 的步骤。请参阅the section called “入门”。

主题

- SMTP 中继
- 地址清单
- 消息存档
- 电子邮件插件
- Mail Manager 的权限策略
- 邮件管理器日志

Mail Manager 入门

要开始使用 Amazon SES Mail Manager，您可以使用 Amazon SES 控制台中的 Mail Manager 入门向导，在该向导中，您将创建一个入口端点，并使用流量策略和规则集对其进行配置。

入口端点是您设置 Mail Manager 的第一个构建基块，该关键基础设施组件利用以下项目：

- 流量策略 – 流量策略包含您定义的策略声明，用于在满足策略声明的条件时，通过允许或阻止特定类型的电子邮件来对传入的邮件进行分类。
- 规则集 – 规则集包含您定义的规则，用于在满足规则条件时，对允许的电子邮件执行操作。

但是，创建入口端点的一部分流程是选择已经创建的流量策略和规则集，然后将其分配给入口端点。以下过程中的步骤将引导您按正确的顺序配置第一个入口端点。

通过 SES 控制台开始使用 Mail Manager

以下过程演示了如何通过 SES 控制台开始使用 Mail Manager。

通过 Amazon SES 控制台开始使用 Mail Manager

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航面板中，选择 Mail Manager，然后在 Mail Manager 概述页面上选择任何 Mail Manager 入门按钮。
3. 在获取设置页面上，选择创建流量策略卡上的创建流量策略。
    a. 完成创建流量策略页面上的工作流程。如需额外信息，请参阅the section called “创建流量策略和策略声明（控制台）”。

b. 在创建第一个流量策略和策略声明后，使用浏览器的返回按钮返回到获取设置页面，或者在左侧导航面板的 Mail Manager 下，选择获取设置。

4. 在获取设置页面上，选择创建规则集卡上的创建规则集。

a. 完成创建规则集页面上的工作流程。如需额外信息，请参阅the section called “创建规则集和规则（控制台）”。

b. 在创建第一个规则集和规则后，使用浏览器的返回按钮返回到获取设置页面，或者在左侧导航面板的 Mail Manager 下，选择获取设置。

5. 现在，您已经创建了第一个流量策略和规则集，接下来就可以创建第一个入口端点。在获取设置页面上，选择创建入口端点卡上的创建入口端点。

- 电子邮件入口端点页面上的部分工作流程是，将您刚刚创建的流量策略和规则集分配给入口端点。如需额外信息，请参阅the section called “创建入口端点（控制台）”。

创建第一个入口端点后，您就可以开始使用 Mail Manager 并利用它的其他功能，例如 SMTP 中继和电子邮件存档。您还可以使用独特的流量策略和规则集创建其他入口端点，以进一步自定义管理所有传入电子邮件的方式。

入口端点

入口端点是 Mail Manager 中的关键基础设施组件，它利用您配置的策略和规则来接收、路由和管理您的电子邮件，以确定哪些电子邮件应被拒绝，哪些应被允许，以及应该对哪些电子邮件采取相应操作。

每个入口端点都有自身的流量策略来确定要阻止或允许的电子邮件，并拥有规则集来对您允许的电子邮件执行操作；因此，通过创建多个入口端点，您可以指派每个入口端点来管理和路由特定类型的电子邮件。这种精细度级别有助于您构建针对您的业务需求量身定制的电子邮件管理系统。

创建入口端点的必备工作流程

在创建入口端点时，必须为其分配已创建的流量策略和规则集。因此，创建入口端点的工作流程应遵循以下顺序：

1. 首先，创建流量策略以确定要阻止或允许的电子邮件。有关更多信息，请参阅 the section called “创建流量策略和策略声明（控制台）”。
2. 接下来，创建一个规则集以对允许传入的电子邮件执行操作。有关更多信息，请参阅 the section called “创建规则集和规则（控制台）”。
3. 最后，创建您的入口端点，并将您刚刚创建的流量策略和规则集或您之前创建的任何其他流量策略和规则集分配给它。

创建入口端点后，您必须根据用于接收邮件的环境对其进行配置，无论是本地 SMTP 客户端的配置还是基于 Web 的 DNS 域主机的配置。下面的the section called “公共端点配置”部分将对此进行讨论。

配置您的环境以使用入口端点

SES 支持公共终端节点和亚马逊虚拟私有云 (VPC) 终端节点，让入口终端节点接受传入的电子邮件。以下各节说明如何配置您的入口端点以使用这两个选项中的任何一个。

主题

通过公共端点接收电子邮件

使用“A”记录

在您创建入口端点时，将生成该端点的“A”记录，其值将显示在 SES 控制台的入口端点摘要屏幕上。您使用此记录值的方式具体取决于您创建的端点类型和您的应用场景：

- 开放端点 – 发送到您域的邮件将直接解析到您的入口端点，无需进行身份验证。
  - 将“A”记录的值直接复制并粘贴到本地 SMTP 客户端的 SMTP 配置中，或 DNS 配置中域的 MX 记录中。
  - 支持的端口：25
  - 支持 STARTTLS：是
- 已验证的端点 – 发送到您的域的邮件必须来自您与之共享 SMTP 凭据的授权发件人，例如您的本地电子邮件服务器。
  - 将“A”记录的值以及您的用户名和密码，直接复制并粘贴到本地 SMTP 客户端的 SMTP 配置中。
  - 支持的端口：25、587（RFC 2476）
  - 支持 STARTTLS：是

如果您在配置中使用 MX 记录，请注意，虽然每个 DNS 提供商的配置记录的程序和界面不同，但您需要在 DNS 设置中输入的关键信息如以下示例所示：

所有发送到 recipient@marketing.example.com 的电子邮件都将发送到您的入口端点，因为您在域的 DNS 设置中，将入口端点的“A”记录作为 MX 记录的值进行了输入：

- 域 – `marketing.example.com`

- MX 记录值 – 890123abcdef.ghijk.mail-manager-smtp.amazonaws.com（这是从您的入口端点复制的“A”记录值。）
- 优先级 - 10

连接到已验证的端点

对于为了连接到已验证的端点而与之共享 SMTP 凭证的授权发件人，用户名和密码必须遵循以下协议，才能成功建立与服务器的连接：

- 用户名 – 这是入口端点 ID，必须使用 Base64 进行编码。（请参阅步骤 11。 在控制台程序中学习如何查找入口端点 ID。）
- 密码 – 这是创建入口端点时使用的密码，必须使用 Base64 进行编码。

以下示例显示了典型的 SMTP AUTH 服务器和客户端在建立连接时的交互过程：

```
S: 250 AUTH LOGIN PLAIN
C: AUTH LOGIN
S: 334 VXNlcm5hbWU6
C: SW5ncmVzc1BvaW50
S: 334 UGFzc3dvcmQ6
C: SW5ncmVzc1Bhc3N3b3Jk
S: 235 Authentication successful
```

此示例包含以下属性：

- S 表示“服务器”– 接受消息的 SMTP 服务器。
- C 表示“客户端”- 与服务器建立连接并向服务器发送消息的 SMTP 客户端。
- 250 AUTH LOGIN PLAIN 是服务器对支持的 AUTH 方法（AUTH LOGIN 或 AUTH PLAIN）的响应，发件人可以选择其中任何一种方法，然后发送符合 SMTP 身份验证服务扩展规范 RFC 2554 的 SMTP 命令。本例中使用 AUTH LOGIN。
- 334 VXNlcm5hbWU6 – 服务器以 Base64 编码提示输入用户名。
- SW5ncmVzc1BvaW50 – 客户端以 Base64 编码响应入口端点 ID。
- 334 UGFzc3dvcmQ6 – 服务器以 Base64 编码提示输入密码。
- SW5ncmVzc1Bhc3N3b3Jk – 客户端以 Base64 编码响应入口端点密码。

通过 Amazon VPC 终端节点接收电子邮件

除了公共入口终端节点外，您还可以将 VPC 终端节点与 SES 入口终端节点配合使用，以便在您的私有网络基础设施中安全地接收私人电子邮件。

与使用公共入口端点相比，配置差异

- 未提供通常可用于公共端点的 “A” 记录。
- 您必须使用 VPC 终端节点提供的 DNS 名称连接到入口终端节点。
- 所有连接都使用您的 VPC 内的私有网络。

VPC 终端节点支持的入口终端节点类型

SES 通过 VPC 终端节点支持两种类型的入口点：

- 打开入口终端节点-直接通过 VPC 终端节点发送到您的域的电子邮件，无需发件人身份验证。

  配置要求：
  - 通过将私有开放入口终端节点与您拥有的 VPC 终端节点 ID 关联来创建该终端节点。
  - 支持的端口：25、587
  - 支持 STARTTLS：是
- 经过身份验证的入口端点 — 发送到您域的邮件必须来自您与之共享 SMTP 凭据的授权发件人，例如您的本地电子邮件服务器。

  配置要求：
  - 通过将私有经过身份验证的入口终端节点与您拥有的 VPC 终端节点 ID 关联来创建该终端节点。
  - 支持的端口：25、587
  - 支持 STARTTLS：是
  - 身份验证使用与经过身份验证的公共端点相同的 base64 编码的用户名和密码机制。

VPC 终端节点要求

要将 VPC 终端节点与 SES 入口终端节点配合使用，必须满足以下要求：

- VPC 终端节点必须处于活动状态且可用。
- VPC 终端节点必须与入口终端节点归同一个 AWS 账户所有（不支持跨账户访问）。

- 必须根据入口终端节点的类型为相应的服务名称创建 VPC 终端节点：
  - 打开入口端点 — com.amazonaws.*region*.mail-manager-smtp.open
  - 经过身份验证的入口端点 — com.amazonaws.*region*.mail-manager-smtp.auth
  - FIPS 开放入口端点 — com.amazonaws.*region*.mail-manager-smtp.open.fips
  - 经过 FIPS 身份验证的入口端点 — com.amazonaws.*region*.mail-manager-smtp.auth.fips

重要的配置说明

- 美国和加拿大区域 — FIPS 终端节点是美国和加拿大地区唯一可用的 VPC 终端节点，也是这些区域中使用的正确终端节点。
- One-to-one 关系 — 每个 VPC 终端节点只能与一个入口终端节点关联。您不能将相同的 VPC 终端节点用于多个入口终端节点。
- 没有 VPC 终端节点策略 — 与其他 AWS 服务不同，与入口终端节点一起使用的 VPC 终端节点不支持 VPC 终端节点策略。SES 会自动验证 VPC 终端节点所有者和入口终端节点所有者是否为同一个 AWS 账户。
- 仅限私有 DN S — VPC 终端节点提供的所有 DNS 名称都将是只能在您的 VPC 内访问的私有 DNS 名称。
- 创建时进行验证 — SES 在资源创建期间执行验证，以确保 VPC 终端节点满足所有要求。

通过 VPC 终端节点连接到您的入口终端节点

配置 VPC 终端节点和入口终端节点后：

1. 检索为您的 VPC 终端节点生成的 DNS 名称。
2. 将 SMTP 客户端或电子邮件服务器配置为使用这些 DNS 名称进行连接。
3. 如果使用经过身份验证的终端节点，请使用与经过身份验证的入口端点一起使用的相应的 base64 编码凭据来配置 SMTP 客户端。

在 SES 控制台中创建入口端点

以下过程演示了如何使用 SES 控制台中的入口端点页面，创建入口端点以及管理已创建的入口端点。

使用控制台创建和管理入口端点

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航面板中，选择 Mail Manager 下的入口端点。
3. 在入口端点页面上，选择创建入口端点。
4. 在创建新入口端点页面上，为您的入口端点输入一个唯一的名称。
5. 选择是创建开放端点还是已验证的端点。

   - 如果您选择已验证，请选择 SMTP 密码并输入密码（与授权发件人共享），或者选择密钥，然后从密钥 ARN 中选择一个密钥。如果您选择之前创建的密钥，则该密钥必须包含以下创建新密钥步骤中指示的策略。
   - 你可以通过选择 Create new 来创建新密钥—— AWS Secrets Manager 控制台将打开，你可以继续创建新密钥：

   a. 在密钥类型中，选择其他密钥类型。

   b. 在密钥/值对中，输入 `password` 作为密钥，并输入实际密码作为值。

   > **Note**
   >
   > 对于密钥，您只需输入 `password`（其他任何内容都会导致身份验证失败）。

   c. 选择添加新密钥以在加密密钥中创建 KMS 客户托管密钥 (CMK) — AWS KMS 控制台将打开。

   d. 在客户自主管理型密钥页面上，选择创建密钥。

   e. 保留配置密钥页面上的默认值，然后选择下一步。

   f. 在别名中输入密钥的名称（您可以选择添加描述和标签），然后单击下一步。

   g. 在密钥管理员中，选择您想要允许管理密钥的任何用户（您自己除外）或角色，然后选择下一步。

   h. 在密钥用户中，选择想要允许使用密钥的任何用户（您自己除外）或角色，然后选择下一步。

   i. 将 KMS CMK 策略 复制并粘贴到 `"statement"` 级别的密钥策略 JSON 文本编辑器中，将其作为额外声明添加进去，并用逗号进行分隔。将区域和账户替换为您自己的信息。

   j. 选择完成。

k. 选择打开 AWS Secrets Manager 存储新密钥页面的浏览器选项卡，然后选择加密密钥字段旁边的刷新图标（圆形箭头），然后在该字段内单击并选择您新创建的密钥。

l. 在配置密钥页面上的密钥名称字段中输入名称。

m. 在资源权限中，选择编辑权限。

n. 将 密钥资源策略 复制并粘贴到资源权限 JSON 文本编辑器中，然后将区域和账户替换为您自己的信息。（请务必删除编辑器中的所有示例代码。）

o. 选择保存，然后选择下一步。

p. （可选）配置轮换，然后选择下一步。

q. 查看您的新密钥并选择存储。

r. 选择浏览器中已打开 SES 创建新入口端点页面的选项卡，然后选择刷新列表，接着在密钥 ARN 中选择新建的密钥。

6. 选择一个规则集，其中包含您要对允许传入的电子邮件执行的规则操作。

7. 选择流量策略以确定要阻止或允许的电子邮件。

8. 选择它是公共网络还是私有网络。

   - 对于公共网络，请选择 “IPv4仅限” 或 Dualstack（IPv4 和 IPv6）寻址。
   - 对于私有网络，请选择或输入您与同一账户中的授权发件人（例如 IAM 用户或角色）共享的 VPC 终端节点。或者，您可以通过选择创建 VPC 终端节点打开 Amazon VPC 控制台来创建新的 VPC 终端节点。

9. 选择创建入口端点。

10. 在常规详细信息中，将在创建入口端点时显示 “配置”，请刷新页面，直到显示 “Active” 并且该ARecord字段包含一个值。复制“A”记录值并将其粘贴到您的 DNS 配置或 SMTP 客户端中，如公共端点配置中所述。

11. 在控制台的一般详细信息容器正上方，有一个以“inp”为前缀的未标记的大数字（在页面顶部的面包屑导航中也同样存在），例如 inp-1abc2de3fghi4jkl5mnop6qr。这称为入口端点 ID，其值用作登录入口服务器的用户名。（您需要与授权发件人共享此信息才能连接到您的端点。）

12. 您可以从入口端点页面查看和管理已经创建的入口端点。如果要删除某个入口端点，请选择其单选按钮，然后选择删除。

13. 要编辑某个入口端点，请选择其名称以打开相应的摘要页面：

    - 您可以通过在一般详细信息中选择编辑，然后选择保存更改来更改端点的活动状态。
    - 您可以通过在规则集或流量策略中选择编辑，然后选择保存更改来选择不同的规则集或流量策略。

流量策略和策略声明

流量策略是您分配给入口端点的策略声明的容器，以便在满足策略声明的条件时，通过允许或阻止特定类型的电子邮件来对传入的邮件进行分类。一个流量策略可以由多个入口端点使用。

> **Tip**
>
> 您可以将流量策略视为“筛选器集”，将策略声明视为“筛选器”。流量策略（筛选器集）包含用于筛选传入邮件的策略（筛选器）。

创建流量策略时，您可以选择设置最大消息大小（以字节为单位）。当一条消息超过该大小时，它就会被丢弃。您还可以将策略的默认操作设置为允许或阻止不符合策略声明条件的电子邮件，可以将其视为流量策略的 “catch all” 操作。

策略声明也是使用允许或阻止操作创建的，当声明的条件满足时执行这些操作。可以通过选择一个电子邮件协议以及为您输入的值指定一个条件运算符来构建条件，只有当传入的消息与这些条件匹配时，策略声明才会允许或阻止该消息。每个策略声明可以包含多个条件。

流量策略可以包含多个策略声明，并根据其评估电子邮件的隐式层次结构的顺序来执行这些声明：

- 阻止的策略声明 – 首先对这些声明进行评估，然后阻止任何符合声明条件的消息。
- 允许的策略声明 – 接着将对这些声明进行评估，并允许任何符合声明条件的消息。
- 流量策略的默认操作 – 对于不符合任何策略声明的其余消息，会根据您定义的此参数来允许或阻止这些消息。
- 最大邮件大小-如果设置了此可选参数，则任何大于此大小的消息都将被丢弃。

流量策略是一种独立的资源，可以由多个入口端点使用，但是策略声明仅属于创建它们的流量策略。因此，您必须先创建流量策略或编辑现有流量策略，然后才能创建策略声明来评估传入您的入口端点的电子邮件。

下一节中的过程说明如何在 SES 控制台中创建流量策略及其策略声明。

在 SES 控制台中创建流量策略和策略声明

以下过程向您展示，如何使用 SES 控制台中的流量策略页面来创建流量策略及其策略声明，以及如何管理已创建的流量策略及其策略声明。

使用控制台创建和管理流量策略和策略声明

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航面板中，选择 Mail Manager 下的流量策略。
3. 在流量策略页面上，选择创建流量策略。
4. 在创建流量策略页面上，请为您的流量策略输入唯一的名称。
5. （可选）如果要丢弃超过一定大小的消息，请最大消息大小字段中输入以字节为单位的值。
6. 在默认操作中，选择当消息不符合策略声明条件（不在策略声明条件范围内）时，流量策略是允许还是拒绝（阻止）这些消息。
7. 选择添加新策略声明，为您的流量策略创建声明。
8. 选择允许或拒绝（阻止），以便在满足声明的条件时执行操作。
9. 通过选择电子邮件协议，以及为输入的值选择条件运算符来构建条件。如果您想在此策略声明添加更多条件，请选择添加新条件。要了解有关条件属性及其运算符和有效值的更多信息，请参阅策略声明条件参考。
   - 如果您订阅了电子邮件插件，则可以在此选择其作为电子邮件协议。
   - 与 IPv6 或双栈入口端点关联的流量策略不会评估通过 IPv6 连接接收的消息，也不会将电子邮件附加条件应用于这些条件。Email Add On 条件仅适用于通过双堆栈终端节点上的 IPv4 连接接收的邮件。
10. 如果想添加更多策略声明和条件，请重复上述步骤 7-9。
11. 创建完策略声明及其条件后，选择创建流量策略。
12. 您可以从流量策略页面查看和管理已经创建的流量策略。如果要删除某个流量策略，请选择相应的单选按钮，然后选择删除。
13. 要编辑流量策略的属性或其任何策略声明，请选择其名称以打开概述页面，然后在其中选择编辑。
14. 在流量策略详细信息中，您可以更改最大消息大小和默认操作。
15. 在任何策略声明容器中，您可以更改允许/拒绝属性并编辑任何条件。您还可以删除策略声明和条件，也可以添加新的声明和条件。
16. 完成所有编辑后，选择保存更改以保存所做的更改。

策略声明条件参考

策略声明条件

以下参考表列出了可用于构建策略声明条件的所有策略声明协议。选择协议的表达式类型将跳转到《SES Mail Manager API 参考》中的相应参考页面，其中列出了该协议的所有可用运算符和有效值。

策略声明条件：协议、运算符和值

协议	表达式类型
接收人地址 Abusix Mail Intelligence（如果已订阅） • 上市于 Spamhaus 域阻止列表（如果已订阅） • 上市于	字符串表达式的有效运算符和值
发件人 IP 范围	IP 表达式的有效运算符和值
TLS 协议版本	TLS 协议表达式的有效运算符和值
Abusix Mail Intelligence（如果已订阅） • 已上市 Spamhaus 域阻止列表（如果已订阅） • 已上市	布尔表达式的有效运算符和值

规则集和规则

规则集是您分配给入口端点的规则的容器，以便它可以对入口端点流量策略允许的电子邮件执行操作。一个规则集可以由多个入口端点使用。

当消息满足规则条件时，规则通过执行规则中定义的操作来告诉您的入口端点如何处理传入的电子邮件。每条规则可以有多个条件和操作。您在规则集中创建的规则将按规则集中指定的顺序执行。

您可以通过选择一个电子邮件属性，并为输入的值选择一个条件运算符来构建规则的条件，只有当消息与这些条件匹配时，规则才会执行其操作，您可以定义要采取的操作及其执行顺序。

为了更精细地控制，您的规则还可以包含与条件定义类似的例外情况，但在这里，您定义的是一个消息必须不匹配的条件。条件和例外情况是独立运行的，如果您愿意，可以仅使用例外情况来构建规则，也可以将条件和例外情况混合使用。

由于在规则集中定义规则的方式非常精细，因此提供了以下列表来帮助说明规则集组件之间的关系：

- 规则集包含：
  - 规则 – 您可以定义规则在规则集中执行的顺序。

    规则包含：
    - 条件 – 如果消息与条件的评估相匹配，则该规则适用；如果规则有例外情况，请参见下文。
    - 例外 – 如果消息与例外情况的评估不匹配，则该规则适用；如果规则有条件，请参见上文。
    - 操作 – 在规则适用时触发操作，此时所有条件都匹配，无例外情况。

      您可以在规则中定义操作的执行顺序。

由于每个规则都可以包含多个条件、例外情况和操作，而且您可以定义规则和操作的执行顺序，因此您可以构建一个高度自定义且自动化的电子邮件处理解决方案，以满足您特定的业务需求。

规则集是一种独立的资源，可以由多个入口端点使用，但规则仅属于创建规则的规则集。因此，您必须先创建规则集或编辑现有规则集，然后才能创建规则以对传入您的入口端点的电子邮件执行操作。

下一节中的过程将指导您在 SES 控制台中创建规则集及其规则。

在 SES 控制台中创建规则集和规则

以下过程向您演示，如何使用 SES 控制台中的规则集页面来创建规则集及其规则，并管理已创建的规则集。

使用控制台创建并管理规则集和规则

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航面板中，选择 Mail Manager 下的规则集。
3. 在规则集页面上，选择创建规则集，并为规则集输入唯一的名称。
4. 在规则集的概述页面上，选择编辑，然后在编辑页面上选择创建新规则。
5. 在规则详细信息侧边栏中，输入规则的唯一名称。

6. 选择添加新条件以创建消息必须匹配的条件；或者选中以下情况除外：复选框，然后选择添加新例外以创建消息不得匹配的条件。
7. 通过选择电子邮件属性，以及为输入的值选择条件运算符来构建条件或例外。如果要向此规则添加更多条件或例外，请选择添加新条件或添加新例外。要了解有关条件属性及其运算符和有效值的更多信息，请参阅规则条件参考。

   - 如果您订阅了电子邮件插件，则可以在此选择其作为电子邮件属性。
8. 选择添加新操作以定义规则条件匹配和/或例外不匹配时要执行的操作。要添加更多要执行的操作，请选择添加新操作。创建两个或更多动作时，会显示向上/向下箭头，以便您可以设置执行顺序。

   > **Note**
   >
   > 要执行这些规则操作中的任何一个，您需要为您的账户启用相应的权限策略；否则，规则操作将失败。
9. 选择操作后，直接从 “规则详细信息” 面板中为其中任何一个操作应用权限策略：

   a. 在 IAM 角色字段中选择创建新角色，然后输入名称，接着选择创建角色。（此角色的 IAM 信任策略将在后台自动生成。）
   b. 由于 IAM 信任策略是自动生成的，因此您只需将操作的权限策略添加到角色中 - 选择 IAM 角色字段下的查看角色即可打开 IAM 控制台。
   c. 在权限选项卡下，选择添加权限，然后选择创建内联策略。
   d. 在指定权限页面上，在策略编辑器中选择 JSON。
   e. 将相应的政策从策略编辑器中复制并粘贴Mail Manager 的权限策略到策略编辑器中，然后将红色文本中的数据替换为自己的数据。（请务必删除编辑器中的所有示例代码。）
   f. 选择下一步。
   g. 选择创建策略，查看并创建您的 IAM 角色权限策略。
   h. 选择已打开 SES Mail Manager 编辑规则集页面的浏览器选项卡，然后继续执行创建规则的其余步骤。
10. 创建完规则的条件、例外情况和操作后，您可以选择位于左侧编辑规则集面板中的保存规则集，将其保存到规则集中。
11. 如果要向规则集添加更多规则，请重复上述步骤 4-9。

    - 创建两个或更多规则时，规则集的重新排序列中会显示向上/向下箭头，以便您可以设置执行顺序。

12. 您可以从规则集页面查看和管理已创建的规则集。如果您要删除某个规则集，请选择其单选按钮，然后选择删除。
13. 要编辑某个规则集，请选择其名称以打开相应的概述页面，在该页面中选择编辑，您可以在其中重新排列其规则的执行顺序，通过选择创建新规则来添加更多规则，或者通过选择某个规则的单选按钮然后选择删除来删除规则。
14. 要编辑某个规则，请选择其单选按钮。在规则详细信息侧边栏上的任何容器中，您可以编辑任何条件或例外情况，也可以更改或重新排序任何操作。您还可以删除条件、例外和操作，也可以添加新的条件、例外和操作。
15. 完成所有编辑后，通过选择左侧编辑规则集面板中的保存规则集来保存更改。

规则条件和操作的参考

规则条件

以下参考表列出了所有可用于构建规则条件（或例外）的规则属性，并按其表达式类型进行了分类。具有相同表达式类型的规则属性也具有相同的运算符和值。选择属性的表达式类型将跳转到《SES Mail Manager API 参考》中的相应参考页面，其中列出了该属性的所有可用运算符和有效值。

规则条件：属性、运算符和值

属性	表达式类型
发件人地址	
收件人地址	
抄送地址	
Mail from	
接收人地址	字符串表达式的有效运算符和值
主题	
Helo	
MIME 标头	
Vade 高级电子邮件安全（如果已订阅）	

属性	表达式类型
• 类别 • 判决 Trend Micro 病毒扫描（如果已订阅） • 类别	
IP 范围	IP 表达式的有效运算符和值
消息最大大小	数字表达式的有效运算符和值
DKIM SPF	判断表达式的有效运算符和值
TLS TLS 包装 已读回执 Vade 高级电子邮件安全（如果已订阅） • 已通过 Trend Micro 病毒扫描（如果已订阅） • 已通过	布尔表达式的有效运算符和值
DMARC 策略	DMARC 表达式的有效运算符和值

规则操作

以下参考表列出了在满足规则条件或不满足其例外情况时，可以执行的所有规则操作。选择某个操作将跳转到《SES Mail Manager API 参考》中的相应操作参考页面，其中列出了该操作的参数及其格式。该表使用了 Mail Manager 控制台中采用的操作名称，API 名称可能略有不同。

> **Note**
>
> 在某些 API 参考中，如果操作失败，可以将 `ActionFailurePolicy` 参数设置为继续或删除，这仅在使用 API 时适用；使用控制台时，`ActionFailurePolicy` 已设置为默认值继续。

规则操作：操作和参数

操作及其参数	描述
写入 S3	将电子邮件的 MIME 内容写入 S3 存储桶中。
SMTP 中继操作	通过 SMTP 将电子邮件中继到另一个特定的 SMTP 服务器。
存档操作	通过将电子邮件传送到 Amazon SES 存档进行存档。
添加标头	向收到的电子邮件添加自定义标头。
电子邮件收件人重写	用给定的收件人列表替换电子邮件信封收件人。如果此操作的条件仅适用于收件人的子集，则仅替换这些收件人。
传送到邮箱	将电子邮件发送到 Amazon WorkMail 邮箱。
配送给 Q Business	向 Amazon Q Business 应用程序发送一封电子邮件，供其提取到其知识库中。
发布到 SNS	将电子邮件内容发布到 Amazon SNS 主题中。
发送到互联网	使用 SES 将电子邮件发送给电子邮件收件人列表中的收件人。
删除操作	对于包含多个收件人的电子邮件，如果此操作适用于其中一个或多个（但不是全部）收件人，则这些收件人将从电子邮件的收件人列表中删除，并且将继续对剩余收件人应用规则处理。如果此操作适用于所有收件人，则规则处理将停止，因

操作及其参数	描述
	为所有收件人都已从收件人列表中删除，并且不会收到该电子邮件。

SMTP 中继

由于 Mail Manager 部署在您的电子邮件环境（例如 Microsoft 365、Google Workspace 或本地 Exchange）和互联网之间，因此 Mail Manager 使用 SMTP 中继将由 Mail Manager 处理的传入电子邮件路由到您的电子邮件环境。它还可以在将出站电子邮件发送给最终收件人之前，将出站电子邮件路由到其他电子邮件基础设施，例如其他 Exchange 服务器或第三方邮件网关。

SMTP 中继是电子邮件基础设施的重要组件，当规则集中定义的规则操作指定时，它负责在服务器之间高效地路由电子邮件。

具体而言，SMTP 中继可以在 SES Mail Manager 和外部电子邮件基础设施（例如 Exchange、本地或第三方电子邮件网关等）之间重定向传入的电子邮件。传入入口端点的电子邮件将由规则处理，该规则将指定的电子邮件路由到指定的 SMTP 中继，而后者又将其传递到 SMTP 中继中定义的外部电子邮件基础设施。

当您的入口端点收到电子邮件时，它会使用流量策略来确定要阻止或允许的电子邮件。您允许传入的电子邮件会传递给一个规则集，该规则集应用条件规则来执行您为特定类型的电子邮件定义的操作。您可以定义的规则操作之一是SMTPRelay 操作，如果您选择此操作，则电子邮件将传递到您的 SMTP 中继中定义的外部 SMTP 服务器。

例如，你可以使用该SMTPRelay 操作将电子邮件从你的入口端点发送到本地的 Microsoft Exchange 服务器。您可以将 Exchange 服务器设置为具有只能使用特定凭证访问的公共 SMTP 端点。创建 SMTP 中继时，输入 Exchange 服务器的服务器名称、端口和凭据，并为 SMTP 中继指定一个唯一的名称，比如 “” RelayToMyExchangeServer。然后，您在入口端点的规则集中创建一条规则，上面写着：“当发件人地址包含 'gmail.com' 时，使用名为的SMTP中继执行SMTPRelay 操作”。RelayToMyExchangeServer

现在，当来自 gmail.com 的电子邮件到达您的入口端点时，该规则将触发该SMTPRelay 操作，并使用您在创建 SMTP 中继时提供的凭据联系您的 Exchange 服务器，并将电子邮件传送到您的 Exchange 服务器。因此，从 gmail.com 收到的电子邮件将中继到您的 Exchange 服务器。

必须先创建 SMTP 中继，然后才能在规则操作中指定该中继。下一节中的过程将指导您在 SES 控制台中创建 SMTP 中继器。

在 SES 控制台中创建 SMTP 中继

以下过程演示了如何在 SES 控制台中使用 SMTP 中继页面，来创建 SMTP 中继并管理已创建的中继。

使用控制台创建和管理 SMTP 中继

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航面板中，选择 Mail Manager 下的 SMTP 中继。
3. 在 SMTP 中继页面上，选择创建 SMTP 中继。
4. 在创建 SMTP 中继页面上，输入您的 SMTP 中继的唯一名称。
5. 根据您是要配置入站（未经验证）还是出站（已验证）SMTP 中继器，请按照相应的说明进行操作：

   Inbound

   配置入站 SMTP 中继

   1. 当使用 SMTP 中继作为入站网关，将 Mail Manager 处理的传入电子邮件路由到您的外部电子邮件环境时，您首先需要配置电子邮件托管环境。尽管每个电子邮件托管提供商都有自己独特的 GUI 和配置工作流程，但将它们配置为使用入站网关（例如您的 Mail Manager SMTP 中继）的原则却是相似的。

      为了帮助说明这一点，我们在以下各节中提供了如何配置 Google Workspaces 和 Microsoft Office 365 以使用您的 SMTP 中继作为入站网关的示例：

      - 设置 Google Workspaces
      - 设置 Microsoft Office 365

      > **Note**
      >
      > 确保目标收件人目的地的域是 SES 验证的域身份。例如，如果您想向收件人 abc@example.com 和 support@acme.com 发送电子邮件，则 example.com 和 acme.com 这两个域都需要在 SES 中进行验证。如果收件人域未经验证，SES 将不会尝试将电子邮件传送到公共 SMTP 服务器。有关更多信息，请参阅 the section called “创建和验证身份”。

2. 将 Google Workspaces 或 Microsoft Office 365 配置为使用入站网关后，请输入公共 SMTP 服务器的主机名，根据您的提供商使用以下相应值：

   - Google Workspaces：`aspmx.l.google.com`
   - Microsoft Office 365：`<your_domain>.mail.protection.outlook.com`

     将域名中的圆点替换为“-”。例如，如果您的域名是 acme.com，则需要输入 `acme-com.mail.protection.outlook.com`

3. 输入公共 SMTP 服务器的端口号 25。
4. 将“身份验证”部分留空（请勿选择或创建密钥 ARN）。

Outbound

配置出站 SMTP 中继

1. 输入您希望中继连接到的公共 SMTP 服务器的主机名。
2. 输入公共 SMTP 服务器的端口号。
3. 从密钥 ARN 中选择一个密钥，为您的 SMTP 服务器设置身份验证。如果您选择之前创建的密钥，则该密钥必须包含以下创建新密钥步骤中指示的策略。

   - 您可以通过选择新建来创建新密钥， AWS Secrets Manager 控制台将打开，您可以在其中继续创建新密钥：

     a. 在密钥类型中，选择其他密钥类型。
     b. 以密钥/值对的形式输入以下密钥和值：

密钥	值
`username`	my_username
`password`	my_password

> **Note**
>
> 对于这两个密钥，您只能输入所示的 `username` 和 `password`（其他任何内容都将导致身份验证失败）。对于相应的值，请分别输入您自己的用户名和密码。

c. 选择添加新密钥以在加密密钥中创建 KMS 客户托管密钥 (CMK) — AWS KMS 控制台将打开。

d. 在客户自主管理型密钥页面上，选择创建密钥。

e. 保留配置密钥页面上的默认值，然后选择下一步。

f. 在别名中输入密钥的名称（您可以选择添加描述和标签），然后单击下一步。

g. 在密钥管理员中，选择您想要允许管理密钥的任何用户（您自己除外）或角色，然后选择下一步。

h. 在密钥用户中，选择想要允许使用密钥的任何用户（您自己除外）或角色，然后选择下一步。

i. 将 KMS CMK 策略 复制并粘贴到 `"statement"` 级别的密钥策略 JSON 文本编辑器中，将其作为额外声明添加进去，并用逗号进行分隔。将区域和账户替换为您自己的信息。

j. 选择完成。

k. 选择打开 AWS Secrets Manager 存储新密钥页面的浏览器选项卡，然后选择加密密钥字段旁边的刷新图标（圆形箭头），然后在该字段内单击并选择您新创建的密钥。

l. 在配置密钥页面上的密钥名称字段中输入名称。

m. 在资源权限中，选择编辑权限。

n. 将 密钥资源策略 复制并粘贴到资源权限 JSON 文本编辑器中，然后将区域和账户替换为您自己的信息。（请务必删除编辑器中的所有示例代码。）

o. 选择保存，然后选择下一步。

p. （可选）配置轮换，然后选择下一步。

q. 查看您的新密钥并选择存储。

r. 选择浏览器中打开 SES C reate SMTP 中继页面的选项卡，然后选择刷新列表，然后在 Secret AR N 中选择新创建的密钥。

6. 选择创建 SMTP 中继。

7. 您可以从 SMTP 中继页面查看和管理已经创建的 SMTP 中继。如果要删除某个 SMTP 中继，请选择其单选按钮，然后选择删除。
8. 要编辑 SMTP 中继，请选择其名称。在详细信息页面上，您可以更改中继的名称、外部 SMTP 服务器的名称、端口和登录凭证，方法是选择相应的编辑或更新按钮，然后选择保存更改。

设置 Google Workspaces 以使用入站（未经验证的）SMTP 中继

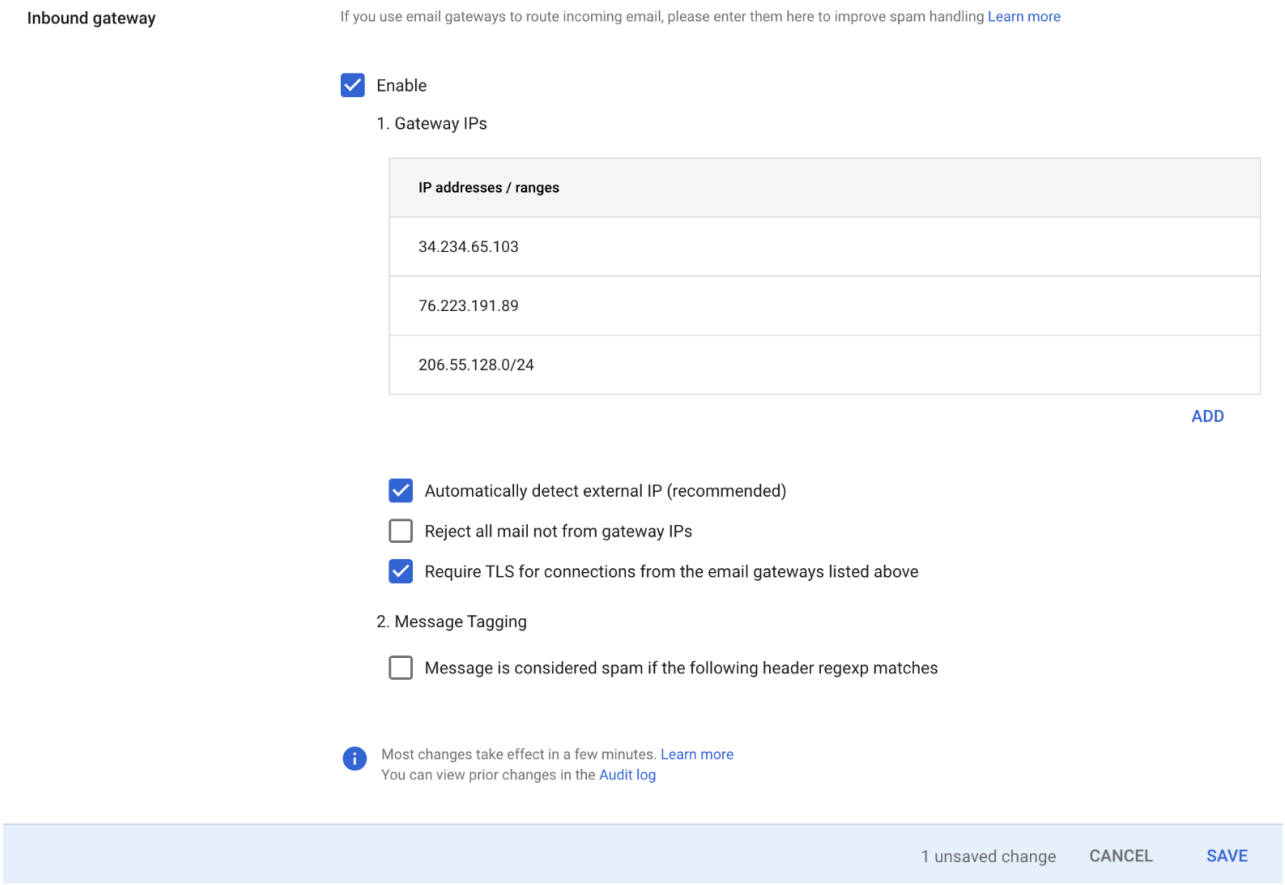
以下演练示例向您展示了如何设置 Google Workspaces 以使用 Mail Manager 入站（未经验证）SMTP 中继。

先决条件

- 有权访问 Google 管理员控制台（Google 管理员控制台 >“应用程序”>“Google Workspace”>“Gmail”）。
- 有权访问域名称服务器（托管将用于 Mail Manager 设置的域的 MX 记录）。

设置 Google Workspaces 以使用入站 SMTP 中继

- 将 Mail Manager IP 地址添加到入站网关配置中
  a. 在 Google 管理员控制台中，转到应用程序 > Google Workspace > Gmail。
  b. 选择垃圾邮件、网络钓鱼和恶意软件，然后转到入站网关配置。
  c. 启用入站网关，并使用以下详细信息对其进行配置：



- 在 Gateway 中 IPs，选择添加，然后从 SMTP 中继 IP 范围表中添加 IPs 特定于您所在地区的入口终端节点。
- 选择自动检测外部 IP。
- 选择要求来自上述电子邮件网关的连接使用 TLS。
- 选择对话框底部的保存以保存配置。保存后，管理员控制台会将入站网关显示为已启用。

设置 Microsoft Office 365 以使用入站（未经验证）SMTP 中继

以下演练示例向您展示了如何设置 Microsoft Office 365 以使用 Mail Manager 入站（未经验证）SMTP 中继。

先决条件

- 有权访问 Microsoft 安全管理中心（Microsoft 安全管理中心 >“电子邮件与协作”>“策略和规则”>“威胁策略”）。
- 有权访问域名称服务器（托管将用于 Mail Manager 设置的域的 MX 记录）。

设置 Microsoft Office 365 以使用入站 SMTP 中继

1. 将 Mail Manager IP 地址添加到允许列表

   a. 在 Microsoft 安全管理中心中，转到电子邮件与协作 > 策略和规则 > 威胁策略。

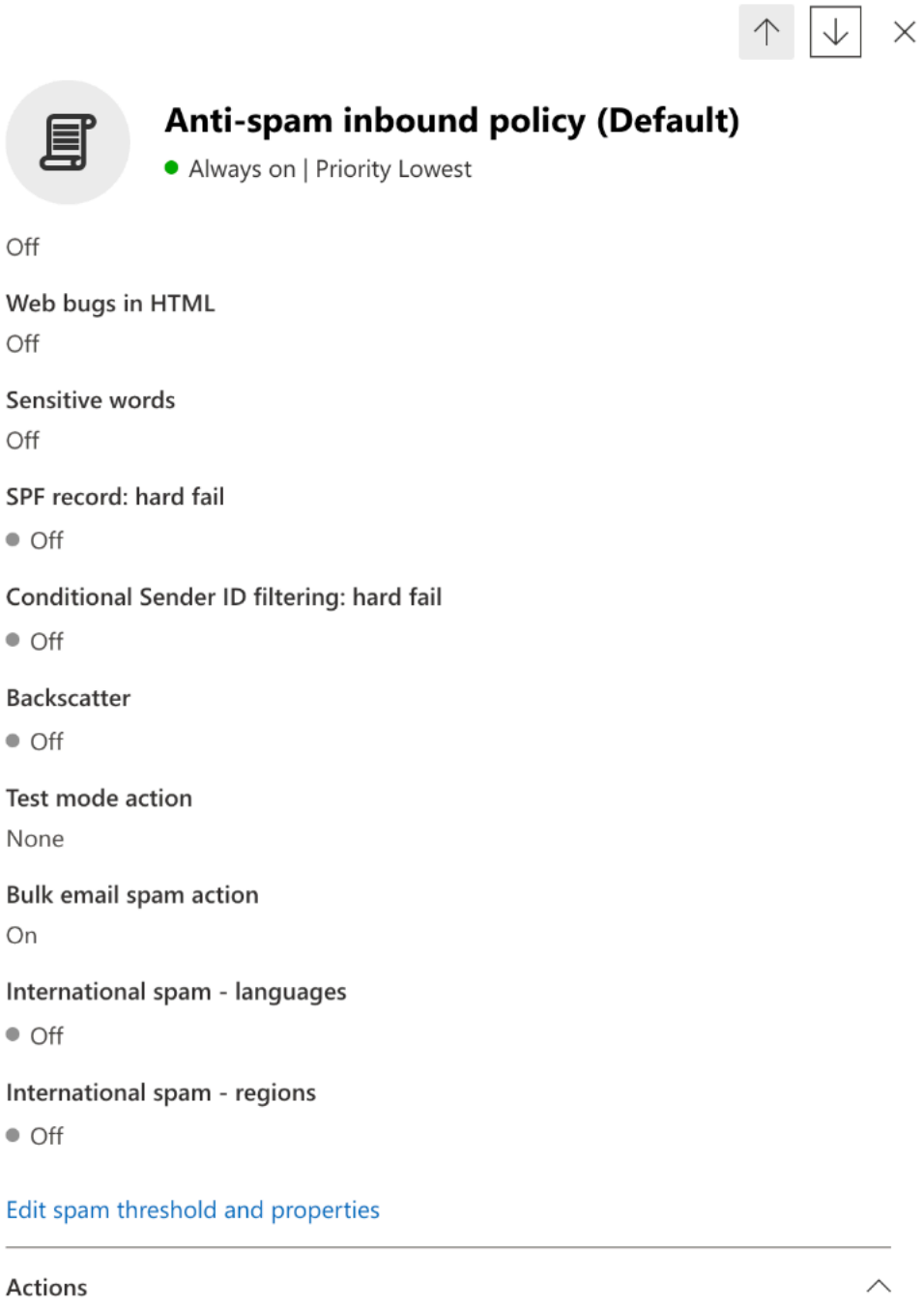
   b. 在策略下，选择反垃圾邮件。

   c. 选择连接筛选策略，然后选择编辑连接筛选策略。

      - 在 “始终允许来自以下 IP 地址或地址范围的消息” 对话框中，从 SMTP 中继 IP 范围表中添加 IPs 特定于您所在地区的入口端点。
      - 选择保存。

   d. 返回反垃圾邮件选项，并选择反垃圾邮件入站策略。

      - 在对话框底部，选择编辑垃圾邮件阈值和属性：



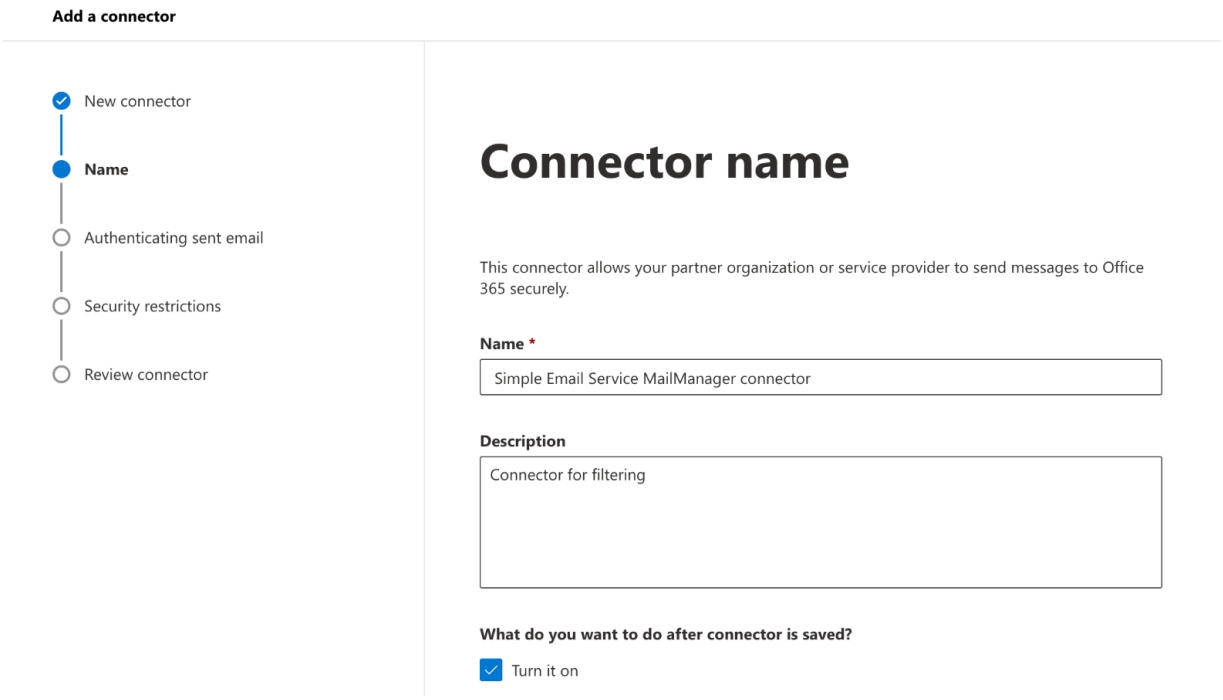
- 滚动到标记为垃圾邮件，并确保 SPF 记录：硬故障设置为关。
- 选择保存。

2. 增强的筛选配置（推荐）

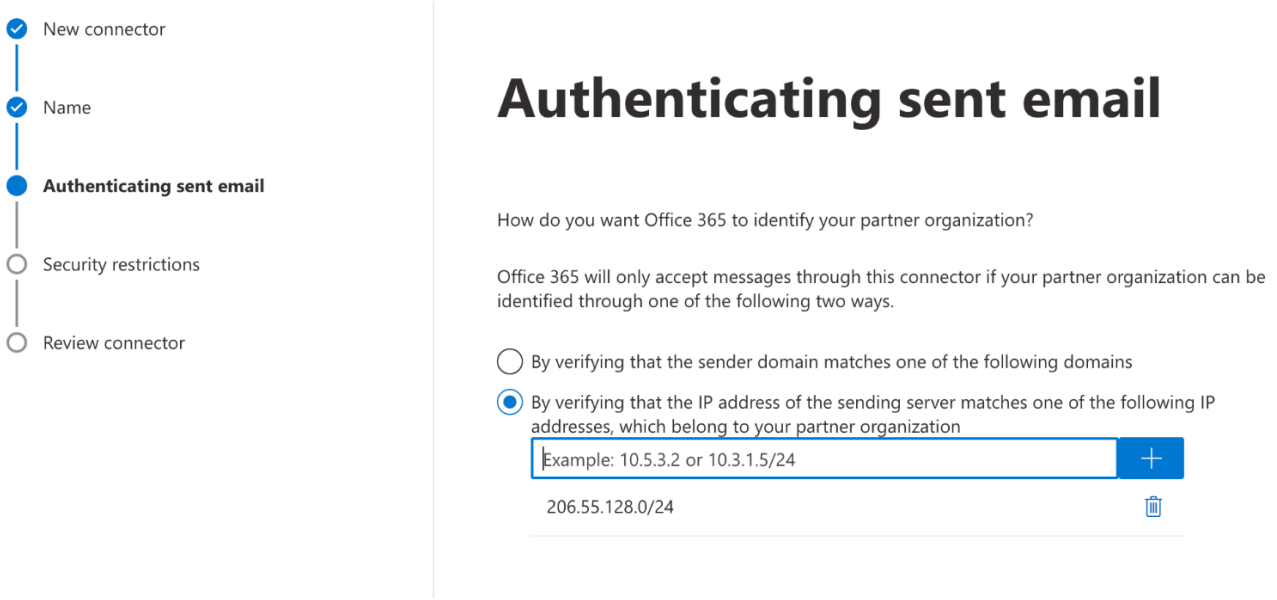
   此选项将允许 Microsoft Office 365 在 SES Mail Manager 收到邮件之前正确识别原始连接 IP。

   a. 创建入站连接器

- 登录到新的 Exchange 管理中心，然后转到邮件流 > 连接器。
- 选择添加连接器。
- 在连接来源中，选择合作伙伴组织，然后选择下一步。
- 按如下所示填写各字段：
  - 名称 – Simple Email Service Mail Manager 连接器
  - 描述 – 用于筛选的连接器



- 选择下一步。
- 在 “验证已发送的电子邮件” 中，选择 “通过验证发送服务器的 IP 地址是否与属于您的合作伙伴组织的 IP 地址之一匹配”，然后从 SMTP 中继 IP 范围表中添加 IPs 特定于您所在地区的入口端点。

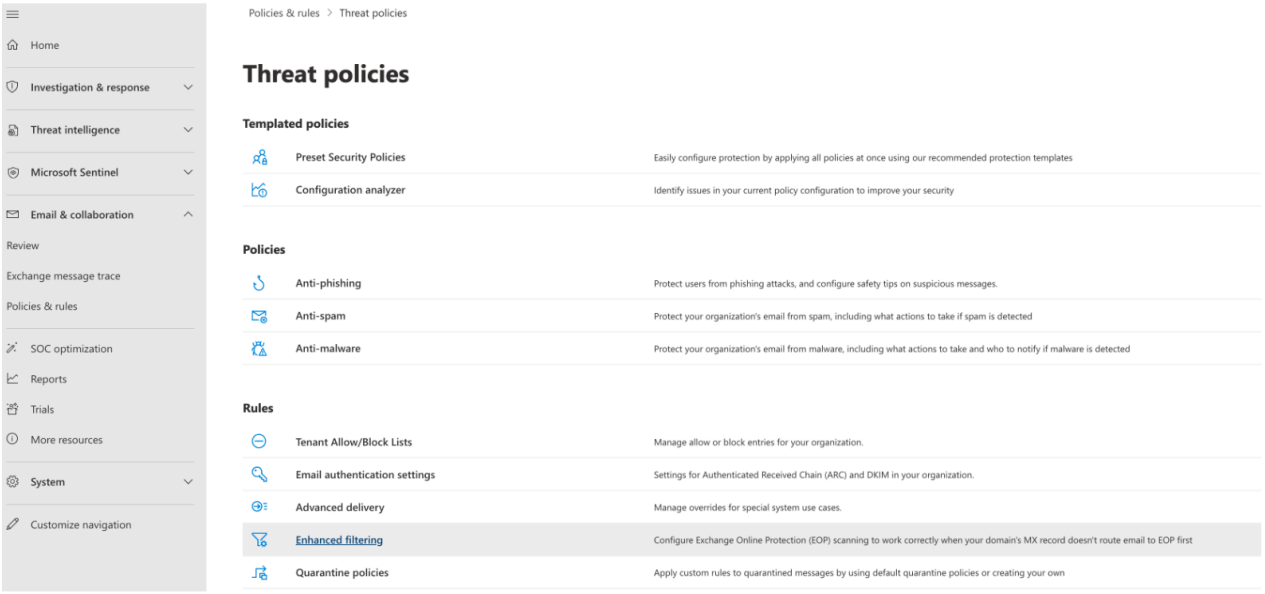


- 选择下一步。
- 在安全限制中，接受默认的如果电子邮件不是通过 TLS 发送的，则拒绝电子邮件设置，然后选择下一步。
- 检查您的设置，然后选择创建连接。

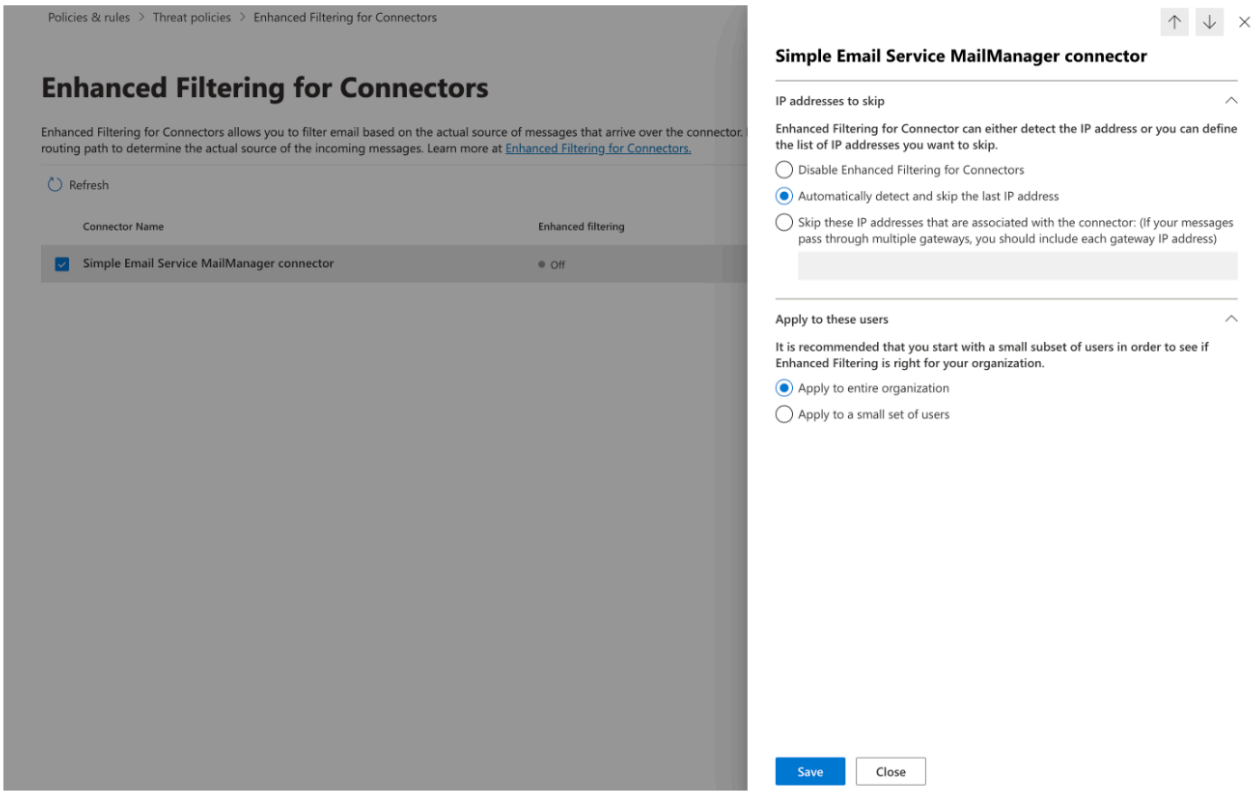
b. 启用增强的筛选

现在，入站连接器已配置完毕，您需要在 Microsoft 安全管理中心启用该连接器的增强筛选配置。

- 在 Microsoft 安全管理中心中，转到电子邮件与协作 > 策略和规则 > 威胁策略。
- 在规则下，选择增强筛选。



- 选择您之前创建的 Simple Email Service Mail Manager 连接器，编辑其配置参数。
- 选择自动检测并跳过最后一个 IP 地址和应用于整个组织。



- 选择保存。

地址清单

地址列表是 Mail Manager 的一项功能，允许您创建和管理电子邮件地址和域列表，这些列表可以在流量策略和规则集中使用，根据邮件的收件人或发件人是否属于特定列表来处理传入的邮件。地址列表可以更精细地控制电子邮件流，并有助于简化复杂电子邮件路由场景的管理。

什么是地址列表？

地址列表是电子邮件地址和域名的容器，可用于筛选和处理电子邮件。它们提供了一种便捷的方法来对相关地址进行分组，并共同应用路由规则和流量策略。

地址列表的主要用例包括：

- 用于阻止已知垃圾邮件发件人或域名的拒绝名单
- 允许列表用于确保来自可信发件人的投递
- 收件人验证可尽早拒绝发送给不存在的收件人的电子邮件
- 基于角色的路由，用于根据收件人角色应用不同的规则
- 基于群组的策略，用于为特定用户组强制执行策略

地址列表的工作原理

SES 中的地址列表允许您创建和维护电子邮件地址和域的集合，从而简化了电子邮件管理。创建后，这些列表将通过流量策略和规则集成到您的电子邮件工作流程中。

当 SES 处理电子邮件时，它会检查相关的地址列表以确定发件人或收件人是否为成员。然后，根据此成员资格以及您配置的策略和规则，SES 会采取适当的措施，例如路由、筛选或拒绝电子邮件。此过程可以对您的电子邮件流量进行高效、精细的控制。

设置地址列表

主题

- 创建和填充地址列表
- 管理地址列表

创建和填充地址列表

在控制台中创建地址列表的一部分是用一个或多个地址填充该列表。使用邮件管理器 APIs，您可以创建空地址列表并在以后填充它们。本节将通过控制台过程和 AWS CLI 示例向您展示如何执行这两种操作。

要创建和填充地址列表，请执行以下操作：

1. 打开 SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在 “邮件管理器” 下的导航窗格中，选择 “地址列表”。
3. 选择 “创建地址列表”，然后在 “地址列表名称” 字段中输入名称。
4. 选择 “手动输入” 或 “批量上传”，然后按照相应的步骤操作：
5. 手动输入-在控制台中输入一个或多个电子邮件地址或域名。

   如果使用星号 (*) 通配符，则以下格式适用：

   1. 地址中*只允许有一个：
      - 如果输入的是电子邮件地址，则*应在 @ 之前或之后。
      - 当*位于局部部分时，局部部分可以为零或 3 到 19 个字符，不包括*.
      - 在域中*时，子域级别可以为 2 到 9，不包括。*
   2. 有效的通配符格式示例：
      - *.domain1.com 到 *.domain8.domain7... domain1.com
      - * @domain .com
      - 123* @domain .com 至 12345678901234567 89* @domain .com
      - local@*.domain1.com 到 local@*.domain8.domain7... domain1.com
6. 对于批量上传-选择 “选择文件”，然后从计算机中选择包含要上传的地址的 CSV 或 JSON 文件。

   对每种文件类型使用示例中显示的格式：

   1. CSV 文件示例（请注意address，标题为必填项。）：

      ```
      address
      user1@domain.com
      user2@*.domain.com
      *@domain.com
      ```

   2. JSON 文件示例：

```
{
  "items": [
    {
      "address": "user1@domain.com"
    },
    {
      "address": "user2@*.domain.com"
    },
    {
      "address": "*@domain.com"
    }
  ]
}
```

7. 添加完地址或选择了批量文件后，选择创建地址列表。

使用 AWS CLI：

创建地址列表

```
aws mailmanager create-address-list --address-list-name "MyDenyList"
```

填充地址列表：

- 单次上传

```
aws mailmanager register-member-to-address-list \
               --address-list-id al-123456789abc \
               --address "user@example.com"
```

- 批量上传

  要进行批量上传，您首先必须创建一个指定 CSV 或 JSON 格式的导入任务：

```
aws mailmanager create-address-list-import-job \
               --address-list-id "al-123456789abc" \
               --name "MyImportJob" \
               --import-data-format ImportDataType=CSV
```

这将返回任务 ID 和预签名 URL。使用此预签名 URL 将您的 CSV 或 JSON 文件上传到 S3 存储桶，如以下示例所示，使用 curl 命令：

```
curl -X PUT -T "/path/to/file" "pre-signed URL"
```

上传后，使用上一个命令中返回的任务 ID 启动导入任务：

```
aws mailmanager start-address-list-import-job --job-id "job-123456789"
```

管理地址列表

您可以根据需要更新、查看和删除地址列表。

主题

- 更新地址列表
- 查看地址列表详情
- 删除地址列表

更新地址列表

您可以通过添加或删除地址来更新地址列表，也可以选择添加或删除标签。

要更新地址列表，请执行以下操作：

1. 在“地址列表”页面上，选择要编辑的地址列表的名称。
2. 要添加地址，请选择添加电子邮件地址，然后继续使用手动输入或批量上传的方法，如中所述创建和填充地址列表。
3. 要删除地址，请选中要删除的每个地址旁边的复选框，然后选择删除电子邮件地址并确认删除。
4. （可选）通过选择“管理标签”在地址列表中添加或移除标签。

使用 AWS CLI：

Add

```
aws mailmanager register-member-to-address-list \
```

```
                --address-list-id al-123456789abc \
                --address "user@example.com"
```

Remove

```
aws mailmanager deregister-member-from-address-list \
                --address-list-id al-123456789abc \
                --address "user@example.com"
```

查看地址列表详情

要查看地址列表详细信息，请执行以下操作：

- 在“地址列表”页面上，选择地址列表的名称以查看其详细信息。

使用 AWS CLI：

```
aws mailmanager list-members-of-address-list --address-list-id al-123456789abc
```

删除地址列表

要删除地址列表，请执行以下操作：

1. 在“地址列表”页面上，选择要删除的地址列表旁边的单选按钮，然后选择“删除”。
2. 键入“确认”，然后键入“删除”，确认删除列表。

使用 AWS CLI：

```
aws mailmanager delete-address-list --address-list-id al-123456789abc
```

在流量策略和规则集中使用地址列表

地址列表可用于流量策略声明和规则条件，根据列表成员资格处理电子邮件，从而控制电子邮件流。以下各节提供了在每个流量策略和规则集中使用的地址列表的示例。

主题

- 在流量策略声明中使用地址列表

- 在规则中使用地址列表

在流量策略声明中使用地址列表

当您创建流量策略声明的条件以允许或拒绝电子邮件进入您的入口端点时，可以选择地址列表。

以下控制台过程及其 AWS CLI 等效程序显示了一个创建策略声明的示例，如果收件人位于指定地址列表中，则允许消息进入您的入口端点。

要在流量策略声明中使用地址列表，请执行以下操作：

1. 创建新的流量策略或编辑现有流量策略，如中所述 创建流量策略和策略声明（控制台）
2. 在“策略语句”容器中，选择“允许”，以便在满足语句的条件时执行操作。
3. 按如下方式构建语句的条件：
   - 在“协议”字段中选择“收件人地址”。
   - 在“操作员”字段中选择“在地址列表中”。
   - 在“值”字段中选择地址列表的名称。
4. 虽然这只是一个示例，但您可以在任何地址列表中添加更多策略条件，这些条件可以基于各种运算符。

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"
      }
  }]
}]'
```

在规则中使用地址列表

当您为其中一个规则集中使用的规则创建条件以触发该规则的操作时，可以选择地址列表。

以下控制台过程及其 AWS CLI 等效程序显示了创建规则的示例，该规则将在收件人位于指定地址列表中时调用丢弃操作。

要在规则条件中使用地址列表，请执行以下操作：

1. 创建新规则或编辑现有规则，如中所述 创建规则集和规则（控制台）
2. 在规则条件容器中，按如下方式构建规则的条件。
   - 在“选择属性”字段中选择“收件人地址”。
   - 在“选择运算符”字段中选择“在地址列表中”。
   - 在“值”字段中选择地址列表的名称。
3. 在“操作”容器中，选择“添加新动作”，然后选择“删除操作”。
4. 虽然这只是一个示例，但您可以添加更多规则条件，这些条件可以基于各种运算符的任意地址列表来执行各种操作。

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

最佳实践和注意事项

- 请注意列表的大小，因为列表过大可能会影响性能。
- 地址列表是特定于账户的，只能在同一个 AWS 账户中使用。
- 目前不支持嵌套地址列表。
- 每个区域最多支持 100 个地址列表。
- 每个地址列表最多支持 100,000 个地址。

消息存档

电子邮件存档为您提供了一种存档进入入口端点的指定电子邮件类型或通过配置集发送的电子邮件的方法，还提供了一种通过丰富的高级搜索过滤器查找存档邮件的方法，并能够导出结果。

电子邮件存档通过将数据存储在持久且安全的长期存储中，来保存和保护您的电子邮件，并为您提供一种快速搜索和存档电子邮件的方法。它提供全天候的企业级存档服务，而不会增加邮箱服务器的存储需求。

存档可以包含已发送和已接收电子邮件的组合：

- 已发送的电子邮件存档-当您通过启用存档选项的配置集发送（出站）电子邮件时，使用该配置集发送的所有电子邮件都将存档到您指定的电子邮件存档中。
- 收到的电子邮件存档 — 当您的入口端点收到（入站）电子邮件时，它会使用流量策略来确定要屏蔽或允许哪些电子邮件。您允许传入的电子邮件会传递给一个规则集，该规则集应用条件规则来执行您为特定类型的电子邮件定义的操作。您可以定义的规则操作之一是存档操作，如果您选择此操作，则电子邮件将存档到您指定的电子邮件存档中。

必须先创建档案，然后才能在规则操作或配置集中对其进行指定。下一节中的过程将指导您在 SES 控制台中创建存档。

在 Amazon SES 控制台中使用电子邮件存档

SES 控制台中的电子邮件存档页面包含以下四个交互式表：搜索存档、搜索历史记录、导出历史记录和管理存档，您使用它们在存档中搜索电子邮件、导出结果以及管理存档。以下过程为每个表都提供了说明。

使用电子邮件存档页面搜索、导出和管理您的存档

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航面板中，选择 Mail Manager 下的电子邮件存档。
3. 电子邮件存档页面包含以下四个表：搜索存档、搜索历史记录、导出历史记录和管理存档。有关每个表的特定说明，请选择下面的相应选项卡：

Search archive

搜索存档是一个交互式表，您可以通过其丰富的筛选器和日期集来搜索和查找已存档的消息。这些详细的搜索条件可协助您从特定的一封电子邮件到匹配更广泛类别的多封电子邮件中查找任何内容。符合搜索条件的消息可以单独下载，也可以批量导出到 S3 存储桶。

搜索、下载或导出存档的电子邮件

1. 在电子邮件存档页面上，选择搜索存档选项卡以显示搜索存档表。
2. 在存档字段内单击，然后从列表中选择一个存档，接着选择搜索，或者使用以下步骤优化搜索范围。
3. 选择日期范围字段以展开搜索的日期范围选项：
   - 相对范围（默认）- 选择与所需天数相对应的单选按钮，或通过选择时间单位和最多 30 天的日期范围来选择自定义范围。
   - 绝对范围 - 输入开始日期和结束日期（如果需要，还可以输入时间），日期范围最多为 30 天。

> **Note**
>
> - 在存档中搜索时，每次搜索的日期范围限制为 30 天。例如，如果要搜索 6 月 1 日至 7 月 31 日的消息，则必须将其分为三个搜索，如下所示：

1. 六月的 30 天。
2. 七月的前 30 天。
3. 七月的第 31 天。

- 对于相对范围日期，最后一天在午夜结束。例如，如果您选择了 Last 7 days（过去 7 天），则第七天将是昨天，在午夜结束。

4. （可选）选择筛选器字段，从以下筛选器中进行选择：发件人、收件人、抄送、主题行和有附件 - 以下属性适用：

   - 您可以创建最多 10 个筛选器。
   - 通过单击筛选器可以对其进行编辑，或者通过选择 X 将其删除。

5. 选择搜索，符合搜索条件的存档电子邮件将填充到搜索结果表中。

   - 邮件 ID 列在默认情况下处于隐藏状态，但可以通过选择齿轮图标以自定义查看表的方式来显示该列。
   - 您执行的每一次搜索都会使用唯一的搜索 ID 自动保存，并将列在搜索历史记录表中。

6. 要查看消息的文本及其信封和标头信息，请选择消息的单选按钮，然后选择查看详细信息，打开消息详细信息侧边栏。
7. 要创建消息的本地文件，请选择消息的单选按钮，然后选择下载消息。
8. 选择导出至 S3，即可将筛选后的搜索结果保存到 Amazon S3 存储桶中。

   a. 如果您知道要使用的 S3 存储桶的 URI，请在 S3 URI 字段中输入；否则，请选择浏览 S3，然后在 S3 页面上选择一个要使用 S3 存储桶和文件夹。

   b. （可选）您可以通过在 KMS AWS KMS 密钥 ARN 字段中输入自己的密钥或选择创建新密钥来加密导出的消息。否则，加密将设置为目标 S3 存储桶上正在使用的任何方法（即使没有任何方法）。

   c. 选择导出，在筛选搜索中找到的所有消息都将作为单个文件保存到您选择的 S3 文件夹中。

> **Note**
>
> 虽然您的存档中可以包含的消息数量没有限制，但搜索结果表中的搜索结果限制在 1000 行以内。

Search history

此表中列出了您的搜索历史记录，以便您可以恢复结果集或访问之前创建的复杂筛选器集。您还可以通过编辑筛选器和日期，在原始搜索的基础上创建新的搜索。任何新的搜索都会自动使用唯一的搜索 ID 保存，并将列在此表中。

查看和处理您之前的搜索

1. 在电子邮件存档页面上，选择搜索历史记录选项卡以显示搜索历史记录表，其中列出了所有已存档电子邮件搜索的历史记录，最新的搜索记录位于顶部。此表会在您第一次访问时加载数据，如果您切换选项卡并返回，请使用刷新图标以检索最新数据。
2. 在存档字段内单击，然后从列表中选择一个存档，属于该存档的所有搜索都将填充到表中。您可以在以下步骤中查看单个搜索并执行更多操作。
3. 选择上一次搜索的单选按钮，然后选择查看搜索结果以恢复其原始搜索结果，此时，搜索存档页面将打开，显示用于原始搜索的筛选器集和日期范围，以及之前根据该条件找到的所有消息。您可以通过以下方式扩展原始搜索结果：
   - 通过修改日期范围和筛选器，然后单击搜索，即可创建新的搜索。
   - 您执行的任何新搜索都将使用唯一的搜索 ID 自动保存，并将列在搜索历史记录表中。

Export history

此表中列出了您的导出历史记录，以便在 S3 控制台中轻松访问导出文件夹的内容。

查看最新的导出

1. 在电子邮件存档页面上，选择导出历史记录选项卡以显示导出历史记录表，其中列出了您在过去 30 天内导出到 S3 存储桶的所有存档电子邮件搜索。此表会在您第一次访问时加载数据，如果您切换选项卡并返回，请使用刷新图标以检索最新数据。
2. 如果导出的状态为已排队、正在预处理或正在处理，则可以通过选择取消将其取消。
3. 选择 S3 URI 以在 S3 控制台中打开导出的存储桶文件夹，您可以在其中查看所包含的文件。

Manage archives

此表列出了您的存档，您可以在其中选择创建新存档、搜索特定的存档并查看其详细信息、编辑存档或删除存档。

创建和管理存档

1. 在电子邮件存档页面上，选择管理存档选项卡以显示存档表，其中列出了您的所有电子邮件存档。此表会在您第一次访问时加载数据，如果您切换选项卡并返回，请使用刷新图标以检索最新数据。
2. 要搜索特定的存档，请在存档字段中键入相应内容。
3. 要查看某个存档的详细信息，请在存档名称列中选择其名称。
4. 要创建存档，请选择创建存档。

   a. 在存档名称字段中输入唯一的名称。

   b. （可选）在保留期字段中选择一个保留期，以覆盖默认的 180 天保留期。

   c. （可选）您可以通过在 KMS AWS KMS 密钥 ARN 字段中输入自己的密钥或选择创建新密钥来加密您的档案。

   选择创建存档。
5. 现在您已经创建了存档，可以在规则集中使用它来存档收到的（入站）电子邮件，或者在配置集中将其指定为存档已发送（出站）的电子邮件。
6. 要编辑某个存档，请选择其单选按钮，然后选择编辑。

   a. 在存档名称字段中编辑或更改名称。

   b. 在保留期字段中更改保留期。

   选择更新存档。
7. 要删除某个存档，请选择其单选按钮，然后选择删除。

   - 在确认字段中键入 `delete`，然后选择删除。

     存档表中的存档状态将切换为待删除，并将在 30 天后自动删除。

电子邮件插件

电子邮件插件是一组来自 SES 认可的提供商的专业安全工具，可用于管理您允许传入入口端点的电子邮件类型，并确定要对特定类型的电子邮件执行的操作。这些工具是经过认证的安全情报和执行解决方案，可以随时集成到您的电子邮件工作流程中，并且可以直接从 Mail Manager 控制台激活。

这些插件提供了从经过审核的电子邮件安全解决方案中进行选择的灵活性，这些解决方案适合您的个人应用场景，可以按计量价格的方式使用，而无需购买可能并不适用于您任何需求的大型单一产品解决方案。电子邮件插件根据每个工作负载扩展其核心威胁情报和安全执行功能，因此无需猜测所需的容量。这些优势使您能够专注于应对电子邮件安全问题，并保持您组织的高服务标准。

您可以直接从 Mail Manager 控制台中的“电子邮件插件”页面了解有关每个插件的更多信息，在该页面上，您可以访问产品描述、主要优势和定价信息。确定要使用的插件后，只需从 Mail Manager 控制台订阅即可。订阅后，您可以选择将其作为流量策略条件，以确定允许传入入站端点的电子邮件，或者作为规则集条件，以确定对特定电子邮件执行的操作。对所有附加组件的主要支持由 AWS Mail Manager 控制台提供，也可以从 Mail Manager 控制台进行访问。

下一节中的步骤将引导您完成在 Mail Manager 控制台中订阅电子邮件插件的过程。

在 Mail Manager 控制台中订阅电子邮件插件

以下过程向您演示如何使用 Mail Manager 控制台中的电子邮件插件页面来订阅插件，以便在您的任何流量策略或规则集中使用。

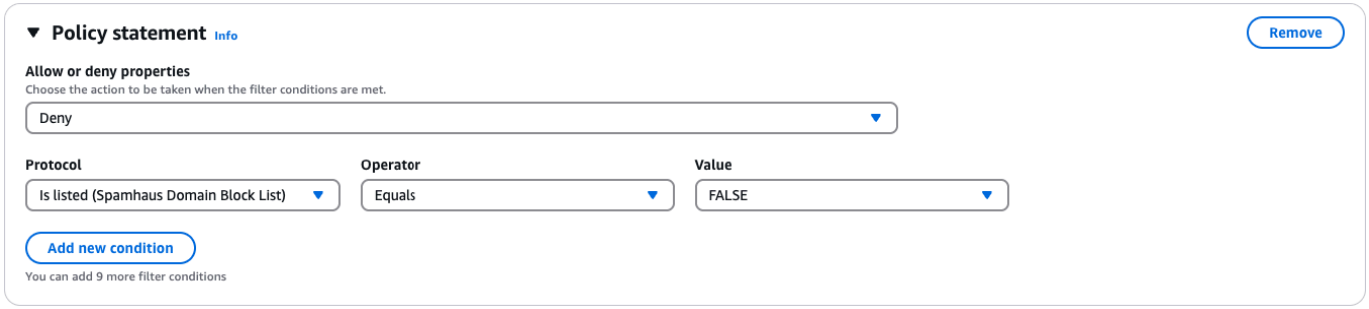
使用控制台订阅电子邮件插件

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在左侧导航面板中，选择 Mail Manager 下的电子邮件插件。
3. 在电子邮件插件页面上，选择任意插件卡的标题以打开其概述页面，您可以在其中详细了解其用途、主要优势和定价信息。如果您想使用此插件，请选择订阅。
   - 阅读显示的条款和条件，然后选中我接受复选框，接着选择订阅。
4. 一旦订阅了某个插件，您可以将其集成到您的电子邮件工作流程中，方法是选择将其作为流量策略条件，以拒绝或允许电子邮件传入您的入口端点，或者选择将其作为规则集条件，来确定对符合条件的消息执行的操作。

   以下示例描述了如何在策略声明条件和规则条件中使用插件：

   使用 Add On 的流量策略示例

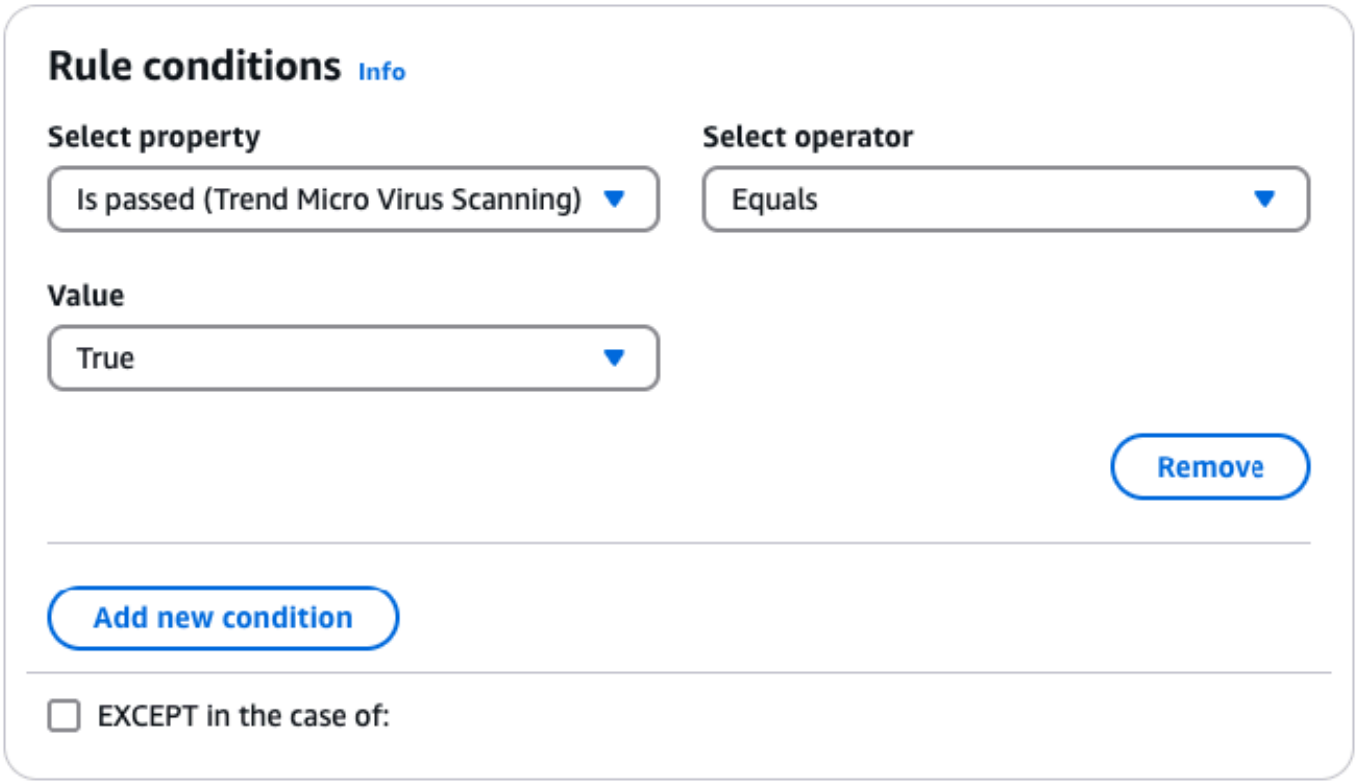
   在策略声明条件中使用 Spamhaus 域阻止列表插件，阻止来自 Spamhaus 中列出的域的电子邮件传入您的入口端点：



有关如何使用电子邮件插件创建流量策略和构建策略声明条件的详细信息，请参阅the section called “创建流量策略和策略声明（控制台）”。

使用 Add On 的规则条件示例

在规则条件下使用 Trend Micro 病毒扫描插件，来确定针对通过病毒扫描的电子邮件的规则操作：



有关如何使用电子邮件插件创建规则集和构建规则条件的详细信息，请参阅the section called “创建规则集和规则（控制台）”。

5. 要查看您订阅的任何插件的一般详细信息或访问支持，请在电子邮件插件页面上，选择其名称以打开其概述页面：

   - 在一般详细信息中，您可以查看订阅日期和插件的 Amazon 资源名称（ARN）。

- 选择支持选项卡，可访问指向 AWS Support 的链接。

6. 取消订阅插件

   a. 您必须先将其从条件中定义的所有流量策略或规则集中删除；否则，以下取消订阅步骤将失败。

   b. 在电子邮件插件页面上选择其名称以打开相应的概述页面，然后选择取消订阅。

   c. 在确认字段中键入 `confirm`，然后选择取消订阅。

Mail Manager 的权限策略

本章中的策略为使用 Mail Manager 所有不同功能所必需的策略提供了一个单一参考点。

在 Mail Manager 功能页面中，提供了链接，单击后会跳转到本页包含您使用该功能所需策略的相应部分。选择所需策略的复制图标，并按照相应功能说明中的指示进行粘贴。

以下策略允许您通过资源权限策略和 AWS Secrets Manager 策略使用 Amazon SES Mail Manager 中包含的不同功能。如果您不熟悉权限策略，请参阅 the section called “策略剖析”和 AWS Secrets Manager的权限策略。

入口端点的权限策略

创建入口端点需要同时使用本节中的两个策略。要了解如何创建入口端点以及在何处使用这些策略，请参阅the section called “创建入口端点（控制台）”。

入口端点的 Secrets Manager 密钥资源权限策略

要使 SES 能够通过入口端点资源访问密钥，需要使用以下 Secrets Manager 密钥资源权限策略。

JSON

```
{
    "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/*"
                }
            }
        }
    ]
}
```

入口端点的 KMS 客户自主管理型密钥（CMK）密钥策略

要使 SES 能够在使用您的密钥资源时使用您的密钥，需要使用以下 KMS 客户自主管理型密钥（CMK）密钥策略。

```
{
    "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 中继的权限策略

创建 SMTP 中继需要同时使用本节中的两个策略。要了解如何创建 SMTP 中继以及在何处使用这些策略，请参阅the section called “创建 SMTP 中继（控制台）”。

SMTP 中继的 Secrets Manager 密钥资源权限策略

要使 SES 能够通过 SMTP 中继资源访问密钥，需要使用以下 Secrets Manager 密钥资源权限策略。

JSON

```
{
    "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/*"
                }
           }
        }
    ]
}
```

SMTP 中继的 KMS 客户自主管理型密钥（CMK）密钥策略

要使 SES 能够在使用您的密钥资源时使用您的密钥，需要使用以下 KMS 客户自主管理型密钥（CMK）密钥策略。

JSON

```
{
    "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/*"
                }
            }
        }
    ]
}
```

电子邮件存档的权限策略

存档导出

IAM 身份调用者`StartArchiveExport`必须有权访问由以下策略配置的目标 S3 存储桶：

JSON

```
{
    "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/*"
        }
    ]
}
```

这是针对目标存储桶的策略。

JSON

```
{
    "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**
>
> 存档不支持混淆的副条件键（aws: SourceArn、aws: SourceAccount、aws: SourceOrg ID 或 aws:SourceOrgPaths）。这是因为 Mail Manager 的电子邮件存档在开始实际导出之前，会使用转发访问会话测试调用身份是否具有对导出目标存储桶的写入权限，从而防止出现混淆代理问题。

使用 KMS CMK 进行存档静态加密

IAM 身份调用`CreateArchive`者`UpdateArchive`必须能够通过以下策略访问 KMS 密钥 ARN：

JSON

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "kms:DescribeKey",
            "kms:Decrypt",
```

```
            "kms:GenerateDataKey"
        ],
        "Resource": "arn:aws:kms:us-west-2:111122223333:key/MyKmsKeyArnID"
    }
}
```

这是电子邮件存档所需的 KMS 密钥策略。

JSON

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

执行规则操作的权限和信任策略

SES 规则执行角色是一个 AWS Identity and Access Management (IAM) 角色，它授予规则执行权限以访问 AWS 服务和资源。在规则集中创建规则之前，必须使用允许访问所需 AWS 资源的策略创建一个 IAM 角色。SES 在执行规则操作时会代入此角色。例如，您可以创建一个规则执行角色，该角色具有将电子邮件消息写入 S3 存储桶的权限，作为在规则条件满足时要采取的规则操作。

因此，除了本节中执行每个特定规则操作所需的单个权限策略外，还需要以下信任策略。

JSON

```
{
    "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/*"
          }
        }
      }
    ]
  }
```

规则操作策略

- 写入 S3 规则操作的权限策略
- 传送到邮箱规则操作的权限策略
- 发送到互联网规则操作的权限策略
- Deliver to Q Business 规则操作的权限策略
- 发布到 SNS 规则操作的权限策略

写入 S3 规则操作的权限策略

要使用写入 S3 规则操作，将收到的电子邮件传送到 S3 存储桶，需要以下策略。

JSON

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

如果您对启用了服务器端加密的 S3 存储桶使用 AWS KMS 客户托管密钥，则需要添加 IAM 角色策略操作。"kms:GenerateDataKey*"使用前面的示例，将此操作添加到您的角色策略中，如下所示：

JSON

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

有关为 AWS KMS 密钥附加策略的更多信息，请参阅《AWS Key Management Service 开发人员指南》AWS KMS中的使用密钥策略。

传送到邮箱规则操作的权限策略

使用将收到的电子邮件发送到亚马逊 WorkMail账户的 “投递到邮箱” 规则操作需要以下策略。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": ["workmail:DeliverToMailbox"],
```

```
        "Resource": "arn:aws:workmail:us-
east-1:888888888888:organization/MyWorkMailOrganizationID>"
      }
    ]
  }
```

发送到互联网规则操作的权限策略

要使用发送到互联网规则操作，将收到的电子邮件发送到外部域，需要以下策略。

> **Note**
>
> 如果您的 SES 身份使用默认配置集，则还需要添加配置集资源，如以下示例所示。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": ["ses:SendEmail", "ses:SendRawEmail"],
        "Resource":[
        "arn:aws:ses:us-east-1:888888888888:identity/example.com",
        "arn:aws:ses:us-east-1:888888888888:configuration-set/my-configuration-
set"
        ]
      }
    ]
  }
```

Deliver to Q Business 规则操作的权限策略

使用 “Deliver to Q Business” 规则操作需要以下策略，该操作将收到的电子邮件发送到 Amazon Q Business 索引。

亚马逊 Q 企业政策：

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccessToQBusiness",
            "Effect": "Allow",
            "Action": [
                "qbusiness:BatchPutDocument"
            ],
            "Resource": [
                "arn:aws:qbusiness:us-
east-1:888888888888:application/ApplicationID/index/IndexID"
            ]
        }
    ]
}
```

Amazon Q Business 的 KMS 政策：

JSON

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

有关为 AWS KMS 密钥附加策略的更多信息，请参阅《AWS Key Management Service 开发人员指南》AWS KMS中的使用密钥策略。

发布到 SNS 规则操作的权限策略

使用 “发布到 SNS” 规则操作需要以下策略，该操作会将收到的电子邮件发送到 Amazon SNS 主题。

Amazon SNS 策略：

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccessToSNSTopic",
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-1:888888888888:MySnsTopic"
            ]
        }
    ]
}
```

亚马逊 SNS 的 KMS 政策：

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccessToKMSKeyForSNS",
            "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"
                    ]
                }
            }
        }
    ]
}
```

有关为 AWS KMS 密钥附加策略的更多信息，请参阅《AWS Key Management Service 开发人员指南》AWS KMS中的使用密钥策略。

邮件管理器日志

邮件管理器日志记录提供邮件管理器操作的详细可见性。日志记录功能可根据您配置的规则集和规则，跟踪从最初在入口端点收到消息到消息处理的消息流。

邮件管理器提供以下资源的日志记录：

- 入口端点
- 规则集

Mail Manager 使用亚马逊 CloudWatch 日志服务传送日志，日志可以传送到以下任何目的地：日CloudWatch 志、Amazon S3 或 Amazon Data Firehose。

设置邮件管理器日志传送

工作日志传输由三个元素组成：

- DeliverySource— 表示发送日志的资源的逻辑对象，可以是入口端点或规则集。
- DeliveryDestination— 表示实际传送目标的逻辑对象（CloudWatch 日志、S3 或 Firehose）。
- 交付-将传送源连接到传送目的地。

本节将说明如何创建这些对象以及使用 Mail Manager 日志记录所需的必要权限。

先决条件

在设置 Mail Manager 日志记录之前，请确保：

1. 您已经创建了 Ingress 端点或规则集。
2. 您拥有必要的 CloudWatch 日志和 SES Mail Manager 权限，可以将邮件管理器资源中的日志发送到其送达目的地。

所需的权限

您需要按照《A mazon Logs 用户指南》的 “需要额外权限的日志记录 [V2]” 部分中的说明设置销售 CloudWatch 日志权限，并应用与您的配送目的地相对应的权限：

- 发送到日志的 CloudWatch 日志
- 发送到 Amazon S3 的日志
- 已发送到 Firehose 的日志

此外，Mail Manager 需要以下用户权限才能配置日志传送：

- `ses:AllowVendedLogDeliveryForResource`— 需要允许 Mail Manager 代表你将 CloudWatch 日志发送到特定资源的日志，如示例所示：

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

在 SES 控制台中启用日志记录

要使用控制台启用邮件管理器资源的日志记录，请执行以下操作：

1. 打开 SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在 Mail Manager 下的导航窗格中，选择入口端点或规则集，然后选择要启用日志记录的特定资源。
3. 在资源的详细信息页面上，展开添加日志传输，然后选择传送到CloudWatch 日志、S3 或 Firehose。
4. 在所选目标的添加投递到对话框中，按照提示配置特定于目标类型的日志传送选项。
5. （可选）展开“其他设置”，自定义记录字段、输出格式、字段分隔符和其他特定于目标类型的参数。

使用 CloudWatch 日志 API 启用日志记录

要使用 CloudWatch 日志 API 为 Mail Manager 资源启用日志记录，您需要：

1. 使用创 DeliverySource 建 PutDeliverySource.
2. 使用创 DeliveryDestination 建 PutDeliveryDestination.
3. 通过使用将一个配送来源和一个配送目的地精确配对来创建配送CreateDelivery。

您可以在《A mazon Logs 用户指南》的 “需要额外权限的日志记录 [V2]” 部分中查看包含特定日志目标所需的所有权限的 IAM 角色和权限策略示例，并按照 CloudWatch 日志目标的 IAM 角色和权限策略示例进行操作，包括允许更新您的特定日志目标资源，例如CloudWatch 日志、S3 或 Fireh os e。

> **Note**
>
> 创建时 DeliverySource，`resourceArn`可以是入口端点 ARN 或规则集 ARN。视情况而定 DeliverySource，`logType`可能如下所示：
>
> - 入口端点 ARN — 或 `APPLICATION_LOGS TRAFFIC_POLICY_DEBUG_LOGS`
> - 规则集 ARN — `APPLICATION_LOGS`

解释日志

在 Mail Manager 处理收到的邮件时，可以使用这些日志进一步了解这些邮件的流向。

以下示例详细说明了每种资源和日志类型的不同日志字段：

日志示例

- 入口端点日志 — APPLICATION_LOGS
- 入口端点日志 — TRAFFIC_POLICY_DEBUG_LOGS
- 规则集日志 — APPLICATION_LOGS

入口端点日志 — **APPLICATION_LOGS**

日志是按每条消息生成的。

```
{
  "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**
>
> 仅为入口端点接受的消息创建日志。拒绝所有传入消息的入口端点不会发布任何应用程序日志。

CloudWatch 日志见解查询示例

查询来自 sender@domain.com 的消息：

```
fields @timestamp, @message, @logStream, @log
| filter smtp_mail_from like /sender@domain.com/
| sort @timestamp desc
| limit 10000
```

查询大小大于 5000 字节的消息：

```
fields @timestamp, @message, @logStream, @log
| filter message_size_bytes > 5000
| sort @timestamp desc
| limit 10000
```

入口端点日志 — **TRAFFIC_POLICY_DEBUG_LOGS**

日志是按收件人生成的。

```
{
    "resource_arn": "arn:aws:ses:us-east-1:1234567890:mailmanager-ingress-point/inp-
xxxxx" CPY,
    "event_timestamp": 1728562395042,
```

```
    "ingress_point_type": "OPEN" | "AUTH",
    "ingress_point_id": "inp-xxxx",
    "ingress_point_session_id": "xxxx",
    "traffic_policy_id": "tp-xxxx",
    "traffic_policy_evaluation": [
           // Array of policy evaluations
           {
                "action": "ALLOW" | "DENY",
                "conditions": [
                // Array of conditions
                {
                    "expression": {
                        "attribute": "RECIPIENT",
                        "operator": "CONTAINS",
                        "value": ["@domain.com", "@mydomain.com"]
                    },
                    "expressionResult": true | false
                }],
                "policyStatementMatched": true | false
            },
            // If no policy statement match then default action will be applied
            {
                "action": "ALLOW" | "DENY",
                "policyStatementMatched": true,
                "type": "DefaultAction"
            }
    ],
    "traffic_policy_verdict": "REJECT" | "ACCEPT",
    "sender_ip_address": "1.2.3.4",
    "smtp_mail_from": "someone@domain.com",
    "smtp_helo": "domain.com",
    "tls_protocol": "TLSv1.2",
    "recipient": "me@mydomain.com",
    "tls_cipher_suite": "TLS_AES_256_GCM_SHA384"
}
```

Note

- 系统会为入口端点的流量策略评估的所有消息创建日志，无论这些消息是被接受还是拒绝。
- 属于同一封邮件（在同一 SMTP 对话中）的所有收件人流量策略评估都有一个共同点`ingress_point_session_id`。此 ID 可用作关联标识符，因为要`message_id`等到接受消息之后才可用。

- `traffic_policy_evaluation`内容因您的配置而异，一旦确定判决，可能会提前终止。

CloudWatch 日志见解查询示例

查询来自 sender@domain.com 的消息：

```
fields @timestamp, @message, @logStream, @log
| filter smtp_mail_from like /sender@domain.com/
| sort @timestamp desc
| limit 10000
```

查询属于特定对象的消息`ingress_point_session_id`：

```
fields @timestamp, @message, @logStream, @log
| filter ingress_point_session_id = 'xxx'
| sort @timestamp desc
| limit 10000
```

查询被拒绝的消息：

```
fields @timestamp, @message, @logStream, @log
| filter traffic_policy_verdict = 'REJECT'
| sort @timestamp desc
| limit 10000
```

规则集日志 — **APPLICATION_LOGS**

日志是按每个操作的每条消息生成的。这意味着，每当规则集中的规则中的操作处理消息时，都会生成一条日志记录：

```
{
   "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— 与正在执行操作的规则条件相匹配的收件人。
- rule_index— 规则在规则集中的顺序。
- action_index— 规则中的动作顺序。
- action_status— 表示对给定消息执行操作的结果。
- action_failure— 表示操作的失败详细信息（仅在操作失败时适用）。例如，如果提供的角色没有足够的权限来执行操作。

此外，如果一条消息的规则条件不匹配，即该消息未被规则处理，则会发布一条日志，表明该消息已由规则集处理，但未对其执行任何操作：

```
{
   "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 日志见解查询示例

查询特定的消息 ID（显示通过规则集的消息流）：

```
fields @timestamp, @message, @logStream, @log
| filter message_id = 'message-id-123'
| sort @timestamp desc
| limit 10000
```

查询失败的 WRITE_TO_S3 操作：

```
fields @timestamp, @message, @logStream, @log
```

```
| filter action_metadata.action_name = 'WRITE_TO_S3'
    and action_metadata.action_status = 'FAILURE'
| sort @timestamp desc
| limit 10000
```

查询未被规则集的第二条规则处理的消息（邮件不符合规则的条件）：

```
fields @timestamp, @message, @logStream, @log
| filter recipients_matched = '[]'
    and rule_index = 2
| sort @timestamp desc
| limit 10000
```

在 Amazon Simple Email Service 中管理列表和订阅

您可以在 Amazon SES 中管理自己的邮件和订阅列表以及电子邮件抑制列表。为了帮助您维护自己的发件人声誉，SES 提供了账户级别和配置集级别抑制，以防止您向无效收件人发送电子邮件并损害您的发件人声誉。作为另一项针对退回邮件和投诉的措施，SES 可以通过订阅管理自动向所有发出的邮件添加取消订阅链接。

本章主题所列各节详细讨论了这些类型的列表中的每一种；但是，为了了解它们之间的差异以及全球抑制列表管理的关键变化，此处先对抑制列表进行概述。建议您在处理本章讨论的任何列表之前阅读此概述。

抑制列表和抑制覆盖机制概述

全球抑制列表删除功能不再面向客户，您也无法再与其交互来管理抑制设置。全局抑制列表运行并由 SES 在后台管理。作为客户，您现在可以使用账户级别的抑制列表和配置集级别的抑制覆盖，它们可让您更自定义地控制如何处理自己账户的电子邮件抑制操作。

下文将介绍不同类型的抑制清单、范围以及它们提供的优势。

- 全局抑制列表 – 由 SES 拥有和管理，以保护 SES 共享 IP 池中地址的声誉。
- 账户级抑制列表 - 由客户拥有和管理，以保护其账户声誉 - 覆盖全局抑制列表。
  - 配置集级别抑制 – 一种覆盖机制，通过使用配置集中指定的覆盖设置，对账户级抑制列表进行条件性或精细的控制。

全局抑制名单在新的 Amazon SES 控制台和 API v2 中引入账户级别和配置集级别抑制之前，是唯一的抑制列表类型。全球抑制清单由 SES 拥有和管理，以保护 SES 的声誉。这是必要的，因为所有 SES 客户都共享相同的 IP 地址池（除非他们有专用的 IP 地址 IPs），因此 SES 必须确保客户不会发送垃圾邮件或任何会对 SES 共享 IP 池中这些 IP 地址的声誉产生负面影响的内容。尽管您不再直接与全球抑制列表互动，但它仍然在后台运行，而且还可以应用全球抑制列表如何运作的一般原则来解释其他类型的抑制如何运作的总体原则。请参阅Amazon SES 全局黑名单。

> **Note**
>
> 全局黑名单删除请求表不再位于 Amazon SES 新控制台中，因为账户级别黑名单已经为本节说明的所有优点所取代。

账户级别的抑制列表的推出是为了让客户可以创建和控制自己的抑制列表和声誉，因此，账户级别的抑制列表仅适用于您的账户。新控制台中的账户级别抑制名单界面为管理账户级别抑制名单中的地址提供了一种简单的方法，包括添加或删除地址的批量操作。如果一个地址在全局抑制列表中，但不在您的账户级别抑制列表中（这意味着您想发送给它），但您确实发送了，Amazon SES 仍将尝试发送，但如果它反弹，反弹将影响您自己的声誉，但没有其他人将得到反弹，因为他们不能发送到该电子邮件地址，如果他们不使用自己的账户级别抑制列表；因此，账户级别的抑制列表仅覆盖您的账户的全局抑制列表。请参阅使用 Amazon SES 账户级黑名单。

配置集级别的抑制本身并非一个列表，而是一种机制，它允许您通过为不同电子邮件发送场景专门创建的配置集，对帐户级别的抑制列表进行自定义配置和覆盖。例如，如果同时为要添加的退回邮件地址和投诉地址配置账户级别黑名单，但您在配置集中定义了特定的电子邮件人口统计，您只对所添加的投诉地址感兴趣 – 您可以通过启用此配置集的抑制覆盖来实现这一目标，以便仅针对使用此配置集发送的电子邮件中的投诉（而不是在账户级别黑名单中设置的退回邮件和投诉）将电子邮件地址添加到您的账户级别黑名单中。使用配置集级别的抑制时，可以在不同级别覆盖您的账户级别抑制，包括根本不使用任何抑制。请参阅使用配置集级别的抑制来覆盖账户级别的黑名单。

Amazon SES 全局黑名单

Amazon SES 维护内部全局黑名单，该名单由 SES 在后台运行和管理。如有 SES 客户发送导致“查无此人的邮件”的电子邮件，则 SES 会将产生“查无此人的邮件”的电子邮件地址添加到全局黑名单。顾名思义，全局黑名单是全局性的，适用于所有 SES 客户。换言之，如果其他客户尝试将电子邮件发送到全局黑名单中的地址，则 SES 会接收该邮件，但不会将其发送出去，因为该电子邮件地址已被禁止。

全球抑制列表电子邮件地址删除请求功能不再面向客户，您无法再与其互动来管理抑制设置。为了取代这一功能，Amazon SES 现在为您提供了一种管理抑制设置的新方法，通过提供账户级别抑制列表和配置集级别抑制覆盖，您可以更自定义地控制如何处理自己账户的电子邮件抑制设置。有关更多信息，请参阅使用 Amazon SES 账户级黑名单 和使用配置集级别的抑制来覆盖账户级别的黑名单。

> **Important**
>
> 全局黑名单电子邮件地址删除请求表格不在 Amazon SES 控制台中，因为账户级别黑名单已取代它。要了解如何使用账户级别黑名单，请参阅 使用 Amazon SES 账户级黑名单。

全局黑名单注意事项

有关全局黑名单的关键因素：

- 全局黑名单由 SES 在后台运行并管理 - 您不能直接与其交互；但是，您可以使用自己的账户级别黑名单覆盖它。
- 默认情况下，已为所有 SES 账户启用全局黑名单。您不能禁用它。
- 由于 SES 将全局黑名单应用于所有客户，因此，您无法查询全局黑名单或手动向其中添加地址。
- 当某个电子邮件地址产生“查无此人的邮件”时，SES 会在短时间内将该地址添加到全局黑名单。经过这段时间之后，SES 会从黑名单中删除该地址。如果该地址产生另一个“查无此人的邮件”，那么 SES 会在更长的时间内将其重新添加到全局黑名单，并在该周期结束时将其删除。每当某个地址产生“查无此人的邮件”，该地址保留在全局黑名单上的时间都会增加。地址可在全局黑名单上保留最多 14 天。
- 如果您尝试将邮件发送到全局黑名单中的地址，则 SES 会接受该邮件，但不会将其发送出去。SES 会生成退回邮件通知，其中 `bounceType` 值为 `Permanent`，`bounceSubType` 值为 `Suppressed`。接收这种类型的退回邮件通知是知道地址是否在全局黑名单中的唯一方法。您无法查询全局黑名单。
- SES 将统计您发送到全局黑名单上地址的消息，包括账户的跳出率和每日发送配额。
- 对于任何造成“查无此人的邮件”的电子邮件地址，您应从邮件发送列表中删除导致黑名单退回邮件的地址，除非您完全确信该地址有效。
- 黑名单退回邮件会计入您账户的退回邮件率。如果邮件退回率过高，我们会对您的账户进行审核，或暂停您的账户发送电子邮件的功能。

> **Note**
>
> 了解 SES 抑制列表如何相互关联及其层次结构很重要，请参阅抑制列表和抑制机制覆盖概述。

使用 Amazon SES 账户级黑名单

我们引入了 Amazon SES 账户级抑制列表，使客户可以创建和控制自己的账户级抑制列表和并管理自己的声誉，因此您的账户级抑制列表只适用于您的账户。SES 控制台中的账户级黑名单界面为管理账户级黑名单中的地址提供了一种简单的方法，包括添加或删除地址的批量操作。

您的 SES 账户级黑名单适用于您在当前 AWS 区域中的 AWS 账户 。您可以使用 Amazon SES API v2 或控制台在账户级黑名单中逐个或批量添加或删除地址。

> **Note**
>
> 要批量添加或删除地址，您必须具有生产访问权限。如需了解有关沙盒的详情，请参阅请求生产访问权限（从 Amazon SES 沙盒中移出）。

Amazon SES 账户级黑名单注意事项

使用账户级黑名单时，您应该注意以下事项：

- 如果您在 2019 年 11 月 25 日之后开始使用 Amazon SES，则默认情况下，您的账户将使用账户级黑名单来处理退信和投诉。如果您在此日期之前开始使用 SES，则必须使用 SES API 中的 `PutAccountSuppressionAttributes` 操作来启用此功能。
- 如果您试图将邮件发送到账户级黑名单上的某个地址，而该地址的禁用原因与为您的账户级禁用设置选择的禁用原因匹配，则 SES 会接受该邮件，但不会发送该邮件；但如果它们不匹配，SES 将发送该邮件。为了帮助澄清这一点，提供了以下示例：
  - 您已设置账户级禁用设置，禁用原因为仅限退信，SES 不会尝试为您的账户级黑名单中禁用原因为退信的地址传递邮件。但是，SES 将尝试配送您的账户级别禁止列表中的地址，其原因为投诉（因为在这种情况下，它们不匹配）。
  - 您已设置账户级禁用设置，禁用原因为退信和投诉，SES 将不会尝试为您的账户级黑名单中禁用原因为退信或投诉的地址传递邮件。
- SES 不会将您发送到账户级抑制列表中的地址的邮件计入信誉。 BounceRate或声誉。 ComplaintRate您的账户的 AWS/SES 命名空间中的指标。此类消息会计入 AWS/SES 命名空间中的退回或投诉指标下。
- 如果某个地址在全局黑名单列表中，但不在您的账户级黑名单列表中（这意味着您要向它发送邮件），而您确实向该地址发送了邮件，SES 仍将尝试送达；但是，如果邮件退回，该邮件仍计入您账户的退回邮件率和每日发送限额。
- SES 将您发送到账户级黑名单地址的邮件计入您的每日发送配额。
- 账户级黑名单中的电子邮件地址会保留在那里，直到您将其删除。
- 如果您账户的电子邮件发送功能已暂停，SES 会在 90 天后自动删除账户级黑名单中的地址。如果您账户的电子邮件发送功能在此 90 天的期限结束之前恢复，则不会删除名单中的地址。
- Gmail 不会向 SES 提供投诉数据。如果收件人使用 Gmail Web 客户端中的 Spam（垃圾邮件）按钮将您发送的电子邮件举报为垃圾邮件，则这些邮件地址不会被添加到账户级黑名单中。

- 如果您的账户位于 SES 沙盒中，则可以启用账户级黑名单。但是，在将您的账户从沙箱中移除之前，您无法使用PutSuppressedDestination或CreateImportJob操作。如需了解有关沙盒的详情，请参阅请求生产访问权限（从 Amazon SES 沙盒中移出）。
- 仅将硬退信添加到账户级黑名单中。要详细了解软退信与硬退信之间的区别，请参阅 the section called “在 Amazon SES 发送电子邮件之后”。
- 在使用账户级黑名单时，SES 还会将导致硬退信的邮件地址添加到全局黑名单。

启用 Amazon SES 账户级黑名单

您可以使用 Amazon SES API v2 中的PutAccountSuppressionAttributes操作来启用和设置您的账户级别禁止列表。您可以使用 AWS CLI轻松快速地配置此设置。有关安装和配置 AWS CLI的更多信息，请参阅 AWS Command Line Interface 用户指南。

要配置账户级别的禁止名单，请使用 AWS CLI

- 在命令行输入以下命令：

  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
  ```

  要启用账户级黑名单，您必须为 `suppressed-reasons` 参数指定至少一个原因。您可以指定 `BOUNCE` 或 `COMPLAINT`，也可以同时指定两者，如上例所示。

要使用 Amazon SES 控制台配置账户级黑名单，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的 Configuration（配置）下方，选择 Suppression list（抑制列表）。
3. 在 Account-level settings（账户级别设置）窗格中，选择 Edit（编辑）。

4. 在黑名单中，勾选已启用方框。
5. 在抑制原因中，选择应将收件人电子邮件地址自动添加到帐户级黑名单的原因之一。
6. 选择保存更改。

为配置集启用 Amazon SES 账户级黑名单

您还可以对 Amazon SES 账户级黑名单进行配置，使其仅适用于指定的配置集。执行此操作后，仅当您在发送导致退信或投诉事件的电子邮件时指定了配置集时，才会将地址添加到黑名单。

> **Note**
>
> 以下过程假定您已安装 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

要为配置集配置配置配置账户级别的禁止列表，请使用 AWS CLI

- 在命令行输入以下命令：

  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
  ```

  在前面的示例中，*configSet*替换为应使用您的账户级别禁止列表的配置集的名称。

要使用 Amazon SES 控制台为配置集配置账户级黑名单，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。

2. 在导航窗格中的配置下，选择配置集。
3. 在配置集中，选择要使用自定义隐藏配置的配置集的名称。
4. 在黑名单选项窗格中，选择编辑。
5. 这些区域有：黑名单部分提供了定义自定义抑制的决策集，首先是使用此配置集来覆盖您的账户级别抑制的选项。configuration set-level suppression logic map（配置集级别抑制逻辑映射）将帮助您了解覆盖组合的影响。这些多层次的覆盖选择可以组合起来实现三种不同级别的抑制：

   a. 使用账户级别抑制：不要覆盖您的账户级别抑制，也不要实施任何配置集级别的抑制 - 基本上，使用此配置集发送的任何电子邮件都将使用您的账户级别抑制。要实现此目的，应按照以下步骤进行：

      - 在 Suppression list settings（抑制列表设置）中，取消选中 Override account level settings（覆盖账户级别设置）的复选框。

   b. 请勿使用任何抑制：在不启用任何配置集级别抑制的情况下覆盖您的账户级别抑制 - 这意味着使用此配置集发送的任何电子邮件都不会使用任何账户级别的抑制；换句话说，所有抑制都将被取消。要实现此目的，应按照以下步骤进行：

      i. 在抑制列表设置中，勾选覆盖账户级别设置复选框。

      ii. 在抑制列表设置中，取消勾选 Enabled (已启用)复选框。

   c. 使用配置集级别抑制：使用此配置集中定义的自定义黑名单设置覆盖您的账户级别抑制 - 这意味着使用此配置集发送的任何电子邮件将仅使用自己的隐藏设置并忽略任何账户级别的抑制设置。要实现此目的，应按照以下步骤进行：

      i. 在黑名单设置中，勾选覆盖账户级别设置复选框。

      ii. 在黑名单中，勾选已启用。

      iii. 在指定原因…中，选择要使用此配置集的抑制原因之一。

6. 选择保存更改。

将单个电子邮件地址添加到 Amazon SES 账户级黑名单

您可以使用 SES API v2 中的PutSuppressedDestination操作将各个地址添加到您的 Amazon SES 账户级别禁止列表中。您可以添加到账户级黑名单中的地址数量没有限制。

> **Note**
>
> 以下过程假定您已安装 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

使用 AWS CLI将单个地址添加到账户级黑名单

- 在命令行输入以下命令：

  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
  ```

  在前面的示例中，*recipient@example.com*替换为要添加到账户级禁止列表的电子邮件地址，以及*BOUNCE*将该地址添加到禁止列表的原因（可接受的值为BOUNCE和COMPLAINT）。

要使用 Amazon SES 控制台将单个地址添加到账户级黑名单，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的 Configuration（配置）下方，选择 Suppression list（抑制列表）。
3. 在黑名单窗格中，选择添加电子邮件地址。
4. 在 Email address（电子邮件地址）字段中输入电子邮件地址，然后在 Suppression reason（抑制原因）中选择原因 - 如果需要输入更多地址，请选择 Enter another address（输入其他地址），每输入一个地址重复一次。
5. 输入地址后，请检查输入是否准确。如果您发现了不应被提交的输入项，请选择它的 Remove（移除）按钮。

6. 请选择 Save changes（保存更改），以便将输入的电子邮件地址添加到您的账户级黑名单。

将电子邮件地址批量添加到 Amazon SES 账户级黑名单

您可以先将联系人列表上传到 Amazon S3 对象，然后使用 Amazon SES API v2 中的CreateImportJob操作来批量添加地址。

> **Note**
>
> - 您可以添加到账户级黑名单的地址数量没有限制，但每次 API 调用的 Amazon S3 对象中的批量添加数量限制为 10 万个地址。
> - 如果您的数据来源是 S3 桶，则该桶必须与您要导入的桶位于同一个区域中。

要将电子邮件地址批量添加到账户级黑名单，请完成以下步骤。

- 以 CSV 或 JSON 格式将您的地址列表上载到 Amazon S3 对象中。

  添加地址的 CSV 格式示例：

  recipient1@example.com,*BOUNCE*

  recipient2@example.com,*COMPLAINT*

  仅支持换行符分隔的 JSON 文件。在此格式中，每一行都是一个完整的 JSON 对象，其中包含单独的地址定义。

  用于添加地址的 JSON 格式示例：

  {"emailAddress":"*recipient1@example.com*","reason":"*BOUNCE*"}

  {"emailAddress":"*recipient2@example.com*","reason":"*COMPLAINT*"}

  在前面的示例中，将*recipient1@example.com*和*recipient2@example.com*替换为要添加到账户级别禁止列表中的电子邮件地址。您将地址添加到黑名单的可接受原因是 *BOUNCE* 和 *COMPLAINT*。
- 向 SES 授予对 Amazon S3 对象的读取权限。

以下策略应用于 Amazon S3 存储桶时，会向 SES 授予对该存储桶的读取权限。有关将策略附加到 Amazon S3 的存储桶的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用存储桶策略和用户策略。

JSON

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

- 为 SES 提供使用您的 AWS KMS 密钥的权限。

  如果 Amazon S3 对象使用 AWS KMS 密钥加密，则需要向 Amazon SES 授予使用该 AWS KMS 密钥的权限。SES 只能从客户托管式密钥获得权限，而不是原定设置的 KMS 密钥。您需要向 SES 提供客户托管式密钥的使用权限，方法是在密钥策略中添加一条语句。

  将以下策略语句粘贴到密钥策略中，以允许 SES 使用您的客户托管式密钥。

```
{
   "Sid": "AllowSESToDecrypt",
   "Effect": "Allow",
   "Principal": {
       "Service":"ses.amazonaws.com"
   },
   "Action": [
       "kms:Decrypt",
```

```
    ],
    "Resource": "*"
}
```

- 使用 SES API v2 中的CreateImportJob操作。

> **Note**
>
> 以下示例假定您已安装 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

在命令行输入以下命令。*s3bucket*替换为 Amazon S3 存储桶的*s3object*名称和 Amazon S3 对象的名称。

```
aws sesv2 create-import-job --import-destination
 SuppressionListDestination={SuppressionListImportAction=PUT} --import-data-source
 S3Url=s3://s3bucket/s3object,DataFormat=CSV
```

要使用 Amazon SES 控制台将电子邮件地址批量添加到您的账户级黑名单中：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的 Configuration（配置）下方，选择 Suppression list（抑制列表）。
3. 在黑名单表格中，展开批量操作按钮，并选择批量添加电子邮件地址。
4. 在批量操作规范中，选择 (a) 从 S3 桶中选择文件 或 (b) 从文件中导入—每个导入方法都给出了过程：

    a. 从 S3 存储桶中选择文件-如果您的源文件已存储在 Amazon S3 存储桶中：

        i. 如果您知道要使用的 Amazon S3 存储桶的 URI，请在 Amazon S3 URI 字段中输入；否则，选择浏览 S3：

            A. 在存储桶中，选择 S3 存储桶的名称。

            B. 在对象中，选择该文件的名称，然后选择选择-您将返回至批量操作规格。

            C. （可选）如果您想转到 Amazon S3 控制台以查看 S3 对象的详细信息，请选择查看。

        ii. 在文件格式中，选择您选择从 Amazon S3 存储桶导入的文件的格式。

iii. 选择添加电子邮件地址，开始从文件中导入地址——将显示批量操作选项卡下的一个表。

b. 从文件导入-如果您有本地源文件要上传到新的或现有的 Amazon S3 存储桶：

i. 在导入源文件，选择选择文件。

ii. 在文件浏览器中选择 JSON 或 CSV 文件，然后选择打开-你会看到文件的名称、大小和日期显示在选择文件按钮下方。

iii. 扩展 Amazon S3 存储桶，然后选择 S3 存储桶。

- 要将文件上载到新存储桶，请选择 Create S3 bucket（创建 S3 存储桶），在 Bucket name（存储桶）字段中输入名称，然后选择 Create bucket（创建存储桶）。

iv. 选择添加电子邮件地址，开始从文件中导入地址——将显示批量操作选项卡下的一个表。

5. 无论使用哪种导入方法，作业 ID 都将在批量操作以及导入类型、状态和日期列出-要查看作业详细信息，请选择作业 ID。

6. 选择黑名单选项卡并显示所有成功导入的电子邮件地址并添加了它们的抑制原因和日期 - 可以使用以下选项：

a. 选择电子邮件地址，或者选中相应的复选框，然后选择查看报告以查看其详细信息。（如果由于退回或投诉而自动添加到您的黑名单中的地址，则将显示有关导致添加反馈事件的信息，包括产生触发事件的电子邮件的详细信息。）

b. 选中要从帐户黑名单中删除的一个或多个电子邮件地址的相应复选框，然后选择移除。

查看 Amazon SES 账户级黑名单中的地址列表

您可以使用 SES API v2 中的ListSuppressedDestinations操作查看账户级别禁止列表中的所有电子邮件地址列表。

> **Note**
>
> 以下过程假定您已安装 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

查看账户级黑名单中的所有电子邮件地址的列表

- 在命令行输入以下命令：

```
aws sesv2 list-suppressed-destinations
```

前一个命令返回您账户的账户级黑名单中的所有电子邮件地址。输出与以下内容类似：

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

- 注意 — 如果您的输出包含带有字符串值的 NextToken “” 字段，则表示您的账户的禁止列表中还有其他电子邮件地址。要查看其他黑名单中的地址，请向 ListSuppressedDestinations 发出另一个请求，并在 --next-token 参数中传递返回的字符串值，如下所示：

  ```
  aws sesv2 list-suppressed-destinations --next-token string
  ```

  在前面的命令中，*string*用返回的 NextToken 值替换。

  有关更多信息，请参阅 How to list over 1000 email addresses from account-level suppression list。

可以使用 StartDate 选项来只显示在特定日期以后 添加到列表中的电子邮件地址。

查看在特定日期以后添加到账户级黑名单中的地址列表

- 在命令行输入以下命令：

```
aws sesv2 list-suppressed-destinations --start-date 1604394130
```

在前面的命令中，*1604394130*替换为开始日期的 Unix 时间戳。

您还可以使用 EndDate 选项来只显示在特定日期之前 添加到列表中的电子邮件地址。

查看在特定日期之前添加到账户级黑名单中的地址列表

- 在命令行输入以下命令：

```
aws sesv2 list-suppressed-destinations --end-date 1611126000
```

在前面的命令中，*1611126000*替换为结束日期的 Unix 时间戳。

在 Linux、macOS 或 Unix 命令行中，您也可以使用内置的 grep 实用程序来搜索特定地址或域。

在账户级黑名单中搜索特定地址

- 在命令行输入以下命令：

```
aws sesv2 list-suppressed-destinations | grep -A2 'example.com'
```

在前面的命令中，*example.com*替换为要搜索的文本字符串（例如地址或域名）。

要使用 Amazon SES 控制台查看账户级黑名单中的所有电子邮件地址的列表，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的 Configuration（配置）下方，选择 Suppression list（抑制列表）。
3. 在黑名单窗格中，将显示账户级黑名单中的所有电子邮件地址，并添加了其抑制原因和日期 - 可以使用以下选项：

   a. 选择电子邮件地址，或者选中相应的复选框，然后选择查看报告以查看其详细信息。（如果由于退回或投诉而自动添加到您的禁止列表中的地址，则将显示有关导致添加反馈事件的信息，包括产生触发事件的电子邮件的详细信息。）

b. 您可以通过选择齿轮图标来自定义隐藏列表表格-将显示模式，您可以在其中自定义页面大小、换行和要查看的列-进行选择后，选择确认。隐藏列表将反映您的查看选择项。

从 Amazon SES 账户级黑名单中删除单个电子邮件地址

如果某个地址在您账户的禁止列表中，但您知道该地址不应该出现在列表中，则可以使用 SES API v2 中的DeleteSuppressedDestination操作将其删除。

> **Note**
>
> 以下过程假定您已安装 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

使用 AWS CLI从账户级黑名单中删除单个地址

- 在命令行输入以下命令：

  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
  ```

  在前面的示例中，*recipient@example.com*替换为要从账户级别的禁止列表中删除的电子邮件地址。

要使用 Amazon SES 控制台从账户级黑名单中删除单个地址，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下方，选择黑名单。
3. 通过 (a) 表选择或 (b) 输入条目，来移除单个电子邮件地址：

a. 从表中选择：在 Suppression list（黑名单）表中，选中一个或多个电子邮件地址的相应复选框，然后选择 Remove（移除）。

b. 在字段中输入：

   i. 在黑名单窗格中，选择添加电子邮件地址。

   ii. 在 Email address（电子邮件地址）字段中输入电子邮件地址 - 如果需要输入更多地址，请选择 Enter another address（输入其他地址），每输入一个地址重复一次。

   iii. 输入地址后，请检查输入是否准确。如果您发现了不应被提交的输入项，请选择它的 Remove（移除）按钮。

   iv. 选择 Save changes（保存更改），以便将输入的电子邮件地址从您的账户级黑名单当中移除。

从 Amazon SES 账户级黑名单中批量删除电子邮件地址

您可以先将联系人列表上传到 Amazon S3 对象中，然后使用 SES API v2 中的CreateImportJob操作来批量删除地址。

> **Note**
>
> - 您可以从账户级黑名单中删除的地址数量没有限制，但每次 API 调用的 Amazon S3 对象中的批量删除数量限制为 10000 个地址。
> - 如果您的数据来源是 S3 桶，则该桶必须与您要导入的桶位于同一个区域中。

要从账户级黑名单中批量删除电子邮件地址，请完成以下步骤。

- 以 CSV 或 JSON 格式将您的地址列表上传到 Amazon S3 对象中。

  用于删除地址的 CSV 格式示例：

  ```
  recipient3@example.com
  ```

  仅支持换行符分隔的 JSON 文件。在此格式中，每一行都是一个完整的 JSON 对象，其中包含单独的地址定义。

  用于添加地址的 JSON 格式示例：

```
{"emailAddress":"recipient3@example.com"}
```

在前面的示例中，*recipient3@example.com*替换为要从账户级别禁止列表中删除的电子邮件地址。

- 向 SES 授予对 Amazon S3 对象的读取权限。

  以下策略应用于 Amazon S3 存储桶时，会向 SES 授予对该存储桶的读取权限。有关将策略附加到 Amazon S3 的存储桶的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用存储桶策略和用户策略。

  JSON

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

- 授予 SES 使用您的 AWS KMS 密钥的权限。

  如果 Amazon S3 对象使用 AWS KMS 密钥加密，则需要向 Amazon SES 授予使用该 AWS KMS 密钥的权限。SES 只能从客户托管式密钥获得权限，而不是原定设置的 KMS 密钥。您需要向 SES 提供客户托管式密钥的使用权限，方法是在密钥策略中添加一条语句。

  将以下策略语句粘贴到密钥策略中，以允许 SES 使用您的客户托管式密钥。

```
{
```

```
    "Sid": "AllowSESToDecrypt",
    "Effect": "Allow",
    "Principal": {
        "Service":"ses.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
    ],
    "Resource": "*"
}
```

- 使用 SES API v2 中的CreateImportJob操作。

> **Note**
>
> 以下示例假定您已安装 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

在命令行输入以下命令。*s3bucket*替换为 Amazon S3 存储桶的*s3object*名称和 Amazon S3 对象的名称。

```
aws sesv2 create-import-job --import-destination
 SuppressionListDestination={SuppressionListImportAction=DELETE} --import-data-source
 S3Url="s3://s3bucket/s3object",DataFormat=CSV
```

要使用 Amazon SES 控制台从账户级黑名单中批量删除电子邮件地址，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的 Configuration（配置）下方，选择 Suppression list（抑制列表）。
3. 在黑名单表中，展开批量操作按钮然后选择批量删除电子邮件地址。
4. 在批量操作规范中，选择 (a) 从 S3 桶中选择文件 或 (b) 从文件中导入 - 每个导入方法都给出了过程：

    a. 从 S3 存储桶中选择文件-如果您的源文件已存储在 Amazon S3 存储桶中：

        i. 如果您知道要使用的 Amazon S3 存储桶的 URI，请在 Amazon S3 URI 字段中输入；否则，选择浏览 S3：

A. 在存储桶中，选择 S3 存储桶的名称。

B. 在对象中，选择该文件的名称，然后选择选择-您将返回至批量操作规格。

C. （可选）如果您想转到 Amazon S3 控制台以查看 S3 对象的详细信息，请选择查看。

ii. 在文件格式中，选择要从 Amazon S3 存储桶中导入的文件的格式。

iii. 选择删除电子邮件地址，从文件中开始地址的导入——将显示批量操作选项卡下的一个表。

b. 从文件导入-如果您有本地源文件要上传到新的或现有的 Amazon S3 存储桶：

i. 在导入源文件，选择选择文件。

ii. 在文件浏览器中选择 JSON 或 CSV 文件，然后选择打开-你会看到文件的名称、大小和日期显示在选择文件按钮下方。

iii. 扩展 Amazon S3 存储桶，然后选择 S3 存储桶。

- 要将文件上载到新存储桶，请选择 Create S3 bucket（创建 S3 存储桶），在 Bucket name（存储桶）字段中输入名称，然后选择 Create bucket（创建存储桶）。

iv. 选择删除电子邮件地址，从文件中开始地址的导入——将显示批量操作选项卡下的一个表。

5. 无论使用哪种导入方法，作业 ID 都将在批量操作以及导入类型、状态和日期列出-要查看作业详细信息，请选择作业 ID。

6. 选择黑名单选项卡，将不再显示从黑名单中删除的所有成功导入的电子邮件地址。

查看账户的导入任务的列表

您可以使用 Amazon SES API v2 中的ListImportJobs操作，查看您的账户的账户级别禁止列表中的所有电子邮件地址列表。

> **Note**
>
> 以下过程假定您已安装 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

查看账户的所有导入任务的列表

- 在命令行输入以下命令：

```
aws sesv2 list-import-jobs
```

前面的命令返回账户的所有导入任务。输出与以下内容类似：

```
{
    "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"
        }
    ]
```

```
}
```

要使用 Amazon SES 控制台来查看账户的所有导入任务的列表，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的 Configuration（配置）下方，选择 Suppression list（抑制列表）。
3. 在禁止名单窗格中，选择批量操作选项卡。
4. 所有导入作业都将列在批量操作表以及导入类型、状态和日期。
5. 要查看作业详细信息，请选择作业 ID，然后显示以下窗格：

    a. 批量操作状态：显示作业的总体状态、完成的时间和日期、导入的记录数以及任何未能成功导入的记录的计数。

    b. 批量操作详情：显示作业 ID、是用于添加还是删除地址、文件格式是 JSON 还是 CSV、存储批量文件的 Amazon S3 存储桶的 URI 以及批量操作的创建时间和日期。

获取有关账户的导入任务的信息

您可以使用 Amazon SES API v2 中的GetImportJob操作来获取有关账户导入任务的信息。

> **Note**
>
> 以下过程假定您已安装 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

获取有关账户的导入任务的信息

- 在命令行输入以下命令：

```
aws sesv2 get-import-job --job-id JobId
```

前面的命令返回有关账户的导入任务的信息。输出与以下内容类似：

```
{
    "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"
}
```

要使用 Amazon SES 控制台获取有关账户导入任务的信息，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的 Configuration（配置）下方，选择 Suppression list（抑制列表）。
3. 在禁止名单窗格中，选择批量操作选项卡。
4. 所有导入作业都将列在批量操作表以及导入类型、状态和日期。
5. 要查看作业详细信息，请选择作业 ID，然后显示以下窗格：

    a. 批量操作状态：显示作业的总体状态、完成的时间和日期、导入的记录数以及任何未能成功导入的记录的计数。

    b. 批量操作详情：显示作业 ID、是用于添加还是删除地址、文件格式是 JSON 还是 CSV、存储批量文件的 Amazon S3 存储桶的 URI 以及批量操作的创建时间和日期。

禁用 Amazon SES 账户级黑名单

您可以使用 SES API v2 中的PutAccountSuppressionAttributes操作，通过从属性中删除值来有效地禁用账户级别的禁止列表。`suppressed-reasons`

> **Note**
>
> 以下过程假定您已安装 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

要禁用账户级别的禁止名单，请使用 AWS CLI

- 在命令行输入以下命令：

```
aws sesv2 put-account-suppression-attributes --suppressed-reasons
```

要使用 Amazon SES 控制台禁用账户级黑名单，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的 Configuration（配置）下方，选择 Suppression list（抑制列表）。
3. 在 Account-level settings（账户级别设置）窗格中，选择 Edit（编辑）。
4. 在黑名单设置中，取消勾选已启用复选框。
5. 选择保存更改。

使用配置集级别的抑制来覆盖账户级别的黑名单

在为整个账户设置账户级别的黑名单时，您可以通过使用配置集级别的抑制覆盖该名单，来为其他配置集单独自定义该名单。利用此更精细的粒度，您可以对分配给他们的配置集的不同电子邮件发送组使用自定义抑制设置。例如，假设同时为要添加的退回邮件地址和投诉地址配置账户级别黑名单，但您在配置集中定义了特定的电子邮件人口统计，您只对所添加的投诉地址感兴趣 – 您可以通过启用此配置集的抑制覆盖来实现这一目标，以便仅针对使用此配置集发送的电子邮件中的投诉（而不是在账户级别黑名单中设置的退回邮件和投诉）将电子邮件地址添加到您的账户级别黑名单中。

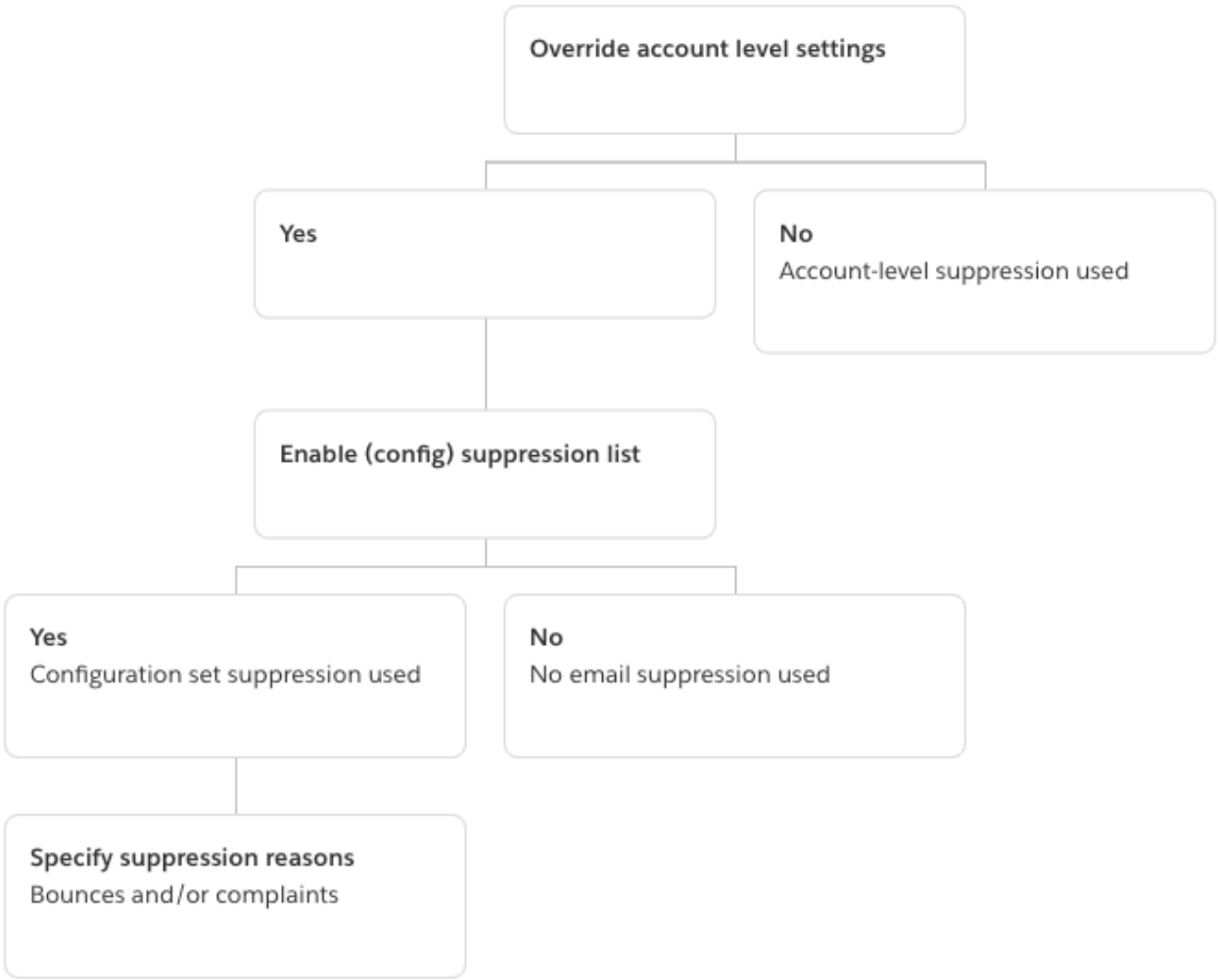
使用配置集级别的抑制时，可以在不同级别覆盖您的账户级别抑制，包括根本不使用任何抑制。为了帮助了解可在以下控制台过程中设置的这些不同级别的抑制，下面的关系映射对有关启用或禁用各种覆盖级别的决策集进行了建模，这些选择取决于它们的组合，可用于实施三个不同程度的抑制。

- 无覆盖（原定设置）– 配置集使用账户级别的黑名单设置。

- 覆盖账户级别设置 – 这将否定任何账户级别的黑名单设置；使用此配置集发送的电子邮件根本不会使用任何抑制设置。
- 覆盖账户级别设置，并启用配置集级别的抑制 – 使用此配置集发送的电子邮件将仅使用您为其启用的抑制条件（退回邮件和/或投诉）– 无论您的账户级别黑名单设置是什么，它都会覆盖这些设置。

> **Note**
>
> 对于任何未在配置集级别指定的抑制条件，由于账户级别的设置已被覆盖，因此抑制行为将回退到全局禁止列表。

Configuration set-level suppression logic



请记住，配置集级别抑制不是实际黑名单，它只是一种用配置集中定义的自定义抑制设置覆盖账户级别黑名单的机制 – 这意味着，使用此配置集发送的任何电子邮件将仅使用自己的抑制设置，并且将忽略任何账户级别的抑制设置。换句话说，只需更改（覆盖）决定哪些电子邮件地址将添加到账户级别黑名单中的抑制理由，配置集级别的抑制就会与您的账户级别黑名单进行交互。

启用配置集级别抑制

要使用 Amazon SES 新控制台启用配置集级别的抑制功能，请执行一下操作：

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中的配置下，选择配置集。
3. 在配置集中，选择要使用自定义隐藏配置的配置集的名称。
4. 在黑名单选项窗格中，选择编辑。
5. 这些区域有：黑名单部分提供了定义自定义抑制的决策集，首先是使用此配置集来覆盖您的账户级别抑制的选项。configuration set-level suppression logic map（配置集级别抑制逻辑映射）将帮助您了解覆盖组合的影响。这些多层次的覆盖选择可以组合起来实现三种不同级别的抑制：

   a. 使用账户级别抑制：不要覆盖您的账户级别抑制，也不要实施任何配置集级别的抑制 - 基本上，使用此配置集发送的任何电子邮件都将使用您的账户级别抑制。要实现此目的，应按照以下步骤进行：

      - 在 Suppression list settings（抑制列表设置）中，取消选中 Override account level settings（覆盖账户级别设置）的复选框。

   b. 请勿使用任何抑制：在不启用任何配置集级别抑制的情况下覆盖您的账户级别抑制 - 这意味着使用此配置集发送的任何电子邮件都不会使用任何账户级别的抑制；换句话说，所有抑制都将被取消。要实现此目的，应按照以下步骤进行：

      i. 在抑制列表设置中，勾选覆盖账户级别设置复选框。

      ii. 在抑制列表设置中，取消勾选 Enabled (已启用)复选框。

   c. 使用配置集级别抑制：使用此配置集中定义的自定义抑制设置覆盖您的账户级别黑名单 - 这意味着，使用此配置集发送的任何电子邮件将仅使用自己的抑制设置，忽略任何账户级别的抑制设置。要实现此目的，应按照以下步骤进行：

      i. 在黑名单设置中，勾选覆盖账户级别设置复选框。

      ii. 在黑名单中，勾选已启用。

iii. 在指定原因...中，选择要使用此配置集的抑制原因之一。

6. 选择 Save changes（保存更改）。

使用列表管理

Amazon SES 提供列表管理功能，这意味着客户可以管理自己的邮件列表，称为联系人列表。联系人列表让您能够存储订阅了一个或多个特定主题的所有联系人。联系人是接收您的电子邮件的最终用户。主题是列表中的兴趣组、主题或标签。列表可以包含多个主题。

通过使用 Amazon SES API v2 中的 `ListContacts` 操作，您可以检索订阅了特定主题的所有联系人的列表，而您可以使用 `SendEmail` 操作向这些联系人发送电子邮件。

有关订阅管理的更多信息，请参阅使用订阅管理。

列表管理概述

在使用列表管理时，您应注意以下事项：

- 您可以在创建列表时指定列表主题。
- 每人只允许有一份联系人名单 AWS 账户。
- 一个列表可以包含最多 20 个主题。
- 您可以更新现有联系人列表，包括向列表添加新主题、在列表中添加或删除联系人，以及更新列表或主题的联系人首选项。
- 您可以更新主题元数据，例如主题显示名称或描述。
- 您可以获取联系人列表中的联系人、订阅某个主题的联系人、取消订阅某个主题的联系人以及取消订阅列表中所有主题的联系人的列表。
- 您可以使用 `CreateImportJob` API 将您的现有联系人列表导入到 Amazon SES。
- 如果有电子邮件发送到您的联系人列表上已取消订阅的联系人，那么 Amazon SES 将退回该邮件。有关更多信息，请参阅 使用订阅管理。
- 每个联系人都可以有关联的属性，您可以使用这些属性来存储有关该联系人的信息。

配置列表管理

您可以使用以下操作来配置列表管理功能。有关联系人列表和联系人操作的完整列表，请参阅 Amazon SES API v2 参考。

创建联系人列表

您可以使用 Amazon SES API v2 中的 CreateContactList 操作来创建联系人列表。您可以使用 AWS CLI轻松快速地配置此设置。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

要创建联系人列表，请使用 AWS CLI

- 在命令行输入以下命令：

```
aws sesv2 create-contact-list --cli-input-json file://CONTACT-LIST-JSON
```

在前面的命令中，*CONTACT-LIST-JSON*替换为CreateContactList请求的 JSON 文件的路径。

请求的 CreateContactList 输入 JSON 文件的示例如下：

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

创建联系人

您可以使用 Amazon SES API v2 中的 `CreateContact` 操作来创建联系人。您可以使用 AWS CLI轻松快速地配置此设置。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

要创建联系人，请使用 AWS CLI

- 在命令行输入以下命令：

```
aws sesv2 create-contact --cli-input-json file://CONTACT-JSON
```

在前面的命令中，*CONTACT-JSON*替换为`CreateContact`请求的 JSON 文件的路径。

请求的 `CreateContact` 输入 JSON 文件的示例如下：

```
{
    "ContactListName": "ExampleContactListName",
    "EmailAddress": "example@amazon.com",
    "UnsubscribeAll": false,
    "TopicPreferences": [
        {
            "TopicName": "Sports",
            "SubscriptionStatus": "OPT_IN"
        }
    ],
    "AttributesData": "{\"Name\": \"John\", \"Location\": \"Seattle\"}"
}
```

在以上示例中，UnsubscribeAll 值为 false 表明联系人未取消订阅所有主题，而值为 true 则意味着联系人已取消订阅所有主题。

TopicPreferences 包含有关联系人的主题订阅状态的信息。在前面的示例中，联系人选择订阅了“Sports”主题，那么他将收到“Sports”主题的所有电子邮件。

AttributesData 是一个 JSON 字段，您可以在其中放置有关我们的联系人的任何元数据。它必须是有效的 JSON 对象。

将联系人批量导入您的联系人列表

您可以手动批量添加地址，方法是首先将您的联系人上载到 Amazon S3 对象中，然后使用 Amazon SES API v2 中的 CreateImportJob 操作，或者使用 SES 控制台。有关更多信息，请参阅将电子邮件地址批量添加到账户级黑名单。

在导入您的联系人之前，您应该创建一个联系人列表。

> **Note**
>
> 每个联系人列表中最多可以添加 100 万个联系人 ImportJob。

要将联系人批量添加到您的联系人列表，请完成以下步骤。

- 以 CSV 或 JSON 格式将您的联系人上传到 Amazon S3 对象中。

  CSV 格式

  上传到 Amazon S3 的文件的第一行应该是标题行。

  使用 CSV 格式需要展平 topicPreferences 对象。topicPreferences 中的每个主题都将有单独的标题字段。

  使用 CSV 格式将联系人批量添加到联系人列表的示例：

```
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 格式

仅支持换行符分隔的 JSON 文件。在此格式中，每一行都是一个完整的 JSON 对象，其中包含一个联系人的信息。

将联系人批量添加到联系人列表的 JSON 格式示例：

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

在前面的示例中，将*example1@amazon.com*和*example2@amazon.com*替换为要添加到联系人列表的电子邮件地址。将 attributesData 值替换为特定于联系人的值。此外，请将*Sports*和*Cycling*topicName，替换为适用于您的联系人的内容。可接受topicPreferences的是*OPT_IN*和*OPT_OUT*。

以 CSV 或 JSON 格式将您的联系人上传到 Amazon S3 对象中时，支持以下属性：

属性	描述
emailAddress	联系人的电子邮件地址。此字段为必填字段。
unsubscribeAll	一个布尔值状态，表示联系人是否已取消订阅所有联系人列表主题。
topicPreferences	联系人选择加入或选择退出主题的首选项。
attributesData	附加到联系人的属性数据。

- 向 Amazon SES 授予对 Amazon S3 对象的读取权限。

  以下策略应用于 Amazon S3 存储桶时，会向 Amazon SES 授予对该存储桶的读取权限。有关将策略附加到 Amazon S3 的存储桶的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用存储桶策略和用户策略。

  JSON

```
{
    "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"
                }
```

```
            }
        }
    ]
}
```

- 授予 Amazon SES 使用您的 AWS KMS 密钥的权限。

  如果 Amazon S3 对象使用 AWS KMS 密钥加密，则需要向 Amazon SES 授予使用 KMS 密钥的权限。Amazon SES 只能从客户托管式密钥获得权限，而不是原定设置的 KMS 密钥。您必须向 Amazon SES 提供客户托管式密钥的使用权限，方法是在密钥策略中添加一条语句。

  将以下策略语句粘贴到密钥策略中，以允许 Amazon SES 使用您的客户托管式密钥。

```
{
   "Sid": "AllowSESToDecrypt",
   "Effect": "Allow",
   "Principal": {
       "Service":"ses.amazonaws.com"
   },
   "Action": [
       "kms:Decrypt",
   ],
   "Resource": "*"
}
```

- 使用 Amazon SES API v2 中的 `CreateImportJob` 操作。

> **Note**
>
> 以下示例假定您已安装 AWS CLI。有关安装和配置的更多信息 AWS CLI，请参阅《AWS Command Line Interface 用户指南》。

在命令行输入以下命令。替*s3bucket*换为 Amazon S3 存储桶的*s3object*名称和 Amazon S3 对象名称的名称。

```
aws sesv2 create-import-job --import-destination
 ContactListDestination={ContactListName=ExampleContactListName,ContactListImportAction=PUT}
 --import-data-source S3Url="s3://s3bucket/s3object",DataFormat=CSV
```

列出管理演练及示例

以下演练提供了一些示例，说明如何使用列表管理来列出联系人，如何使用 ListManagementOptions 在电子邮件中指定联系人列表和主题名称，以及如何插入取消订阅链接。

1. 使用列出联系人 AWS CLI — 您可以使用该ListContacts操作来检索已订阅特定主题的所有联系人的列表，以及允许您向他们发送电子邮件的SendEmail操作。

   在命令行输入以下命令：

   ```
   aws sesv2 list-contacts --cli-input-json file://LIST-CONTACTS-JSON
   ```

   在前面的命令中，*LIST-CONTACTS-JSON*替换为ListContacts请求的 JSON 文件的路径。

   请求的 ListContacts 输入 JSON 文件的示例如下：

   ```
   {
       "ContactListName": "ExampleContactListName",
       "Filter": {
           "FilteredStatus": "OPT_IN",
           "TopicFilter": {
               "TopicName": "Cycling",
               "UseDefaultIfPreferenceUnavailable": true
           }
       },
       "PageSize": 50
   }
   ```

   FilteredStatus 显示您要筛选的订阅状态，即 OPT_IN 或 OPT_OUT。

   TopicFilter 是一个可选的筛选器，它指定您希望获得其结果的主题，在以上示例中为“Cycling”。

   UseDefaultIfPreferenceUnavailable 可以具有值 true 或 false。如果为 true，当联系人没有对主题的任何明确首选项时，将使用主题默认首选项。如果为 false，则只考虑筛选具有明确设置的首选项的联系人。

2. 在启用 **ListManagementOptions** 的情况下发送邮件–使用上述 ListContacts 操作在列表中列出联系人后，您可以使用 SendEmail 操作，通过 ListManagementOptions 标头指定联系人列表和主题名，以向每个联系人发送电子邮件。

要将 ListManagementOptions 与 SendEmail 操作一起使用，请包括电子邮件所属的 contactListName 和 topicName（可选）：

```
ListManagementOptions:
    String contactListName
    String topicName
```

如果您在发送给不在您联系人列表中的收件人电子邮件地址的 SendEmail 请求中包含 ListManagementOptions，那么您的列表中会自动创建联系人。

如果将电子邮件发送给您的联系人列表中已取消订阅的联系人，那么 Amazon SES 将退回该电子邮件，这意味着您不需要更新您的 SendEmail 请求即可避免向已取消订阅的联系人发送邮件。

3. 指明取消订阅链接的位置 – 利用 ListManagementOptions 时，您可以通过使用 {{amazonSESUnsubscribeUrl}} 占位符指定 Amazon SES 需要插入取消订阅 URL 的位置，选择让 SES 在电子邮件中添加取消订阅脚注链接。仅支持 HTML 和 TEXT 内容类型的占位符替换。您可以最多两次包含该占位符。如果使用了两次以上，那么仅替换前两次出现的占位符。有关更多信息，请参阅 使用订阅管理。

   或者，如果使用 SMTP 接口发送电子邮件，则可以使用 X-SES-LIST-MANAGEMENT-OPTIONS 标头指定列表和主题名称。

   要指定在使用 SMTP 接口发送电子邮件时的列表和主题名称，请将以下电子邮件标头添加到您的邮件：

   ```
   X-SES-LIST-MANAGEMENT-OPTIONS: {contactListName}; topic={topicName}
   ```

使用订阅管理

Amazon SES 提供订阅管理功能，当您在 SendEmail 操作请求中的 ListManagementOptions 中指定 contactListName 和 topicName 时，Amazon SES 通过该功能自动启用每一封外发电子邮件中的取消订阅链接。

如果联系人取消订阅特定主题或列表，那么 Amazon SES 在未来不允许向该联系人发送该主题或列表的电子邮件。

> **Note**
>
> - Amazon SES 订阅管理支持许多电子邮件服务提供商强制执行的批量发件人要求，有关更多信息，请参阅批量发件人更改概述中的第 2 节。
> - 订阅管理可用于使用 Amazon SES 中的 Easy DKIM 的发件人，但是对于在调用 Amazon SES 之前自行为电子邮件签名的发件人，Amazon SES 无法为其添加您的电子邮件的取消订阅链接。

有关列表管理以及如何使用它的信息（包括检索订阅了特定主题的所有联系人的列表），请参阅使用列表管理。

订阅管理概览

在使用订阅管理时，您应注意以下事项：

- 订阅管理将由 Amazon SES 完全管理。这意味着 Amazon SES 会从取消订阅网页接收取消订阅电子邮件和请求，然后更新您的列表中的联系人的首选项。您可以使用配置集通知来接收取消订阅通知。有关配置集的更多信息，请参阅在 SES 中使用配置集。
- 您需要在发送电子邮件时指定联系人列表。通过 `List-Unsubscribe` 标头和 `ListManagementOptions` 脚注链接进行的订阅管理将得到相应的处理。
- Amazon SES 添加了对 `List-Unsubscribe` 标头标准的支持，将使电子邮件客户端和收件箱提供商能够在电子邮件的顶部显示取消订阅链接（如果支持）- 并非所有电子邮件服务提供商都支持这些标头。
- `List-Unsubscribe` 标头遵循以下行为：
  - 如果联系人单击同时指定了联系人列表和主题的电子邮件中的取消订阅链接，那么该联系人将仅取消订阅该特定主题。
  - 如果未指定主题，那么该联系人将取消订阅列表中的所有主题。
- 当联系人单击电子邮件脚注中的取消订阅链接时，联系人将转到取消订阅登录页面。
- 取消订阅登录页面将让联系人选择是否更新其对特定列表中的所有主题的首选项，也就是 `OPT_IN` 或 `OPT_OUT`。登录页面还提供取消订阅列表中的所有主题的选项。
- 如果使用 `ListManagementOptions`，则必须在您的电子邮件中包含 `{{amazonSESUnsubscribeUrl}}` 占位符，用于指明 Amazon SES 需要在哪里插入取消订阅 URL。您可以最多两次包含该占位符。如果使用了两次以上，那么仅替换前两次出现的占位符。

- 只有将电子邮件发送给单个收件人时，才会添加 List-Unsubscribe 标头和 ListManagementOptions 脚注链接。
- 对于您不希望联系人能够取消订阅的事务性电子邮件，您可以在 SendEmail 请求中省略 ListManagementOptions 字段。

取消订阅标头注意事项

当电子邮件包含以下标头时，将启用通过取消订阅链接进行的订阅管理：

List-Unsubscribe

List-Unsubscribe-Post

当您使用 Amazon SES 的订阅管理 (ListManagementOptions) 时，如果电子邮件中存在这些标头，Amazon SES 将覆盖这些标头。

通过单击由这些标头生成的链接来取消订阅的收件人将获得不同的体验，具体取决于联系人的电子邮件客户端或收件箱提供商，因为一些提供商无法识别 List-Unsubscribe 和 List-Unsubscribe-Post 标头；使用此类提供商发送给收件人的电子邮件将不会显示“Unsubscribe”（取消订阅）链接。

其电子邮件客户端可识别这些标头的收件人将看到“Unsubscribe”（取消订阅）链接，并可以通过该链接取消订阅，但无法选择取消订阅哪些主题，只能取消订阅该电子邮件所属的主题。

有关 List-Unsubscribe 标头的更多信息，请参阅 RFC 2369，有关 List-Unsubscribe-Post 标头，请参阅 RFC 8058。

> **Note**
>
> 根据许多电子邮件服务提供商强制执行的批量发件人要求，Amazon SES 支持一键式取消订阅，有关更多信息，请参阅在 Using one-click unsubscribe with Amazon SES。

添加取消订阅脚注链接

您需要在模板化和非模板化电子邮件中使用 {{amazonSESUnsubscribeUrl}} 占位符，用于指定 Amazon SES 需要在哪里插入取消订阅 URL。

仅支持 HTML 和 TEXT 内容类型的占位符替换。

您可以最多两次包含该占位符。如果使用了两次以上，那么仅替换前两次出现的占位符。

> **Note**
>
> 只有在使用SendEmail操作时指定ListManagementOptions为标题或在使用 SMTP 接口时将 X-SES-LIST-MANAGEMENT-OPTIONS 指定为标题时，才能使用占{{amazonSESUnsubscribeUrl}}位符。（不要混淆 List-Unsubscribe 或 List-Unsubscribe-Post 标头，它们不依赖于 ListManagementOptions 并且可以单独使用。）

监控您的 Amazon SES 发送活动

Amazon SES 提供了使用事件、指标和统计信息监控发送活动的几种方法。事件是与您指定作为指标跟踪的发送活动相关的事情。指标代表一个按时间顺序排列的数据点集，它们代表生成统计信息的受监控事件类型的值。统计信息是指定时间段（包括到现在）的指标数据聚合。

这些监控方法可以帮助您跟踪重要的指标，如您账户的退信率、投诉率和拒绝率。过高的退信率和投诉率可能会影响您使用 SES 发送电子邮件的能力。此外，还可以使用这些方法来衡量客户与您发送的电子邮件的互动率，方法是帮助您利用事件发布和与配置集相关的自定义域来确定整体打开率和点击率 - 请参阅 配置自定义域以处理打开和单击跟踪。

设置监控的第一步是确定要使用 SES 衡量和监控的与发送活动相关的电子邮件事件类型。您可以在 SES 中选择以下要监控的事件类型：

- Send（发送）– 发送请求成功，Amazon SES 将尝试将邮件发送到收件人的邮件服务器。（如果使用账户级别或全局抑制，SES 仍会将其计为发送，但会抑制送达。）
- RenderingFailure— 由于模板渲染问题，电子邮件未发送。当模板数据丢失或模板参数与数据不匹配时，可能会发生此事件类型。（此事件类型仅在您使用 `SendTemplatedEmail` 或 `SendBulkTemplatedEmail` API 操作发送电子邮件时发生。）
- Reject（拒绝）– Amazon SES 已接受电子邮件，但确定它包含病毒，而未尝试将其发送到收件人的邮件服务器。
- Delivery（送达）– Amazon SES 成功将电子邮件发送到收件人的邮件服务器。
- 退信 – 收件人的邮件服务器永久拒绝了电子邮件的硬退信。（仅当 S ES 不再尝试发送电子邮件时，才会包含软退件。 通常，这些软退件表示投递失败，尽管在某些情况下，即使邮件成功到达收件人收件箱，也可以返回软退件。 当收件人发送 out-of-office自动回复时，通常会发生这种情况。 在这篇 re AWS : Post 文章中了解有关软反弹的更多信息。）
- Complaint（投诉）– 电子邮件已成功送达收件人的邮件服务器，但收件人将其标记为垃圾邮件。
- DeliveryDelay— 由于出现临时问题，无法将电子邮件发送到收件人的邮件服务器。例如，当收件人的收件箱已满，或者当接收电子邮件服务器遇到临时问题时，可能会发生传送延迟。
- Subscription（订阅）– 电子邮件已成功发送，但收件人通过单击电子邮件标头中的 `List-Unsubscribe` 或脚注中的 `Unsubscribe` 链接更新了订阅首选项。
- Open（打开）– 收件人已收到邮件并在其电子邮件客户端中打开了邮件。
- Click（单击）– 收件人单击了电子邮件中包含的一个或多个链接。

可通过多种方法监控电子邮件发送事件。选择的方法取决于要监控的事件的类型、要通过其监控的事件的粒度和详细程度，以及希望 SES 发布数据的位置。需要使用反馈通知或事件发布跟踪退信和投诉事件。还可选择使用多种监控方法。每种方法的特征如下表中所列。

监控方法	可监控的事件	如何访问数据	详细程度	粒度
SES 控制台	账户运行状况、发送的电子邮件、已使用的配额、成功发送请求、拒绝、退回和投诉（当前声誉的最近历史记录）	SES 控制台中的账户控制面板页面	计数和百分比	在整个 AWS 账户中
SES 控制台	账户运行状况、发送的电子邮件、退回和投诉（当前的声誉）	SES 控制台中的声誉指标页面	仅计算的比率	在整个 AWS 账户中
Virtual Deliverability Manager	账户统计信息、ISP、发送身份、配置集、已发送、已送达、投诉、暂时和永久退回、打开和单击、送达率和声誉。	SES 控制台中的 the section called “控制面板”。 SES 控制台中的 the section called “Advisor”。	计数和百分比	在整个 AWS 账户中
SES API	送达、退信、投诉和拒绝	`GetSendStatistics` API 操作	仅计数	在整个 AWS 账户中
亚马逊 CloudWatch 控制台	发送、发送、打开、点击、退回、退回率、投	CloudWatch 控制台	仅计数	在整个 AWS 账户中

监控方法	可监控的事件	如何访问数据	详细程度	粒度
	诉率、投诉率、拒绝、渲染失败以及列入黑名单。 IPs	Note 有些指标要等到关联事件发生 CloudWatch 后才会显示在中。例如，退回指标要等 CloudWatch 到您发送了至少一封退回邮件，或者直到您使用邮箱模拟器生成模拟退回事件后才会显示在中。		
反馈通知	送达、退信和投诉	Amazon SNS 通知（送达、退信和投诉）或电子邮件（仅退信和投诉）。请参阅设置事件通知。	每个事件的详细信息	在整个 AWS 账户中

监控方法	可监控的事件	如何访问数据	详细程度	粒度
事件发布	发送、送达、打开、点击、退信、投诉、拒绝和呈现失败。	亚马逊 CloudWatch 或亚马逊 Data Firehose，或者通过亚马逊 SNS 通知，请参阅。使用事件发布监控电子邮件发送 （需支付额外费用，请参阅每个指标的价格 CloudWatch。）	每个事件的详细信息	精细（基于用户可定义的电子邮件特征）
利用与配置集关联的自定义域发布事件 - 更多信息	打开率和点击率跟踪。	亚马逊 CloudWatch 或亚马逊 Data Firehose，或者通过亚马逊 SNS 通知。 （需支付额外费用，请参阅每个指标的价格 CloudWatch。）	每个事件的详细信息。	精细（基于用户可定义的电子邮件特征）

> **Note**
>
> 按电子邮件发送事件衡量的指标可能与您的发送配额不完全符合。这种差异可能是由电子邮件退回邮件和拒绝，或者使用 SES 收件箱模拟器导致的。要了解您有多接近发送配额，请参阅监控您的发送配额。

有关如何使用每种监控方法的信息，请参阅以下主题：

使用 Amazon SES 控制台监控您的发送统计信息

从 Amazon SES 控制台的账户控制面板、声誉指标和 SMTP 设置页面，您可以监控所有电子邮件发送、使用情况、统计数据、SMTP 设置、整体账户运行状况以及声誉指标。以下各节介绍在其中每个控制面板页面上提供的指标和统计数据。

应该指出的是，虽然the section called “账户控制面板”和the section called “声誉指标”控制台页面都包含退回邮件和投诉指标，但这两组退回率和投诉率之间存在细微差异，解释如下：

- 账户控制面板页面 – 根据选定的日期范围，您可以查看过去的退回邮件率和投诉率，其中显示了截至目前的指标变化进展情况。
- 声誉指标页面 — 跳出率和投诉率基于从高水平计算您的总体历史平均值时收到的最新数据点（不应将其与您的常规bounce/complaint rate, which corresponds to precise bounce/complaint事件混淆，因为这些事件是实时发生的，如账户控制面板页面所示）。

作为比较 Reputation metrics（声誉指标）页面和 Account dashboard（账户控制面板）页面之间的退回邮件率或投诉率的简单示例，假设昨天的比率是 2%，现在是 1%，在 Reputation metrics（声誉指标）页面上，您只能看到当前 1% 的比率，而在 Account dashboard（账户控制面板）页面上，以图表形式绘制进展情况，显示昨天的比率是 2%，今天的比率是 1%。

账户控制面板

您可以直接从 SES 控制台 Account dashboard（账户控制面板）页面的 Daily email usage（每日电子邮件使用情况）窗格中监控从您的账户发送的电子邮件数量，以及已使用的发送限额百分比。可以在 Sending Statistics（发送统计数据）窗格中监控您账户的送达率和拒绝率，也可以在以下窗格中监控与您的电子邮件发送相关的其他关键因素：

- 发送限制 – 包含以下通过 SES 发送邮件的适用配额：
  - Daily sending quota（每日发送限额）– 您在 24 小时内可发送的最大电子邮件数量。
  - Maximum send rate（最大发送速率）– 每秒可从账户发送的最大电子邮件数量。
- 账户运行状况 – 您的 SES 账户的状态：
  - `Healthy` – 目前没有影响您的账户的声誉相关问题。

  - `Under review` – 已确定与 SES 账户相关的潜在问题 – 您的账户正在接受审查，而您需要努力纠正问题。
  - `Paused` – 由于从您的账户发送的电子邮件存在问题，因此，当前已暂停您账户的电子邮件发送功能。在纠正问题后，您可以请求恢复账户的电子邮件发送功能。
- 每日电子邮件使用情况 – 检查每日使用情况以确保您没有达到发送限制：
  - Emails sent（已发送电子邮件数）– 在 24 小时内发送的电子邮件总数。
  - Remaining sends（剩余发送数）-在 24 小时内可以发送的剩余电子邮件总数。
  - Sending quota used（已使用发送限额）– 已使用的每日发送配额的百分比。
- 发送统计信息 – 由图表组成，这些图表显示按时间排序的一组数据点中四个基本指标的进展情况，这些数据点表示受监控事件类型的值，使用 1 小时的聚合周期生成选定日期范围内的统计信息。您可以选择起始值从 `Last 1 day` 到 `Last 14 days` 的数据范围以筛选以下图表：
  - Sends（发送)– 所选日期范围内成功的电子邮件发送请求数的总和。
  - Rejects（拒绝）– SES 在所选日期范围内拒绝的发送请求的平均比率，计算方式为 `Rejects/Sends * 100`。
  - Bounces（退回邮件）– 从总体历史发件人声誉指标得出的平均比率，显示所选日期范围的进展情况。
  - Complaints（投诉）– 从总体历史发件人声誉指标得出的平均比率，显示所选日期范围的进展情况。

这些图表中的每一个都包含一个 “查看范围 CloudWatch” 按钮，该按钮将在 Amazon CloudWatch 控制台中打开相应的指标，允许查看详细数据、执行自定义指标运算以及创建警报 CloudWatch。

声誉指标

除了退回邮件率和投诉率之外，Reputation metrics（声誉指标）页面还可让您深入了解影响您声誉的关键因素，包括以下窗格：

- 摘要 – 概述了您的声誉状况。
  - Status（状态）– 基于历史退回邮件率和投诉率的整体声誉状况：
    - `Healthy` – 两个指标都处于正常水平。
    - `Under review` – 一个或两个指标已自动促使您的账户被置于审查状态。
    - `At risk` – 一个或两个指标已达到不健康水平，您账户的电子邮件发送功能可能存在风险。
  - 已发送的电子邮件（过去 24 小时）— 过去 24 小时内发送的电子邮件总数。
  - 剩余的发送次数 — 24 小时内可供发送的剩余电子邮件总数。

    - 已使用的发送限额 — 已使用的每日发送限额的百分比。
- 账户级别选项卡内容：
    - 退回邮件率
        - Status（状态）– 使用与 Summary（摘要）窗格中描述的相同值来指示退回邮件率的状况。
        - Historic bounce rate（历史退回邮件率）– 您账户中导致硬退信的邮件的电子邮件百分比，从基于表示典型发送实践的典型量的总体历史平均值计算得出。
    - 投诉率
        - Status（状态）– 使用与 Summary（摘要）窗格中描述的相同值来指示投诉率的状况。
        - Historic bounce rate（历史退回邮件率）– 从您账户发送的导致收件人将其报告为垃圾邮件的电子邮件百分比，从基于表示典型发送实践的典型量的总体历史平均值计算得出。
- Configuration set（配置集）选项卡内容：
    - 按配置集显示的声誉
        - Configuration set（配置集）– 可让您键入或选择启用了声誉指标的配置集，以便基于使用所选配置集发送的电子邮件查看摘要、退回邮件和投诉数据。选择配置集后显示的结果窗格与上面的声誉指标页面所述的相同，只不过它们仅基于使用所选配置集发送的电子邮件，而不是您的总体账户级别发送指标。

SMTP 设置

本页列出了通过 SES API 或以编程方式使用 Amazon SES SMTP 接口所需的 SMTP 设置，并提供了用于创建和管理 SMTP 凭证的链接：

- SMTP 设置 – 如果您想使用支持 SMTP 的编程语言、电子邮件服务器或应用程序连接到 Amazon SES SMTP 接口，需提供以下信息：
    - SMTP 终端节点
    - STARTTLS 端口
    - 传输层安全性协议（TLS）
    - TLS 包装器端口
    - 为创建和管理 SMTP 和 IAM 凭证提供的身份验证链接

使用控制台监控发送和信誉指标

以下过程将帮助您开始探索发送和声誉指标，您可以使用 Account dashboard（账户控制面板）页面获取基于最近历史记录的指标（最多 14 天），也可以使用 Reputation metrics（声誉指标）页面获取基于当前整体历史记录的指标。

查看已发送的电子邮件和已使用的发送配额

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在导航窗格中，选择 Account dashboard（账户控制面板）。您的使用情况统计信息显示在 Daily email usage（每日电子邮件使用情况）下。

查看发送次数、拒绝率、退回邮件率和投诉率

1. 在导航窗格中，选择 Account dashboard（账户控制面板）。
2. 在 Sending statistics（发送统计信息）部分中，使用 Date range（日期范围）下拉菜单选择日期范围的起始值，以筛选 Sending statistics（发送统计信息）部分正下方的四个图表。
3. 根据选定的日期范围，您可以查看过去的计数和比率，其中显示了截至目前的指标变化进展情况。
4. 在任何图表中，选择 “查看范围 CloudWatch” 按钮，在 Amazon CloudWatch 控制台中打开相应的指标，您可以在其中查看详细数据、执行自定义指标数学运算以及创建监控警报 CloudWatch。

查看总体历史退回邮件率和投诉率

1. 在导航窗格中，选择 Reputation metrics（声誉指标）。
2. 在 Bounce rate（退回邮件率）窗格中，您可以查看从账户发送的导致查无此人的邮件的电子邮件的百分比，在 Complaint rate（投诉率）窗格中，您可以查看从账户发送的导致收件人将其报告为垃圾邮件的电子邮件的百分比；这两个指标都是从基于典型发送实践的典型电子邮件量计算得出的。
3. 在任一窗格中，选择 “查看范围 CloudWatch” 按钮，在 Amazon CloudWatch 控制台中打开相应的指标，您可以在其中查看详细数据、执行自定义指标数学运算以及创建监控警报。 CloudWatch

按配置集查看声誉指标

1. 在导航窗格中，选择 Reputation metrics（声誉指标）。
2. 在声誉指标页面上，选择 Configuration set（配置集）选项卡。

3. 在 Reputation by configuration set（按配置集显示的声誉）窗格中，在 Configuration set（配置集）字段中单击，然后开始键入或选择启用了声誉指标的配置集。
4. 选择配置集后，它将加载 Summary（摘要）、Bounce（退回邮件）和 Complaint（投诉）窗格，其中显示仅基于使用选定配置集发送的电子邮件的指标。

使用 Amazon SES API 监控您的使用情况统计数据

Amazon SES API 提供 `GetSendStatistics` 操作，用于返回有关服务使用情况的信息。我们建议您定期查看您的发送统计数据，以便根据需要进行调整。

当您调用 `GetSendStatistics` 操作时，您会收到表示您最近两周的发送活动的数据点的列表。此列表中的每个数据点代表一项为时 15 分钟的活动，并包含该时间段的以下信息：

- 查无此人的邮件的数量
- 投诉的数量
- 传送尝试的次数 (对应于您已发送的电子邮件数)
- 被拒绝的发送尝试的次数
- 分析时段的时间戳

有关 `GetSendStatistics` 操作的完整描述，请参阅 Amazon Simple Email Service API 参考。

在本节中，您将找到以下主题：

- the section called “使用调用 GetSendStatistics API 操作 AWS CLI”
- the section called “以编程方式调用 GetSendStatistics 操作”

使用调用 **GetSendStatistics** API 操作 AWS CLI

调用 `GetSendStatistics` API 操作的最简单方法是使用 AWS Command Line Interface (AWS CLI)。

要调用 **`GetSendStatistics`** API 操作，请使用 AWS CLI

1. 如果您尚未这样做，请安装 AWS CLI。有关更多信息，请参阅《AWS Command Line Interface 用户指南 AWS Command Line Interface》中的 “安装”。

2. 如果您尚未执行此操作，请将配置 AWS CLI 为使用您的 AWS 证书。有关更多信息，请参阅《AWS Command Line Interface 用户指南 AWS CLI》中的 “配置”。
3. 在命令行处，运行以下命令：

```
aws ses get-send-statistics
```

如果配置 AWS CLI 正确，您将看到以 JSON 格式发送统计信息的列表。每个 JSON 对象都包含 15 分钟时段内的聚合发送统计数据。

以编程方式调用 GetSendStatistics 操作

您也可以使用调用该GetSendStatistics操作 AWS SDKs。本节包括 for Go、PHP、Python 和 Ruby 的代码示例。 AWS SDKs 选择以下链接之一可查看相应语言的代码示例：

- 适用于 Go 的 AWS SDK的代码示例
- 适用于 PHP 的 AWS SDK的代码示例
- AWS SDK for Python (Boto)的代码示例
- 适用于 Ruby 的 AWS SDK的代码示例

> **Note**
>
> 这些代码示例假设您已创建 AWS 共享凭证文件，其中包含您的 AWS 访问密钥 ID、私有访问 AWS 密钥和首选 AWS 区域。有关更多信息，请参阅共享凭证和配置文件。

GetSendStatistics使用拨打电话 适用于 Go 的 AWS SDK

```
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使用拨打电话 适用于 PHP 的 AWS SDK

```
<?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使用拨打电话 AWS SDK for Python (Boto)

```
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`使用拨打电话 适用于 Ruby 的 AWS SDK

```
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
```

使用 Amazon SES 事件发布监控电子邮件发送

为了使您能够精细地跟踪电子邮件的发送，您可以将 Amazon SES 设置为根据您定义的特征向亚马逊、Amazon Data Firehose CloudWatch、Amazon Pinpoint、亚马逊简单通知服务或 EventBridge 亚马逊发布电子邮件发送事件。

您可以跟踪多种类型的电子邮件发送事件，包括发送、送达、打开、点击、退回、投诉、拒绝、呈现失败和送达延迟。此信息可用于操作和分析目的。例如，您可以将电子邮件发送数据发布到 CloudWatch 并创建控制面板来跟踪电子邮件活动的效果，也可以使用 Amazon SNS 在某些事件发生时向您发送通知。

事件发布如何与配置集和消息标签协同工作

要使用事件发布，您首先要设置一个或多个配置集。配置集用于指定发布事件的位置和要发布的事件。然后，每次发送电子邮件时，您都要提供配置集的名称和一个或多个邮件标签（采用名称/值对的形式）以对邮件进行分类。例如，如果您要宣传书籍，则当您为关联的营销活动发送电子邮件时，可以将邮件标签命名为 genre，并指定值 sci-fi 或 western。

根据您使用的电子邮件发送接口，您可以将消息标签作为参数提供给 SendEmail API 操作的 EmailTags 字段，或者将消息标签添加到 SES 特定的电子邮件头 X-SES-MESSAGE-TAGS。有关配置集的更多信息，请参阅在 SES 中使用配置集。

除了您指定的消息标签之外，SES 还会将自动标签添加到您发送的消息。您无需执行任何其他步骤就能使用自动标签。

下表列出了自动应用于您使用 SES 发送的消息的自动标签。

SES 自动标签

自动标签名称	描述
ses:caller-identity	发送电子邮件的 SES 用户的 IAM 身份。
ses:configuration-set	与电子邮件关联的配置集的名称。
ses:from-domain	“发件人”地址的域。
ses:outgoing-ip	SES 用来发送电子邮件的 IP 地址。
ses:source-ip	发起人用来发送电子邮件的 IP 地址。
ses:source-tls-version	调用方用来发送电子邮件的 TLS 协议版本。
ses:outgoing-tls-version	SES 用来发送电子邮件的 TLS 协议版本。

电子邮件营销活动的细粒度反馈

ses:feedback-id-<*a* or *b*> 标签是一个可选的消息标签，您可以将其视为混合或半自动标签，虽然它与上一节中讨论的自动标签类似，但不同之处在于，您必须手动添加它并使用 ses: 前缀键。您最多可以使用两个这样的标签，分别定义 ses:feedback-id-a 和 ses:feedback-id-b。

当您指定这些标签时，SES 会自动将它们附加到标准 Feedback-ID 标头中，这些标头用于在反馈循环（FBL）中提供传送统计信息，例如投诉率和垃圾邮件率，请参阅反馈循环。标Feedback-ID题由 SES 用于收集投诉信息的标识符 SESInternalID 和标识 SES 为发送平台的静态标签 Amazon SES 组成，例如：

```
FeedBackId:feedback-id-a:feedback-id-b:((SESInternalID):(AmazonSES))
```

这些可选的反馈 ID 标签为您提供了一种生成细粒度反馈的方式，例如，针对您作为电子邮件营销活动一部分发送的邮件。您可以通过在 SendEmail 操作请求的 EmailTags 字段中将其指定为消息标签来使用 ses:feedback-id-<*a* or *b*>，如以下示例所示：

```
{
  "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"
}
```

如果以原始格式发送，则可以将其 ses:feedback-id-<*a* or *b*> 作为消息标签添加到特定于 SES 的标头 X-SES-MESSAGE-TAGS 中。

也可以在亚马逊 CloudWatch 中跟踪`ses:feedback-id-<a or b>`消息标签，方法是将其指定为 CloudWatch 价值来源，就像任何其他消息标签一样，请参阅 the section called “添加 CloudWatch 活动目的地详情”（需支付额外费用，参见每个指标的价格 CloudWatch。）

如何使用事件发布

以下几节包含设置和使用 SES 事件发布所需的信息。

- 设置事件发布
- 使用事件数据

事件发布术语

以下列表定义了与 SES 事件发布相关的术语。

电子邮件发送事件

与您提交到 SES 的电子邮件的结果关联的信息。发送事件包括：

- Send（发送）– 发送请求成功，Amazon SES 将尝试将邮件发送到收件人的邮件服务器。（如果使用账户级别或全局抑制，SES 仍会将其计为发送，但会抑制送达。）
- RenderingFailure— 由于模板渲染问题，电子邮件未发送。当模板数据丢失或模板参数与数据不匹配时，可能会发生此事件类型。（此事件类型仅在您使用 `SendTemplatedEmail` 或 `SendBulkTemplatedEmail` API 操作发送电子邮件时发生。）
- Reject（拒绝）– Amazon SES 已接受电子邮件，但确定它包含病毒，而未尝试将其发送到收件人的邮件服务器。
- Delivery（送达）– Amazon SES 成功将电子邮件发送到收件人的邮件服务器。
- 退信 – 收件人的邮件服务器永久拒绝了电子邮件的硬退信。（仅当 S ES 不再尝试发送电子邮件时，才会包含软退件。 通常，这些软退件表示投递失败，尽管在某些情况下，即使邮件成功到达收件人收件箱，也可以返回软退件。 当收件人发送 out-of-office自动回复时，通常会发生这种情况。 在这篇 re AWS : Post 文章中了解有关软反弹的更多信息。）
- Complaint（投诉）– 电子邮件已成功送达收件人的邮件服务器，但收件人将其标记为垃圾邮件。
- DeliveryDelay— 由于出现临时问题，无法将电子邮件发送到收件人的邮件服务器。例如，当收件人的收件箱已满，或者当接收电子邮件服务器遇到临时问题时，可能会发生传送延迟。
- Subscription（订阅）– 电子邮件已成功发送，但收件人通过单击电子邮件标头中的 `List-Unsubscribe` 或脚注中的 `Unsubscribe` 链接更新了订阅首选项。

- Open（打开）– 收件人已收到邮件并在其电子邮件客户端中打开了邮件。
- Click（单击）– 收件人单击了电子邮件中包含的一个或多个链接。

配置集

一组规则，用于定义 SES 将电子邮件发送事件发布到的目标，以及要发布的电子邮件发送事件的类型。在发送您希望用于事件发布的电子邮件时，您将指定与电子邮件关联的配置集。

事件目标

您向其发布 SES 电子邮件发送事件的 AWS 服务。您设置的每个事件目标都属于一个（且仅属于一个）配置集。

邮件标签

一个名称/值，用于出于事件发布目的对电子邮件进行分类。示例包括营销活动/书籍 和营销活动/服装。在发送电子邮件时，您可以将邮件标签指定为 API 调用的参数，或指定为特定于 SES 的电子邮件标头。

自动标签

自动包含在事件发布报告中的邮件标签。有一个用于配置集名称、“发件人”地址的域、发起人的出站 IP 地址、SES 出站 IP 地址和发起人的 IAM 身份的自动标签。

设置 Amazon SES 事件发布

本节介绍配置 Amazon SES 以将您的电子邮件发送事件发布到以下 AWS 服务所需的操作：

- Amazon CloudWatch
- Amazon Data Firehose
- Amazon Pinpoint
- Amazon Simple Notification Service (Amazon SNS)

以下主题介绍了设置事件发布所需的以下步骤：

1. 必须使用 Amazon SES 控制台或 API 创建一个配置集。
2. 将一个或多个事件目标（Firehose CloudWatch、Pinpoint 或 SNS）添加到配置集，并配置事件目标的唯一参数。
3. 在发送电子邮件时，指定要使用的包含事件目标的配置集。

本节中的主题

步骤 1：创建配置集

必须首先具有配置集以设置事件发布。如果您还没有配置集，或者希望创建新配置集，请参阅 在 SES 中创建配置集

您也可以使用 Amazon SES API V2 或 Amazon SES CLI v2 中的CreateConfigurationSet操作创建配置集，请参阅。创建一个配置集 (AWS CLI)

步骤 2：添加事件目标

事件目标是将 Amazon SES 事件发布到的位置。您设置的每个事件目标都属于一个（且仅属于一个）配置集。使用 Amazon SES 设置活动目的地时，您可以选择 AWS 服务目的地，然后指定与该目的地关联的参数。

设置活动目的地时，可以选择将事件发送到以下 AWS 服务之一：

- Amazon CloudWatch
- Amazon Data Firehose
- Amazon EventBridge
- Amazon Pinpoint
- Amazon Simple Notification Service (Amazon SNS)

您选择的事件目标取决于您所需的事件详细级别，以及您所需的接收事件信息的方式。如果您只想了解每一类事件的运行总数（例如，为了可以设置在总数过高时发出警报），可以使用 CloudWatch。

如果您想将详细的事件记录输出到其他服务（例如亚马逊 OpenSearch 服务或亚马逊 Redshift）进行分析，则可以使用 Firehose。

如果您想要在某些事件发生时接收通知，可以使用 Amazon SNS。

本节包含以下主题

- 为 CloudWatch 活动发布设置活动目的地
- 为 Amazon SES 事件发布设置 Data Firehose 事件目标
- 为活动发布设置一个 Amazon EventBridge 目的地
- 针对事件发布设置 Amazon Pinpoint 事件目标
- 针对事件发布设置 Amazon SNS 事件目标

为 CloudWatch 活动发布设置活动目的地

借助亚马逊 CloudWatch 指标，您可以使用事件目标发布向发送事件的 Amazon SES 电子邮件 CloudWatch。由于只能在配置集中设置 CloudWatch 事件目的地，因此必须先创建配置集，然后将事件目标添加到配置集中。

将 CloudWatch 事件目标添加到配置集时，必须选择一个或多个与发送电子邮件时使用的消息标签相对应的 CloudWatch 维度。与消息标签一样， CloudWatch 维度是可以帮助您唯一标识指标的名称/值对。

例如，您可以有名为 `campaign` 的一个邮件标签和一个维度，用于标识您的电子邮件营销活动。当您将电子邮件发送事件发布到时 CloudWatch，选择消息标签和维度非常重要，因为这些选择会影响您的 CloudWatch 账单，并决定如何筛选电子邮件发送事件数据 CloudWatch。

本节提供的信息可帮助您选择维度，然后说明如何向配置集添加 CloudWatch 事件目的地。

本节中的主题

- 添加 CloudWatch 事件目标
- 选择 CloudWatch 维度

添加 CloudWatch 事件目标

本节中的过程说明如何将 CloudWatch 事件目标详细信息添加到配置集中，并假设您已经完成了中的步骤 1 到步骤 6 创建事件目标。

您还可以使用 Amazon SES API V2 中的UpdateConfigurationSetEventDestination操作来创建和修改事件目标。

使用控制台向配置集添加 CloudWatch 事件目标详细信息

1. 以下是在步骤 7 中选择 CloudWatch 活动目的地类型的详细说明，并假设您已完成中之前的所有步骤创建事件目标。选择 CloudWatch 目标类型、输入目标名称并启用事件发布后，将显示

Amazon CloudWatch 维度窗格，其字段将在以下步骤中进行处理。（需支付额外费用，请参阅每个指标的价格 CloudWatch。）

2. 对于值来源，指定 Amazon SES 将如何获取其传递到的数据 CloudWatch。提供了以下值来源：

   - 邮件标签 – Amazon SES 从使用 `X-SES-MESSAGE-TAGS` 标头或 `EmailTags` API 参数指定的标签中检索维度名称和值。有关使用邮件标签的更多信息，请参阅the section called “步骤 3：在发送时指定配置集”。

     > **Note**
     >
     > 邮件标签可以包含数字 0-9、字母 A-Z（大写和小写字母）、连字符（-）和下划线（_）。

     您也可以使用 Message Tag (邮件标签) 值来源根据 Amazon SES 自动标签创建维度。要使用自动标签，请键入自动标签的完整名称以作为 Dimension Name (维度名称)。例如，要根据配置集自动标签创建维度，请将 `ses:configuration-set` 作为 Dimension Name (维度名称) 并将配置集的名称作为 Default Value (默认值)。有关完整的自动标签列表，请参阅事件发布如何与配置集和消息标签协同工作。

   - Email Header (电子邮件标头) – Amazon SES 从电子邮件中的标头检索维度名称和值。

     > **Note**
     >
     > 您不能使用以下任意电子邮件标头作为 Dimension Name (维度名称)：`Received`、`To`、`From`、`DKIM-Signature`、`CC`、`message-id` 或 `Return-Path`。

   - Link Tag (链接标签) – Amazon SES 从您在链接中指定的标签检索维度名称和值。有关向链接添加标签的更多信息，请参阅我能否用唯一标识符来标记链接？。

3. 对于 Dimension Name (维度名称)，请键入要传递到 CloudWatch 的维度的名称。

   > **Note**
   >
   > 维度名称只能包含 ASCII 字母 (a-z , A-Z)、数字 (0 - 9)、下划线 (_) 和dashes (-)。不允许使用空格、重音字符、非拉丁字符和其他特殊字符。

4. 对于 Default Value（默认值），请键入维度的值。

> **Note**
>
> 维度值只能包含 ASCII 字母（a-z、A—Z）、数字 (0-9)、下划线 (_)、短划线 (-)、@ 符号和句点 (.)。不允许使用空格、重音字符、非拉丁字符和其他特殊字符。

5. 如果要添加更多维度，请选择 Add Dimension（添加维度）。否则，请选择 Next（下一步）。
6. 在审核屏幕上，如果您对定义事件目标的方式感到满意，请选择添加目标。

选择 CloudWatch 维度

选择用作 CloudWatch 维度的名称和值时，请考虑以下因素：

- 每个指标的价格 — 您可以免费查看基本的 Amazon SES 指标。 CloudWatch 但是，当您使用事件发布收集指标时，会产生CloudWatch 详细监控费用。事件类型、维度名称和维度值的每个唯一组合都会在中创建不同的指标 CloudWatch。使用 CloudWatch详细监控时，您需要为每个指标付费。出于此原因，您可能想要避免选择需要许多不同值的维度。例如，除非对通过“发件人”域跟踪电子邮件发送事件非常感兴趣，否则可能不希望为 Amazon SES 自动标签 `ses:from-domain` 定义维度，因为它可能需要许多不同的值。有关更多信息，请参阅CloudWatch 定价。
- 指标筛选-如果一个指标有多个维度，则无法 CloudWatch 根据每个维度分别访问该指标。因此，在为单个 CloudWatch 活动目的地添加多个维度之前，请仔细考虑。例如，如果要按 `campaign` 或 `campaign` 和 `genre` 组合查看指标，您需要添加两个事件目标：一个目标仅将 `campaign` 作为维度，另一个目标将 `campaign` 和 `genre` 作为维度。
- 维度值来源 – 除了使用特定于 Amazon SES 的标头或 API 的参数指定维度值之外，您还可以选择让 Amazon SES 从您自己的 MIME 消息标头中获取维度值。如果您已在使用自定义标头并且不想更改电子邮件或对电子邮件发送 API 的调用以根据标头值收集指标，则可以使用此选项。如果对 Amazon SES 事件发布使用自己的 MIME 邮件标头，则用于 Amazon SES 事件发布的标头名称和值只能包含字母 A-Z、数字 0-9、下划线（_）、at 符号（@）、连字符（-）和句点（.）。如果您指定的名称或值包含其他字符，则发送电子邮件的呼叫仍将成功，但不会将事件指标发送到 Amazon CloudWatch。

有关 CloudWatch 概念的更多信息，请参阅亚马逊 CloudWatch 用户指南中的亚马逊 CloudWatch 概念。

为 Amazon SES 事件发布设置 Data Firehose 事件目标

Amazon Data Firehose 事件目标代表将特定的 Amazon SES 电子邮件发送事件发布到 Firehose 的实体。由于 Firehose 事件目标只能在配置集中设置，因此首先必须创建配置集。接下来，将事件目标添加到配置集中。

本节中的过程演示如何将 Firehose 事件目标详细信息添加到配置集，并假设您已完成 创建事件目标 中的步骤 1 到 6。

您还可以使用 Amazon SES API V2 目标中的UpdateConfigurationSetEventDestination操作来创建和更新事件目标。

使用控制台将 Firehose 事件目标详细信息添加到配置集

1. 以下是步骤 7 中选择 Firehose 作为事件目标类型的详细说明，并假设您已完成创建事件目标中的所有之前的步骤。在选择 Firehose 目标类型，输入目标名称并启用事件发布后，Amazon Data Firehose 传输留窗格将显示，其字段将在以下步骤中得到处理。
2. 对于传输流，请选择现有的 Firehose 传输流，或选择创建新流以使用 Firehose 控制台创建新的传输流。

   有关使用 Firehose 控制台创建流的信息，请参阅《Amazon Data Firehose 开发人员指南》中的创建 Amazon Kinesis Firehose 传输流。
3. 对于 Identity and Access Management（IAM）角色，请选择一个 Amazon SES 有权限代表您发布到 Firehose 的 IAM 角色。您可以选择现有角色，或者让 Amazon SES 为您创建一个角色，也可以创建自己的角色。

   如果您选择现有角色或创建自己的角色，您必须手动修改该角色的策略，授予该角色访问 Firehose 传输流的权限，并授予 Amazon SES 代入该角色的权限。有关示例策略，请参阅 授予 Amazon SES 发布到 Firehose 传输流的权限。
4. 选择下一步。
5. 在审核屏幕上，如果您对定义事件目标的方式感到满意，请选择添加目标。

有关如何使用 `UpdateConfigurationSetEventDestination` API 添加 Firehose 事件目标的信息，请参阅 Amazon Simple Email Service API 参考。

授予 Amazon SES 发布到 Firehose 传输流的权限

要允许 Amazon SES 向您的 Firehose 传输流发布记录，您必须使用 AWS Identity and Access Management （IAM）角色，并附加或修改该角色的权限策略和信任策略。权限策略允许该角色向您的 Firehose 传输流发布记录，信任策略允许 Amazon SES 代入该角色。

本部分提供有关两个策略的一些示例。有关向 IAM 角色添加策略的信息，请参阅《IAM 用户指南》中的修改角色。

权限策略

以下权限策略允许角色发布数据记录到您的 Firehose 传输流。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": [
        "firehose:PutRecordBatch"
      ],
      "Resource": [
        "arn:aws:firehose:delivery-region:111122223333:deliverystream/delivery-
stream-name"
      ]
    }
  ]
}
```

对前面的策略示例进行以下更改：

- *delivery-region*替换为您创建 Firehose 传送流的 AWS 区域。
- 将 *111122223333* 替换为您的 AWS 账户 ID。
- *delivery-stream-name*替换为 Firehose 传送流的名称。

信任策略

以下信任策略使 Amazon SES 能够代入该角色。

JSON

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

对前面的策略示例进行以下更改：

- *delivery-region*替换为您创建 Firehose 传送流的 AWS 区域。
- 将 *111122223333* 替换为您的 AWS 账户 ID。
- 替换为*configuration-set-name*与 Firehose 交付流关联的配置集的名称。

为活动发布设置一个 Amazon EventBridge 目的地

Amazon EventBridge 事件目标会通知您有关您在配置集中指定的电子邮件发送事件。SES 生成电子邮件发送事件，并将您在创建事件目的地时定义的事件发送到 EventBridge 默认事件总线。事件总线是接收事件并将其传送到多个目的地的路由器。您可以在中详细了解如何将电子邮件发送事件与 Amazon

EventBridge 集成使用监控 EventBridge。由于只能在配置集中设置 EventBridge事件目的地，因此必须先创建配置集，然后才能将事件目标添加到配置集中。

本节中的过程说明如何将 EventBridge 事件目标详细信息添加到配置集中，并假设您已经完成了中的步骤 1 到步骤 6 创建事件目标。

您还可以使用 Amazon SES API V2 中的UpdateConfigurationSetEventDestination操作来创建和修改事件目的地。

使用控制台将 EventBridge 事件目标详细信息添加到配置集中

1. 以下是在步骤 7 中选择 EventBridge 活动目的地类型的详细说明，并假设您已完成中之前的所有步骤创建事件目标。选择亚马逊 EventBridge目标类型、输入目标名称并启用活动发布后，将显示亚马逊 EventBridge 事件总线信息窗格。
2. 选择下一步。
3. 在审核屏幕上，如果您对定义事件目标的方式感到满意，请选择添加目标。这将打开活动目的地的摘要页面，成功横幅将确认您的活动目的地是否已成功创建或修改。

针对事件发布设置 Amazon Pinpoint 事件目标

Amazon Pinpoint 事件目标会通知您有关您在配置集中指定的电子邮件发送事件。由于 Amazon Pinpoint 事件目标只能在配置集中设置，您必须先创建配置集，再将事件目标添加到该配置集。

本节中的过程演示如何将 Amazon Pinpoint 事件目标详细信息添加到配置集，并假设您已完成创建事件目标中的步骤 1 到 6。

您还可以使用 Amazon SES API V2 中的UpdateConfigurationSetEventDestination操作来创建和修改事件目标。

您在 Amazon Pinpoint 项目中配置的通道类型需要额外支付费用。有关更多信息，请参阅 Amazon Pinpoint 定价。

使用控制台将 Amazon Pinpoint 事件目标详细信息添加到配置集

1. 以下是步骤 7 中选择 Amazon Pinpoint 作为事件目标类型的详细说明，并假设您已完成创建事件目标中所有之前的步骤。

> **Note**
>
> Amazon Pinpoint 不支持 Delivery delays（送达延迟）或 Subscriptions（订阅）事件类型。

在选择 Amazon Pinpoint 目标类型，输入目标名称并启用事件发布后，Amazon Pinpoint 项目详细信息窗格将显示，其字段将在以下步骤中得到处理。

2. 对于 Project（项目），选择现有的 Amazon Pinpoint 项目，或者选择 Create a new project in Amazon Pinpoint（在 Amazon Pinpoint 中创建新项目）以创建新项目。

   有关创建项目的信息，请参阅《Amazon Pinpoint 用户指南》中的创建项目。

3. 选择下一步。
4. 在审核屏幕上，如果您对定义事件目标的方式感到满意，请选择添加目标。这将打开活动目的地的摘要页面，成功横幅将确认您的活动目的地是否已成功创建或修改。

针对事件发布设置 Amazon SNS 事件目标

Amazon SNS 事件目标会通知您有关您在配置集中指定的电子邮件发送事件。由于 Amazon SNS 事件目标只能在配置集中设置，您必须先创建配置集，再将事件目标添加到该配置集。

本节中的过程演示如何将 Amazon SNS 事件目标详细信息添加到配置集，并假设您已完成 创建事件目标 中的步骤 1 到 6。

您还可以使用 Amazon SES API V2 中的UpdateConfigurationSetEventDestination操作来创建和修改事件目的地。

> **Note**
>
> 也可以通过 Amazon SNS 为任何经过验证的发送身份设置有关退信、投诉和送达的反馈通知。有关更多信息，请参阅the section called “配置 Amazon SNS 通知”。

向订阅 Amazon SNS 主题的端点发送消息会产生额外费用。有关更多信息，请参阅 Amazon SNS 定价。

使用控制台将 Amazon SNS 事件目标详细信息添加到配置集

1. 以下是步骤 7 中选择 Amazon SNS 作为事件目标类型的详细说明，并假设您已完成创建事件目标中的所有之前的步骤。在选择 Amazon SNS 目标类型，输入目标名称并启用事件发布后，Amazon Simple Notification Service（SNS）主体窗格将显示，其字段将在以下步骤中得到处理。
2. 对于 SNS topic (SNS 主题)，选择现有的 Amazon SNS 主题，或选择 Create SNS topic (创建 SNS 主题) 以创建一个新主题。

   有关创建主题的信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的创建主题。

   > **Important**
   >
   > 当您使用 Amazon SNS 创建主题时，对于 Type (类型)，仅选择 Standard (标准)。（SES 不支持 FIFO 类型主题。）
3. 选择下一步。
4. 在审核屏幕上，如果您对定义事件目标的方式感到满意，请选择添加目标。这将打开活动目的地的摘要页面，成功横幅将确认您的活动目的地是否已成功创建或修改。
5. 无论是创建新的 SNS 主题还是选择现有主题，都需要授予 SES 访问权限，才能向该主题发布通知。在上一步的事件目标的摘要页面中，从 Destination type（目标类型）列中选择 Amazon SNS - 这将转到 Amazon Simple Notification Service 控制台中的 Topics（主题）列表 - 从 Amazon SNS 控制台执行以下步骤：

   a. 选择您在上一步中创建或修改的 SNS 主题的名称。

   b. 在主题的详细信息屏幕上，选择编辑。

   c. 要授予 SES 发布主题通知的权限，请在 SNS 控制台的编辑主题屏幕上，展开访问策略，并在 JSON 编辑器中，添加以下权限策略：

      JSON

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

对前面的策略示例进行以下更改：

- *topic_region*替换为您创建 SNS 主题的 AWS 区域。
- *111122223333*替换为您的 AWS 账户 ID。
- *topic_name*替换为您的 SNS 主题的名称。
- 替换为*configuration-set-name*与 SNS 事件目标关联的配置集的名称。

d. 选择保存更改。

步骤 3：在发送电子邮件时指定配置集

在创建配置集和添加事件目标后，事件发布的最后一步是发送您的电子邮件。

要发布与电子邮件相关联的事件，您必须提供配置集名称以便与电子邮件相关联。您还可以选择性地提供邮件标签以对电子邮件进行分类。

您可以以电子邮件发送 API 的参数、特定于 Amazon SES 的电子邮件标头或 MIME 邮件中的自定义标头的形式向 Amazon SES 提供这些信息。您选择的方法取决于所使用的电子邮件发送接口，如下表所示。

电子邮件发送接口	发布事件的方式
SendEmail	API 参数

电子邮件发送接口	发布事件的方式
`SendTemplatedEmail`	API 参数
`SendBulkTemplatedEmail`	API 参数
`SendCustomVerificationEmail`	API 参数
`SendRawEmail`	API 参数、特定于 Amazon SES 的电子邮件标头或自定义 MIME 标头 **Important** 如果同时使用标头和 API 参数指定邮件标签，Amazon SES 仅使用通过 API 参数提供的邮件标签。Amazon SES 不会合并由 API 参数和标头指定的邮件标签。
SMTP 接口	特定于 Amazon SES 的电子邮件标头

以下部分介绍如何使用标头和使用 API 参数指定配置集和邮件标签。

- 使用 Amazon SES API 参数
- 使用特定于 Amazon SES 的电子邮件标头
- 使用自定义电子邮件标头

Note

还可以选择性地在电子邮件标头中包含邮件标签。邮件标签可以包含数字 0-9、字母 A-Z（大写和小写字母）、连字符（-）和下划线（_）。

使用 Amazon SES API 参数

要将SendEmail、SendTemplatedEmail、SendBulkTemplatedEmailSendCustomVerificationEmail、或SendRawEmail与事件发布一起使用，您可以通过将调用的数据结构传递给 API 调用MessageTag来指定配置集ConfigurationSet和消息标签。

有关使用 Amazon SES API 的更多信息，请参阅 Amazon Simple Email Service API 参考。

使用特定于 Amazon SES 的电子邮件标头

在使用 SendRawEmail 或 SMTP 接口时，您可以通过将特定于 Amazon SES 的标头添加到电子邮件来指定配置集和邮件标签。Amazon SES 将删除标头，然后再发送电子邮件。下表显示了要使用的标头的名称。

事件发布信息	标题
配置集	X-SES-CONFIGURATION-SET
邮件标签	X-SES-MESSAGE-TAGS

以下示例显示了在提交至 Amazon SES 的原始电子邮件中标头的具体形式。

```
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--
```

使用自定义电子邮件标头

尽管您必须使用特定于 Amazon SES 的标头 `X-SES-CONFIGURATION-SET` 指定配置集名称，但您可以使用自己的 MIME 标头指定邮件标签。

> **Note**
>
> Amazon SES 事件发布使用的标头名称和值必须是 ASCII 码。如果您为 Amazon SES 事件发布指定非 ASCII 标头名称或值，则电子邮件发送调用仍会成功，但不会向亚马逊发送事件指标。 CloudWatch

使用 Amazon SES 事件数据

您设置事件发布并为发送的电子邮件指定配置集后，可以从您在设置与电子邮件关联的配置集时指定的事件目标检索电子邮件发送事件。

本节介绍如何从亚马逊 CloudWatch 和亚马逊 Data Firehose 检索您的电子邮件发送事件，以及如何解释亚马逊 SNS 提供的事件数据。

- 正在从中检索 Amazon SES 事件数据 CloudWatch
- 从 Firehose 检索 Amazon SES 事件数据
- 解释来自 Amazon SNS 的 Amazon SES 事件数据

正在从中检索 Amazon SES 事件数据 CloudWatch

Amazon SES 可以发布您向亚马逊发送电子邮件事件的指标 CloudWatch。当您将事件数据发布到时 CloudWatch，它会将这些指标作为一组有序的时间序列数据提供。您可以使用这些指标来监控您的电子邮件发送的性能。例如，您可以监控投诉指标并设置 CloudWatch 警报，使其在指标超过特定值时触发。

Amazon SES 可以在两个级别上将这些事件发布到 CloudWatch：

- 在您的 AWS 账户 — 这些粗略指标与您使用 Amazon SES 控制台和 `GetSendStatistics` API 监控的指标相对应，是您的整个指标的总和。 AWS 账户 Amazon SES CloudWatch 会自动将这些指标发布到。
- 精细 – 这些指标将按您使用邮件标记定义的电子邮件特征分类。要将这些指标发布到 CloudWatch，您必须使用事件目标设置 CloudWatch 事件发布，并在发送电子邮件时指定配置集。您也可以指定邮件标记或使用 Amazon SES 自动提供的 auto-tags。

此部分介绍可用指标以及如何在 CloudWatch 中查看这些指标。

可用指标

您可以将以下 Amazon SES 电子邮件发送指标发布到 CloudWatch：

- Send（发送）– 发送请求成功，Amazon SES 将尝试将邮件发送到收件人的邮件服务器。（如果使用账户级别或全局抑制，SES 仍会将其计为发送，但会抑制送达。）
- RenderingFailure— 由于模板渲染问题，电子邮件未发送。当模板数据丢失或模板参数与数据不匹配时，可能会发生此事件类型。（此事件类型仅在您使用 `SendTemplatedEmail` 或 `SendBulkTemplatedEmail` API 操作发送电子邮件时发生。）
- Reject（拒绝）– Amazon SES 已接受电子邮件，但确定它包含病毒，而未尝试将其发送到收件人的邮件服务器。
- Delivery（送达）– Amazon SES 成功将电子邮件发送到收件人的邮件服务器。
- 退信 – 收件人的邮件服务器永久拒绝了电子邮件的硬退信。（仅当 S ES 不再尝试发送电子邮件时，才会包含软退件。 通常，这些软退件表示投递失败，尽管在某些情况下，即使邮件成功到达收件人收件箱，也可以返回软退件。 这通常发生在收件人发送 out-of-office自动回复时。 在这篇 re AWS：Post 文章中了解有关软反弹的更多信息。）
- Complaint（投诉）– 电子邮件已成功送达收件人的邮件服务器，但收件人将其标记为垃圾邮件。
- DeliveryDelay— 由于出现临时问题，无法将电子邮件发送到收件人的邮件服务器。例如，当收件人的收件箱已满，或者当接收电子邮件服务器遇到临时问题时，可能会发生传送延迟。
- Subscription（订阅）– 电子邮件已成功发送，但收件人通过单击电子邮件标头中的 `List-Unsubscribe` 或脚注中的 `Unsubscribe` 链接更新了订阅首选项。
- Open（打开）– 收件人已收到邮件并在其电子邮件客户端中打开了邮件。
- Click（单击）– 收件人单击了电子邮件中包含的一个或多个链接。

可用维度

CloudWatch 使用您在向 Amazon SES 中的配置集添加 CloudWatch 事件目标时指定的维度名称。有关更多信息，请参阅 为 CloudWatch 活动发布设置活动目的地。

在 CloudWatch 控制台中查看 Amazon SES 指标

以下过程介绍如何使用 CloudWatch 控制台查看 Amazon SES 事件发布指标。

使用 CloudWatch 控制台查看指标

1. 登录 AWS Management Console 并打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 如果需要，可以更改区域。在导航栏中，选择您的 AWS 资源所在的区域。有关更多信息，请参阅区域和端点。
3. 在导航窗格中，选择所有指标。
4. 在指标窗格中，选择 SES。
5. 选择要查看的指标。要查看精细事件发布指标，请选择在设置 CloudWatch 事件目标时指定的维度组合。要详细了解如何使用查看指标 CloudWatch，请参阅使用 Amazon CloudWatch 指标。

要查看指标，请使用 AWS CLI

- 在命令提示符处输入下面的命令：

```
aws cloudwatch list-metrics --namespace "AWS/SES"
```

从 Firehose 检索 Amazon SES 事件数据

Amazon SES 将电子邮件发送事件作为 JSON 记录发布到 Firehose。然后，Firehose 会将记录发布到你在 Firehose 中设置传送流时选择的 AWS 服务目的地。有关设置 Firehose 传输流的更多信息，请参阅《Amazon Data Firehose Developer Guide》中的 Creating an Firehose Delivery Stream。

本节中的主题：

- Amazon SES 发布到 Firehose 的事件数据的内容
- Amazon SES 发布到 Firehose 的事件数据的示例

Amazon SES 发布到 Firehose 的事件数据的内容

Amazon SES 以 JSON 格式将电子邮件发送事件记录发布到 Amazon Data Firehose。在将事件发布到 Firehose 时，Amazon SES 在每个 JSON 记录后跟一个换行符。

您可以在 Amazon SES 发布到 Firehose 的事件数据的示例 中找到所有这些通知类型的示例记录。

本节中的主题

- 顶级 JSON 对象

- 邮件对象
- 退信对象
- 投诉对象
- 送达对象
- 发送对象
- 拒绝对象
- 打开对象
- 单击对象
- 呈现失败对象
- DeliveryDelay 对象
- 订阅对象

顶级 JSON 对象

电子邮件发送事件记录中的顶级 JSON 对象包含以下字段。

字段名称	描述
eventType	一个描述事件类型的文本字符串。 可能的值：Bounce、Complaint 、Delivery、Send、Reject、Open、Click、Rend Failure、DeliveryDelay 或 Subscript ion 。 如果尚未设置事件发布，则此字段将命名为 notificationType 。
mail	JSON 对象，包含有关生成事件的电子邮件的信息。
bounce	仅当 eventType 为 Bounce 时，此字段才存在。它包含有关退信的信息。
complaint	仅当 eventType 为 Complaint 时，此字段才存在。它包含有关投诉的信息。

字段名称	描述
delivery	仅当 eventType 为 Delivery 时，此字段才存在。它包含有关送达的信息。
send	仅当 eventType 为 Send 时，此字段才存在。
reject	仅当 eventType 为 Reject 时，此字段才存在。它包含有关拒绝的信息。
open	仅当 eventType 为 Open 时，此字段才存在。它包含有关打开事件的信息。
click	仅当 eventType 为 Click 时，此字段才存在。它包含有关点击事件的信息。
failure	仅当 eventType 为 Rendering Failure 时，此字段才存在。它包含有关呈现失败事件的信息。
deliveryDelay	仅当 eventType 为 DeliveryDelay 时，此字段才存在。它包含有关延迟送达电子邮件的信息。
subscription	仅当 eventType 为 Subscription 时，此字段才存在。它包含有关订阅首选项的信息。

邮件对象

每个电子邮件发送事件记录都包含有关 mail 对象中的原始电子邮件的信息。包含有关 mail 对象的信息的 JSON 对象具有以下字段。

字段名称	描述
timestamp	发送消息的日期和时间，采用 ISO86 01 格式（YYYY-MM--: mm: ss.sz DDThh）。

字段名称	描述
messageId	Amazon SES 分配给电子邮件的唯一 ID。Amazon SES 在您发送邮件时已向您返回此值。 **Note** 此邮件 ID 已由 Amazon SES 分配。您可以在 mail 对象的 headers 和 commonHeaders 字段中找到原始电子邮件的邮件 ID。
source	发送邮件的电子邮件地址（信封 MAIL FROM 地址）。
sourceArn	已用于发送电子邮件的身份的 Amazon Resource Name（ARN）。在发送授权的情况下，sourceArn 是身份拥有者授权委托发件人用于发送电子邮件的身份的 ARN。有关发送授权的更多信息，请参阅电子邮件身份验证方法。
sendingAccountId	用于发送电子邮件的账户的账户 ID。 AWS 在发送授权的情况下，sendingAccountId 是委托发件人的账户 ID。
destination	作为原始邮件的收件人的电子邮件地址的列表。
headersTruncated	一个字符串，指定标头是否会在通知中被截断，如果标头大于 10 KB，则会发生截断。可能的值为 true 和 false。

字段名称	描述
headers	电子邮件的原始标头的列表。列表中的每个标头均有一个 name 字段和一个 value 字段。 **Note** headers 字段内的任何邮件 ID 均来自您传递至 Amazon SES 的原始邮件。Amazon SES 随后分配给电子邮件的邮件 ID 位于 messageId 对象的 mail 字段中。
commonHeaders	电子邮件的原始常用标头的映射。 **Note** commonHeaders 字段中的任何邮件 ID 是 Amazon SES 随后在 mail 对象的 messageId 字段中分配给邮件的邮件 ID。
tags	与电子邮件关联的标签的列表。

退信对象

包含有关 Bounce 事件的信息的 JSON 对象将始终具有以下字段。

字段名称	描述
bounceType	退信的类型，由 Amazon SES 确定。
bounceSubType	退信的子类型，由 Amazon SES 确定。
bouncedRecipients	包含已退回的原始邮件收件人相关信息的列表。

字段名称	描述
timestamp	互联网服务提供商发送退回通知的日期和时间，采用 ISO86 01 格式（YYYY-MM--: mm: ss.sz DDThh）。
feedbackId	退信的唯一 ID。
reportingMTA	DSN 中的 Reporting-MTA 字段的值。这是尝试执行 DSN 所述的传输、中继或网关操作的 Message Transfer Authority（MTA）的值。 **Note** 此字段仅在传输状态通知（DSN）附加到退信时显示。

退信的收件人

退信事件可能与一个收件人或多个收件人有关。bouncedRecipients 字段包含一系列对象（与退信事件相关的每个收件人各有一个对象）并将始终包含以下字段。

字段名称	描述
emailAddress	收件人的电子邮件地址。如果 DSN 可用，这将是 DSN 中的 Final-Recipient 字段的值。

（可选）如果 DSN 已附加到退信，则以下字段也可能存在。

字段名称	描述
action	DSN 中的 Action 字段的值。这表示报告 MTA 因尝试将邮件传输至此收件人而需执行的操作。

字段名称	描述
status	DSN 中的 Status 字段的值。这是每个收件人与传输无关的状态代码，用于指示邮件的传输状态。
diagnosticCode	报告 MTA 发放的状态代码。这是 DSN 中的 Diagnostic-Code 字段的值。此字段可能不在 DSN 中（因此也不在 JSON 中）。

退信类型

每个退信事件都属于下表中所示的类型之一。

事件发布系统仅发布不再由 Amazon SES 重试的硬退信和软退信。收到标有 Permanent 的退信时，应从邮件列表中删除对应的电子邮件地址；您将来无法向这些地址发送邮件。当某封邮件被软退回若干次并且 Amazon SES 已停止尝试再次传送它之后，系统将向您发送 Transient 退信。您将来或许能够成功重新发送到最初导致了 Transient 退信的地址。

bounceType	bounceSubType	描述
Undetermined	Undetermined	Amazon SES 无法确定特定的退信原因。
Permanent	General	Amazon SES 收到了一封常规硬退信。如果您收到此类退信，您应从邮件列表中删除收件人的电子邮件地址。
Permanent	NoEmail	Amazon SES 收到永久硬退信，因为目标电子邮件地址不存在。如果您收到此类退信，您应从邮件列表中删除收件人的电子邮件地址。
Permanent	Suppressed	Amazon SES 已禁止发送邮件到此地址，因为该地址最近出现过做作为无效地址被退回的历史记录。要覆盖全局黑名单，请参阅 使用 Amazon SES 账户级黑名单。

bounceType	bounceSubType	描述
Permanent	OnAccountSuppressionList	Amazon SES 已禁止发送到此地址，因为该地址已被加入账户级黑名单。这不计入您的跳出率指标。
Transient	General	Amazon SES 收到常规退信。将来，您也许能够向该收件人成功发送电子邮件。
Transient	MailboxFull	Amazon SES 收到邮箱完全退信。将来，您也许能够向该收件人成功发送电子邮件。
Transient	MessageTooLarge	Amazon SES 收到消息太大退信。如果您减小邮件的大小，则也许能够向该收件人成功发送电子邮件。
Transient	CustomTimeoutExceeded	Amazon SES 无法在电子邮件发件人指定的时间内成功传送电子邮件。（退回消息将指定在定义的 TTL 内任何可能的传送尝试失败的原因。）
Transient	ContentRejected	Amazon SES 收到内容拒绝退信。如果您更改邮件的内容，则也许能够向该收件人成功发送电子邮件。
Transient	AttachmentRejected	Amazon SES 收到附件拒绝退信。如果您删除或更改附件，则也许能够向该收件人成功发送电子邮件。

投诉对象

包含有关 Complaint 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
complainedRecipients	包含可能已提交投诉的收件人的相关信息列表。

字段名称	描述
timestamp	互联网服务提供商发送投诉通知的日期和时间，采用 ISO86 01 格式（YYYY-MM--: mm: ss.sz DDThh）。
feedbackId	投诉的唯一 ID。
complaintSubType	投诉的子类型，由 Amazon SES 确定。

此外，如果反馈报告已附加到投诉，则以下字段也可能存在。

字段名称	描述
userAgent	反馈报告中的 User-Agent 字段的值。此值表示生成了报告的系统的名称和版本。
complaintFeedbackType	从 ISP 收到的反馈报告中的 Feedback-Type 字段的值。此值包含反馈的类型。
arrivalDate	反馈报告中Arrival-Date 或Received-Date 字段的值，格式为 ISO86 01（YYYY-MM--: mm: ss.sz）。DDThh此字段可能不在报告中（因此也不在 JSON 中）。

已投诉的收件人

complainedRecipients 字段包含可能已提交投诉的收件人的列表。

> **Important**
>
> 由于大多数 ISPs 人会从投诉通知中删除提交投诉的收件人的电子邮件地址，因此此列表包含有关可能发送投诉的收件人的信息，这些信息基于原始邮件的收件人和我们收到投诉的互联网服务提供商。Amazon SES 对原始邮件执行查找以确定此收件人列表。

此列表中的 JSON 对象包含以下字段。

字段名称	描述
emailAddress	收件人的电子邮件地址。

投诉类型

根据complaintFeedbackType互联网编号分配机构网站，您可在由报告 ISP 分配的 字段中看到以下投诉类型：

字段名称	描述
abuse	指示未经请求的电子邮件或某种其他类型的电子邮件滥用。
auth-failure	电子邮件身份验证失败报告。
fraud	指示某种欺诈或网络钓鱼活动。
not-spam	指示提供报告的实体不会将邮件视为垃圾邮件。这可用于更正被错误地标记或分类为垃圾邮件的邮件。
other	指示不适合其他已注册类型的任何其他反馈。
virus	报告在原始邮件中发现病毒。

送达对象

包含有关 Delivery 事件的信息的 JSON 对象将始终具有以下字段。

字段名称	描述
timestamp	Amazon SES 将电子邮件发送到收件人的邮件服务器的日期和时间，格式为 ISO8601（YYYY-MM--: mm: ss.sz）。DDThh

字段名称	描述
processingTimeMillis	从 Amazon SES 接受来自发件人的请求到 Amazon SES 将邮件传递到收件人的邮件服务器的时间（以毫秒为单位）。
recipients	传送事件应用于的预定收件人的列表。
smtpResponse	从 Amazon SES 接受电子邮件的远程 ISP 的 SMTP 响应消息。此消息因电子邮件、接收邮件服务器以及接收 ISP 而异。
reportingMTA	发送邮件的 Amazon SES 邮件服务器的主机名。
remoteMtaIp	Amazon SES 将电子邮件送达的 MTA 的 IP 地址。

发送对象

包含有关 send 事件的信息的 JSON 对象始终为空。

拒绝对象

包含有关 Reject 事件的信息的 JSON 对象将始终具有以下字段。

字段名称	描述
reason	电子邮件被拒绝的原因。唯一可能的值为 Bad content，这意味着 Amazon SES 检测到该电子邮件含有病毒。当某个邮件被拒绝时，Amazon SES 会停止处理该邮件，并且不会尝试将邮件发送到收件人的邮件服务器。

打开对象

包含有关 Open 事件的信息的 JSON 对象将始终包含以下字段。

字段名称	描述
ipAddress	收件人的 IP 地址。
timestamp	公开事件发生的日期和时间采用 ISO86 01 格式（YYYYY-MM--: mm: ss.sz）。DDThh
userAgent	收件人用于打开电子邮件中的设备或电子邮件客户端的用户代理。

单击对象

包含有关 Click 事件的信息的 JSON 对象将始终包含以下字段。

字段名称	描述
ipAddress	收件人的 IP 地址。
timestamp	点击事件发生的日期和时间，格式为 ISO86 01（YYYY-MM--: mm: ss.sz）。DDThh
userAgent	收件人单击电子邮件中链接时使用的客户端的用户代理。
link	收件人点击的链接的 URL。
linkTags	使用 ses:tags 属性添加到链接的标签的列表。有关向电子邮件中的链接添加标签的更多信息，请参阅 Amazon SES 电子邮件发送指标 FAQs中的问题 5：我能否用唯一标识符来标记链接？。

呈现失败对象

包含有关 Rendering Failure 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
templateName	用于发送电子邮件的模板的名称。
errorMessage	提供有关呈现失败详细信息的消息。

DeliveryDelay 对象

包含有关 DeliveryDelay 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
delayType	延迟的类型。可能的值有： • InternalFailure— Amazon SES 内部问题导致消息延迟。 • General – SMTP 对话期间发生了常规故障。 • MailboxFull— 收件人的邮箱已满，无法接收其他消息。 • SpamDetected— 收件人的邮件服务器检测到来自您的帐户的大量未经请求的电子邮件。 • RecipientServerError— 收件人的电子邮件服务器暂时出现问题，导致邮件无法传送。 • IPFailure— 发送邮件的 IP 地址被收件人的电子邮件提供商屏蔽或限制。 • TransientCommunicationFailure— 在与收件人的电子邮件提供商进行 SMTP 对话期间，出现临时通信故障。 • BYOIPHostNameLookupUnavailable— Amazon SES 无法查找你的 IP 地址的 DNS 主机名。这种类型的延迟仅在您使用自带 IP 时发生。 • Undetermined – Amazon SES 无法确定送达延迟的原因。

字段名称	描述
	• SendingDeferral— Amazon SES 认为在内部推迟该消息是适当的。
delayedRecipients	包含有关电子邮件收件人的信息的对象。
expirationTime	Amazon SES 将停止尝试传输邮件的日期和时间。此值以 ISO 8601 格式显示。
reportingMTA	报告延迟的邮件传输代理（MTA）的 IP 地址。
timestamp	发生延迟的日期和时间，以 ISO 8601 格式显示。

延迟的收件人

delayedRecipients 对象包含以下值。

字段名称	描述
emailAddress	导致邮件送达延迟的电子邮件地址。
status	与送达延迟关联的 SMTP 状态代码。
diagnosticCode	接收邮件传输代理（MTA）提供的诊断代码。

订阅对象

包含有关 Subscription 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
contactList	联系人所在的列表的名称。
timestamp	互联网服务提供商发送订阅通知的日期和时间，采用 ISO86 01 格式（YYYY-MM--: mm: ss.sz DDThh）。

字段名称	描述
source	发送邮件的电子邮件地址（信封 MAIL FROM 地址）。
newTopicPreferences	一个 JSON 数据结构（映射），它指定联系人列表中所有主题的订阅状态，用于指示更改后的状态（联系人已订阅或已取消订阅）。
oldTopicPreferences	一个 JSON 数据结构（映射），它指定联系人列表中所有主题的订阅状态，用于指示更改前的状态（联系人已订阅或已取消订阅）。

新/旧主题首选项

newTopicPreferences 和 oldTopicPreferences 对象包含以下值。

字段名称	描述
unsubscribeAll	指定联系人是否已取消订阅联系人列表中的所有主题。
topicSubscriptionStatus	在 topicName 字段中指定主题的订阅状态，该字段指示该主题当前是否已订阅以接收来自 SES 的指定事件类型的通知。字段中可能的值为 OptIn（已订阅）或 OptOut（取消订阅）。subscriptionStatus
topicDefaultSubscriptionStatus	在 topicName 字段中指定主题的默认订阅状态，以确定默认情况下是订阅还是取消订阅添加到事件目标的新主题。字段中可能的值为 OptIn（默认情况下已订阅）或 OptOut（默认取消订阅）。subscriptionStatus

Amazon SES 发布到 Firehose 的事件数据的示例

本节提供了一些示例，介绍 Amazon SES 发布到 Firehose 的电子邮件发送事件记录类型。

本节中的主题：

> **Note**
>
> 在以下使用 tag 字段的示例中，它使用通过配置集发布事件，而 SES 支持发布所有事件类型的标签。如果直接使用关于身份的反馈通知，则 SES 不会发布标签。阅读有关在创建配置集或修改配置集时添加标签的内容。

退信记录

下面是 Amazon SES 发布到 Firehose 的 Bounce 事件记录的一个示例。

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

投诉记录

下面是 Amazon SES 发布到 Firehose 的 Complaint 事件记录的一个示例。

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

送达记录

下面是 Amazon SES 发布到 Firehose 的 Delivery 事件记录的一个示例。

```
{
  "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",
    "remoteMtaIp": "123.456.789.012",
    "reportingMTA": "mta.example.com"
  }
}
```

发送记录

下面是 Amazon SES 发布到 Firehose 的 Send 事件记录的一个示例。

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

拒绝记录

下面是 Amazon SES 发布到 Firehose 的 `Reject` 事件记录的一个示例。

```
{
  "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"
  }
}
```

打开记录

下面是 Amazon SES 发布到 Firehose 的 Open 事件记录的一个示例。

```
{
  "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"
  }
}
```

单击记录

下面是 Amazon SES 发布到 Firehose 的 `Click` 事件记录的一个示例。

```
{
  "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"
  }
}
```

呈现失败记录

下面是 Amazon SES 发布到 Firehose 的 `Rendering Failure` 事件记录的一个示例。

```
{
  "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 记录

下面是 Amazon SES 发布到 Firehose 的 `DeliveryDelay` 事件记录的一个示例。

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

订阅记录

下面是 Amazon SES 发布到 Firehose 的 Subscription 事件记录的一个示例。

```
{
  "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"
        }
      ]
```

```
    }
  }
}
```

解释来自 Amazon SNS 的 Amazon SES 事件数据

Amazon SES 将电子邮件发送事件作为 JSON 记录发布到 Amazon Simple Notification Service（Amazon SNS）。然后，Amazon SNS 将通知发送到订阅了与事件目标关联的 Amazon SNS 主题的终端节点。有关设置 Amazon SNS 主题和订阅的信息，请参阅 Amazon Simple Notification Service 开发人员指南中的入门。

有关记录内容 (例如示例记录) 的说明，请参阅以下部分。

- 事件记录内容
- 事件记录示例

Amazon SES 发布到 Amazon SNS 的事件数据的内容

Amazon SES 以 JSON 格式将电子邮件发送事件记录发布到 Amazon Simple Notification Service。

您可以在 Amazon SES 发布到 Amazon SNS 的事件数据示例 中找到所有这些通知类型的示例记录。

本节中的主题：

- 顶级 JSON 对象
- 邮件对象
- 退信对象
- 投诉对象
- 送达对象
- 发送对象
- 拒绝对象
- 打开对象
- 单击对象
- 呈现失败对象
- DeliveryDelay 对象
- 订阅对象

顶级 JSON 对象

电子邮件发送事件记录中的顶级 JSON 对象包含以下字段。事件类型决定了存在哪些其他对象。

字段名称	描述
eventType	一个描述事件类型的文本字符串。 可能的值：Bounce、Complaint 、Delivery、Send、Reject、Open、Click、Rend Failure、DeliveryDelay 或 Subscript ion 。 如果尚未设置事件发布，则此字段将命名为 notificationType 。
mail	JSON 对象，包含有关生成事件的电子邮件的信息。
bounce	仅当 eventType 为 Bounce 时，此字段才存在。它包含有关退信的信息。
complaint	仅当 eventType 为 Complaint 时，此字段才存在。它包含有关投诉的信息。
delivery	仅当 eventType 为 Delivery 时，此字段才存在。它包含有关送达的信息。
send	仅当 eventType 为 Send 时，此字段才存在。
reject	仅当 eventType 为 Reject 时，此字段才存在。它包含有关拒绝的信息。
open	仅当 eventType 为 Open 时，此字段才存在。它包含有关打开事件的信息。
click	仅当 eventType 为 Click 时，此字段才存在。它包含有关点击事件的信息。

字段名称	描述
failure	仅当 eventType 为 Rendering Failure 时，此字段才存在。它包含有关呈现失败事件的信息。
deliveryDelay	仅当 eventType 为 DeliveryDelay 时，此字段才存在。它包含有关延迟送达电子邮件的信息。
subscription	仅当 eventType 为 Subscription 时，此字段才存在。它包含有关订阅首选项的信息。

邮件对象

每个电子邮件发送事件记录都包含有关 mail 对象中的原始电子邮件的信息。包含有关 mail 对象的信息的 JSON 对象具有以下字段。

字段名称	描述
timestamp	发送消息的日期和时间，采用 ISO86 01 格式（YYYY-MM--: mm: ss.sz DDThh）。
messageId	Amazon SES 分配给电子邮件的唯一 ID。Amazon SES 在您发送邮件时已向您返回此值。 **Note** 此邮件 ID 已由 Amazon SES 分配。您可以在 mail 对象的 headers 和 commonHeaders 字段中找到原始电子邮件的邮件 ID。
source	发送邮件的电子邮件地址（信封 MAIL FROM 地址）。

字段名称	描述
sourceArn	已用于发送电子邮件的身份的 Amazon Resource Name（ARN）。在发送授权的情况下，sourceArn 是身份拥有者授权委托发件人用于发送电子邮件的身份的 ARN。有关发送授权的更多信息，请参阅电子邮件身份验证方法。
sendingAccountId	用于发送电子邮件的账户的账户 ID。 AWS 在发送授权的情况下，sendingAccountId 是委托发件人的账户 ID。
destination	作为原始邮件的收件人的电子邮件地址的列表。
headersTruncated	一个字符串，指定标头是否会在通知中被截断，如果标头大于 10 KB，则会发生截断。可能的值为 true 和 false。
headers	电子邮件的原始标头的列表。列表中的每个标头均有一个 name 字段和一个 value 字段。 **Note** headers 字段内的任何邮件 ID 均来自您传递至 Amazon SES 的原始邮件。Amazon SES 随后分配给电子邮件的邮件 ID 位于 messageId 对象的 mail 字段中。

字段名称	描述
commonHeaders	电子邮件的原始常用标头的映射。 **Note** commonHeaders 字段中的任何邮件 ID 是 Amazon SES 随后在 mail 对象的 messageId 字段中分配给邮件的邮件 ID。
tags	与电子邮件关联的标签的列表。

退信对象

包含有关 Bounce 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
bounceType	退信的类型，由 Amazon SES 确定。
bounceSubType	退信的子类型，由 Amazon SES 确定。
bouncedRecipients	包含已退回的原始邮件收件人相关信息的列表。
timestamp	互联网服务提供商发送退回通知的日期和时间，采用 ISO86 01 格式（YYYY-MM--: mm: ss.sz DDThh）。
feedbackId	退信的唯一 ID。
reportingMTA	DSN 中的 Reporting-MTA 字段的值。这是尝试执行 DSN 所述的传输、中继或网关操作的 Message Transfer Authority（MTA）的值。

字段名称	描述
	Note 此字段仅在传输状态通知（DSN）附加到退信时显示。

退信的收件人

退信事件可能与一个收件人或多个收件人有关。bouncedRecipients 字段包含一系列对象（其电子邮件地址导致退信的每个收件人各有一个对象）并包含以下字段。

字段名称	描述
emailAddress	收件人的电子邮件地址。如果 DSN 可用，这将是 DSN 中的 Final-Recipient 字段的值。

（可选）如果 DSN 已附加到退信，则以下字段也可能存在。

字段名称	描述
action	DSN 中的 Action 字段的值。这表示报告 MTA 因尝试将邮件传输至此收件人而需执行的操作。
status	DSN 中的 Status 字段的值。这是每个收件人与传输无关的状态代码，用于指示邮件的传输状态。
diagnosticCode	报告 MTA 发放的状态代码。这是 DSN 中的 Diagnostic-Code 字段的值。此字段可能不在 DSN 中（因此也不在 JSON 中）。

退信类型

每个退信事件都属于下表中所示的类型之一。

事件发布系统仅发布不再由 Amazon SES 重试的硬退信和软退信。收到标有 Permanent 的退信时，应从邮件列表中删除对应的电子邮件地址；您将来无法向这些地址发送邮件。当某封邮件被软退回若干次并且 Amazon SES 已停止尝试再次传送它之后，系统将向您发送 Transient 退信。您将来或许能够成功重新发送到最初导致了 Transient 退信的地址。

bounceType	bounceSubType	描述
Undetermined	Undetermined	Amazon SES 无法确定特定的退信原因。
Permanent	General	Amazon SES 收到了一封常规硬退信。如果您收到此类退信，您应从邮件列表中删除收件人的电子邮件地址。
Permanent	NoEmail	Amazon SES 收到永久硬退信，因为目标电子邮件地址不存在。如果您收到此类退信，您应从邮件列表中删除收件人的电子邮件地址。
Permanent	Suppressed	Amazon SES 已禁止发送邮件到此地址，因为该地址最近出现过做作为无效地址被退回的历史记录。要覆盖全局黑名单，请参阅 使用 Amazon SES 账户级黑名单。
Permanent	OnAccountSuppressionList	Amazon SES 已禁止发送到此地址，因为该地址已被加入账户级黑名单。这不计入您的跳出率指标。
Transient	General	Amazon SES 收到常规退信。将来，您也许能够向该收件人成功发送电子邮件。
Transient	MailboxFull	Amazon SES 收到邮箱完全退信。将来，您也许能够向该收件人成功发送电子邮件。
Transient	MessageTooLarge	Amazon SES 收到消息太大退信。如果您减小邮件的大小，则也许能够向该收件人成功发送电子邮件。
Transient	CustomTimeoutExceeded	Amazon SES 无法在电子邮件发件人指定的时间内成功传送电子邮件。（退回消息将指定在

bounceType	bounceSubType	描述
		定义的 TTL 内任何可能的传送尝试失败的原因。）
Transient	ContentRejected	Amazon SES 收到内容拒绝退信。如果您更改邮件的内容，则也许能够向该收件人成功发送电子邮件。
Transient	AttachmentRejected	Amazon SES 收到附件拒绝退信。如果您删除或更改附件，则也许能够向该收件人成功发送电子邮件。

投诉对象

包含有关 Complaint 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
complainedRecipients	包含可能已提交投诉的收件人的相关信息列表。
timestamp	互联网服务提供商发送投诉通知的日期和时间，采用 ISO86 01 格式（YYYY-MM--: mm: ss.sz DDThh）。
feedbackId	投诉的唯一 ID。
complaintSubType	投诉的子类型，由 Amazon SES 确定。

此外，如果反馈报告已附加到投诉，则以下字段也可能存在。

字段名称	描述
userAgent	反馈报告中的 User-Agent 字段的值。此值表示生成了报告的系统的名称和版本。
complaintFeedbackType	从 ISP 收到的反馈报告中的 Feedback-Type 字段的值。此值包含反馈的类型。

字段名称	描述
arrivalDate	反馈报告中Arrival-Date 或Received-Date 字段的值，格式为 ISO86 01（YYYY-MM--: mm: ss.sz）。DDThh此字段可能不在报告中（因此也不在 JSON 中）。

已投诉的收件人

complainedRecipients 字段包含可能已提交投诉的收件人的列表。

> **Important**
>
> 大多数 ISPs 会删除提交投诉的收件人的电子邮件地址。出于此原因，complainedRecipients 字段包含域中已向其发送电子邮件而该地址发布了投诉通知的所有人的列表。

此列表中的 JSON 对象包含以下字段。

字段名称	描述
emailAddress	收件人的电子邮件地址。

投诉类型

根据complaintFeedbackType互联网编号分配机构网站，您可在由报告 ISP 分配的 字段中看到以下投诉类型：

字段名称	描述
abuse	指示未经请求的电子邮件或某种其他类型的电子邮件滥用。
auth-failure	电子邮件身份验证失败报告。
fraud	指示某种欺诈或网络钓鱼活动。

字段名称	描述
not-spam	指示提供报告的实体不会将邮件视为垃圾邮件。这可用于更正被错误地标记或分类为垃圾邮件的邮件。
other	指示不适合其他已注册类型的任何其他反馈。
virus	报告在原始邮件中发现病毒。

投诉子类型

complaintSubType 字段的值可以为 null 或 OnAccountSuppressionList。如果该值为 OnAccountSuppressionList，则表示 Amazon SES 已接受邮件，但未尝试发送邮件，因为该地址已被加入账户级黑名单。

送达对象

包含有关 Delivery 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
timestamp	Amazon SES 将电子邮件发送到收件人的邮件服务器的日期和时间，格式为 ISO8601（YYYY-MM--: mm: ss.sz）。DDThh
processingTimeMillis	从 Amazon SES 接受来自发件人的请求到 Amazon SES 将邮件传递到收件人的邮件服务器的时间（以毫秒为单位）。
recipients	传送事件应用于的预定收件人的列表。
smtpResponse	从 Amazon SES 接受电子邮件的远程 ISP 的 SMTP 响应消息。此消息因电子邮件、接收邮件服务器以及接收 ISP 而异。
reportingMTA	发送邮件的 Amazon SES 邮件服务器的主机名。

字段名称	描述
remoteMtaIp	Amazon SES 将电子邮件送达的 MTA 的 IP 地址。

发送对象

包含有关 send 事件的信息的 JSON 对象始终为空。

拒绝对象

包含有关 Reject 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
reason	电子邮件被拒绝的原因。唯一可能的值为 Bad content，这意味着 Amazon SES 检测到该电子邮件含有病毒。当某个邮件被拒绝时，Amazon SES 会停止处理该邮件，并且不会尝试将邮件发送到收件人的邮件服务器。

打开对象

包含有关 Open 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
ipAddress	收件人的 IP 地址。
timestamp	公开事件发生的日期和时间采用 ISO86 01 格式（YYYYY-MM--: mm: ss.sz）。DDThh
userAgent	收件人用于打开电子邮件中的设备或电子邮件客户端的用户代理。

单击对象

包含有关 Click 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
ipAddress	收件人的 IP 地址。
timestamp	点击事件发生的日期和时间，格式为 ISO86 01（YYYY-MM--: mm: ss.sz）。DDThh
userAgent	收件人单击电子邮件中链接时使用的客户端的用户代理。
link	收件人点击的链接的 URL。
linkTags	使用 ses:tags 属性添加到链接的标签的列表。有关向电子邮件中的链接添加标签的更多信息，请参阅 Amazon SES 电子邮件发送指标 FAQs中的问题 5：我能否用唯一标识符来标记链接？。

呈现失败对象

包含有关 Rendering Failure 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
templateName	用于发送电子邮件的模板的名称。
errorMessage	提供有关呈现失败详细信息的消息。

DeliveryDelay 对象

包含有关 DeliveryDelay 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
delayType	延迟的类型。可能的值有： • InternalFailure— Amazon SES 内部问题导致消息延迟。

字段名称	描述
	• General – SMTP 对话期间发生了常规故障。 • MailboxFull— 收件人的邮箱已满，无法接收其他消息。 • SpamDetected— 收件人的邮件服务器检测到来自您的帐户的大量未经请求的电子邮件。 • RecipientServerError— 收件人的电子邮件服务器暂时出现问题，导致邮件无法传送。 • IPFailure— 发送邮件的 IP 地址被收件人的电子邮件提供商屏蔽或限制。 • TransientCommunicationFailure— 在与收件人的电子邮件提供商进行 SMTP 对话期间，出现临时通信故障。 • BYOIPHostNameLookupUnavailable— Amazon SES 无法查找你的 IP 地址的 DNS 主机名。这种类型的延迟仅在您使用自带 IP 时发生。 • Undetermined – Amazon SES 无法确定送达延迟的原因。 • SendingDeferral— Amazon SES 认为在内部推迟该消息是适当的。
`delayedRecipients`	包含有关电子邮件收件人的信息的对象。
`expirationTime`	Amazon SES 将停止尝试传输邮件的日期和时间。此值以 ISO 8601 格式显示。
`reportingMTA`	报告延迟的邮件传输代理（MTA）的 IP 地址。
`timestamp`	发生延迟的日期和时间，以 ISO 8601 格式显示。

延迟的收件人

`delayedRecipients` 对象包含以下值。

字段名称	描述
emailAddress	导致邮件送达延迟的电子邮件地址。
status	与送达延迟关联的 SMTP 状态代码。
diagnosticCode	接收邮件传输代理（MTA）提供的诊断代码。

订阅对象

包含有关 Subscription 事件的信息的 JSON 对象具有以下字段。

字段名称	描述
contactList	联系人所在的列表的名称。
timestamp	互联网服务提供商发送订阅通知的日期和时间，采用 ISO86 01 格式（YYYY-MM--: mm: ss.sz DDThh）。
source	发送邮件的电子邮件地址（信封 MAIL FROM 地址）。
newTopicPreferences	一个 JSON 数据结构（映射），它指定联系人列表中所有主题的订阅状态，用于指示更改后的状态（联系人已订阅或已取消订阅）。
oldTopicPreferences	一个 JSON 数据结构（映射），它指定联系人列表中所有主题的订阅状态，用于指示更改前的状态（联系人已订阅或已取消订阅）。

新/旧主题首选项

newTopicPreferences 和 oldTopicPreferences 对象包含以下值。

字段名称	描述
unsubscribeAll	指定联系人是否已取消订阅联系人列表中的所有主题。
topicSubscriptionStatus	在 topicName 字段中指定主题的订阅状态，该字段指示该主题当前是否已订阅以接收来自 SES 的指定事件类型的通知。字段中可能的值为 OptIn（已订阅）或 OptOut（取消订阅）。subscriptionStatus
topicDefaultSubscriptionStatus	在 topicName 字段中指定主题的默认订阅状态，以确定默认情况下是订阅还是取消订阅添加到事件目标的新主题。字段中可能的值为 OptIn（默认情况下已订阅）或 OptOut（默认取消订阅）。subscriptionStatus

Amazon SES 发布到 Amazon SNS 的事件数据示例

本节提供了一些示例，介绍 Amazon SES 发布到 Amazon SNS 的电子邮件发送事件记录类型。

本节中的主题：

- 退信记录
- 投诉记录
- 送达记录
- 发送记录
- 拒绝记录
- 打开记录
- 单击记录
- 呈现失败记录
- DeliveryDelay记录
- 订阅记录

> **Note**
>
> 在以下使用 tag 字段的示例中，它使用通过配置集发布事件，而 SES 支持发布所有事件类型的标签。如果直接使用关于身份的反馈通知，则 SES 不会发布标签。阅读有关在创建配置集或修改配置集时添加标签的内容。

退信记录

下面是 Amazon SES 发布到 Amazon SNS 的 Bounce 事件记录的一个示例。

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

```
}
```

投诉记录

下面是 Amazon SES 发布到 Amazon SNS 的 Complaint 事件记录的一个示例。

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

送达记录

下面是 Amazon SES 发布到 Amazon SNS 的 `Delivery` 事件记录的一个示例。

```
{
  "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",
    "remoteMtaIp": "123.456.789.012",
    "reportingMTA": "mta.example.com"
  }
}
```

发送记录

下面是 Amazon SES 发布到 Amazon SNS 的 Send 事件记录的一个示例。有些字段并不总是存在。例如，使用模板电子邮件，主题稍后会呈现并包含在后续活动中。

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

拒绝记录

下面是 Amazon SES 发布到 Amazon SNS 的 `Reject` 事件记录的一个示例。

```
{
  "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"
  }
}
```

打开记录

下面是 Amazon SES 发布到 Amazon SNS 的 Open 事件记录的一个示例。

```
{
  "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"
  }
}
```

单击记录

下面是 Amazon SES 发布到 Amazon SNS 的 `Click` 事件记录的一个示例。

```
{
  "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"
  }
}
```

呈现失败记录

下面是 Amazon SES 发布到 Amazon SNS 的 `Rendering Failure` 事件记录的一个示例。

```
{
  "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记录

下面是 Amazon SES 发布到 Amazon SNS 的 `DeliveryDelay` 事件记录的一个示例。

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

订阅记录

下面是 Amazon SES 发布到 Firehose 的 `Subscription` 事件记录的一个示例。

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

监控您的 Amazon SES 发件人声誉

Amazon SES 会主动跟踪多个可能导致您作为发件人的声誉受损或者可能导致您的电子邮件送达率下降的指标。我们考虑的两个重要指标是您的账户的退回邮件率和投诉率。如果账户的邮件退回率或投诉率过高，我们会对账户进行审核，或暂停账户发送电子邮件的功能。

由于您的退回邮件率和投诉率对您账户的运行状况至关重要，因此 Amazon SES 在 Amazon SES 控制台中加入了一个声誉指标页面来供您跟踪这些指标。声誉指标还可显示与损害您的发件人声誉的退回邮件或投诉无关的因素。例如，如果您向已知的垃圾邮件陷阱发送了电子邮件，则会在此控制面板上看到一条消息。

本节包含有关访问声誉指标、解释其包含的信息以及将系统设置为主动向您通知可能影响您的发件人声誉的因素的信息。

在本节中，您将找到以下主题：

- 使用声誉指标跟踪退回邮件率和投诉率
- 声誉指标消息
- 使用创建信誉监控警报 CloudWatch
- 专用 SNDS 指标 IPs
- 自动暂停电子邮件发送

使用声誉指标跟踪退回邮件率和投诉率

声誉指标控制台页面包含 Amazon SES 团队在确定各个账户的运行状况时看到的同一信息。

查看声誉指标

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在屏幕左侧的导航窗格中，选择声誉指标。

   控制面板将显示以下信息：

   - 账户状态 – 您的退回邮件率和投诉率的综合健康摘要。可能的值包括：
     - 正常 - 目前没有影响账户的问题。

  - 审查中 – 正在审查您的账户。如果审核期结束时仍未解决导致对账户进行审核的问题，我们会暂停账户发送电子邮件的功能。
  - 等待审查结束决定 – 正在审查您的账户。鉴于导致账户审核的问题的性质，我们需要对账户进行人工审核，然后才能采取任何进一步措施。
  - Sending paused (暂停发送功能) – 我们暂停了您的账户发送电子邮件的功能。在暂停账户发送电子邮件的功能期间，您将无法使用 Amazon SES 发送电子邮件。您可以要求我们对这一决定进行审核。有关请求审核的更多信息，请参阅Amazon SES 发送审核流程 FAQs。
  - Pending sending pause (暂停发送功能待处理) – 正在审核您的账户。导致账户审核的问题仍未解决。在这种情况下，我们通常会暂停账户发送电子邮件的功能。但是，由于您的账户的性质，我们需要审核您的账户，然后采取任何进一步措施。
- Bounce Rate – 从您的账户发送的导致查无此人的邮件的电子邮件所占的百分比。请参阅如何计算退回邮件率。
- Complaint Rate – 从您的账户发送的导致收件人将其报告为垃圾邮件的电子邮件所占的百分比。请参阅如何计算投诉率。

> **Note**
>
> Bounce Rate 和 Complaint Rate 部分还包含其各自的指标的状态消息。下面是可能为这些指标显示的状态消息的列表：
>
> - Healthy (正常) – 指标在正常范围内。
> - Almost healed (几乎已愈合) – 指标导致您的账户置于审核状态。由于审核期已开始，指标已保持低于最大速率。如果指标保持低于最大速率，则在审核期结束前，此指标的状态将更改为 Healthy。
> - Under review (正在审核) – 指标导致您的账户置于审核状态，并且仍然高于最大速率。如果审核期结束时仍未解决导致指标超出最大速率的问题，我们会暂停账户发送电子邮件的功能。
> - Sending pause (发送暂停) – 指标导致我们暂停您账户发送电子邮件的功能。在暂停账户发送电子邮件的功能期间，您将无法使用 Amazon SES 发送电子邮件。您可以要求我们对这一决定进行审核。有关提交审核请求的更多信息，请参阅Amazon SES 发送审核流程 FAQs。
> - Pending sending pause (暂停发送功能待处理) – 指标导致我们对账户进行审核。导致此审核期的问题尚未解决。这些问题可能导致我们暂停账户发送电子邮件的功能。在我们采取任何进一步行动之前，Amazon SES 团队的成员必须审核您的账户。

- Other Notifications (其他通知) – 如果您的账户遇到与退回邮件或投诉无关的声誉相关问题，此处将显示一条简短消息。要了解有关此区域中可显示的通知的更多信息，请参阅声誉指标消息。

声誉指标消息

Amazon SES 声誉指标控制台页面可提供与您的账户相关的重要指标。下面几节介绍此控制面板中可能显示的消息，并提供可用于解决与您的发件人声誉相关的问题的提示和信息。

本部分包含有关以下类型的通知的信息：

- 状态消息
- 退回邮件率通知
- 投诉率通知
- 反垃圾邮件组织通知
- 列表轰炸通知
- 直接反馈通知
- 域阻止列表通知
- 内部审查通知
- 邮箱提供商通知
- 收件人反馈通知
- 相关账户通知
- 垃圾邮件陷阱通知
- 脆弱站点通知
- 凭证泄露通知
- 其他通知

状态消息

使用声誉指标控制台页面时，您将看到一条描述您的 Amazon SES 账户状态的消息。以下是可能的账户状态值的列表：

- 正常 - 目前没有影响账户的问题。

- 审查中 – 正在审查您的账户。如果审核期结束时仍未解决导致对账户进行审核的问题，我们会暂停账户发送电子邮件的功能。
- 等待审查结束决定 – 正在审查您的账户。鉴于导致账户审核的问题的性质，我们需要对账户进行人工审核，然后才能采取任何进一步措施。
- Sending paused (暂停发送功能) – 我们暂停了您的账户发送电子邮件的功能。在暂停账户发送电子邮件的功能期间，您将无法使用 Amazon SES 发送电子邮件。您可以要求我们对这一决定进行审核。有关请求审核的更多信息，请参阅Amazon SES 发送审核流程 FAQs。
- Pending sending pause (暂停发送功能待处理) – 正在审核您的账户。导致账户审核的问题仍未解决。在这种情况下，我们通常会暂停账户发送电子邮件的功能。但是，由于您的账户的性质，我们需要审核您的账户，然后采取任何进一步措施。

此外，声誉指标页面的 Bounce Rate 和 Complaint Rate 部分将显示其各自指标的状态摘要。以下是可能的指标状态值的列表：

- Healthy (正常) – 指标在正常范围内。
- Almost healed (几乎已愈合) – 指标导致您的账户置于审核状态。由于审核期已开始，指标已保持低于最大速率。如果指标保持低于最大速率，则在审核期结束前，此指标的状态将更改为 Healthy。
- Under review (正在审核) – 指标导致您的账户置于审核状态，并且仍然高于最大速率。如果审核期结束时仍未解决导致指标超出最大速率的问题，我们会暂停账户发送电子邮件的功能。
- Sending pause (发送暂停) – 指标导致我们暂停您账户发送电子邮件的功能。在暂停账户发送电子邮件的功能期间，您将无法使用 Amazon SES 发送电子邮件。您可以要求我们对这一决定进行审核。有关提交审核请求的更多信息，请参阅Amazon SES 发送审核流程 FAQs。
- Pending sending pause (暂停发送功能待处理) – 指标导致我们对账户进行审核。导致此审核期的问题尚未解决。这些问题可能导致我们暂停账户发送电子邮件的功能。在我们采取任何进一步行动之前，Amazon SES 团队的成员必须审核您的账户。

退回邮件率通知

本节包含有关 Amazon SES 声誉指标页面中显示的退回邮件率通知的其他信息。

您为什么会收到此通知

您会收到此通知是因为您账户的邮件退回率太高。退回邮件率基于您的 Amazon SES 账户生成的查无此人的邮件数。电子邮件提供商将高邮件退回率视为发件人对其收件人列表管理不善以及发件人可能在发送未经请求的邮件的标志。

当电子邮件被发送到不存在的地址时，就会发生查无此人的邮件。Amazon SES 在此计算中不考虑软退回邮件（当接收方的地址暂时无法接收邮件时发生）。在此计算中也不会考虑您发送到经过验证的地址和域后退回的电子邮件，以及您发送到 Amazon SES 收件箱模拟器的电子邮件。

我们会根据典型的电子邮件量来计算您的邮件退回率。典型量是指能够代表您的典型发送操作的电子邮件量。为了对发件量大和发件量小的发件人公平起见，每个账户的典型量都不同并且会随着账户的发送模式的变化而变化。

为了获得最佳结果，请将邮件退回率保持在 5% 以下。退回邮件率高会影响电子邮件的传送。如果邮件退回率达到 5%，我们会自动将您的账户切换到审核状态。如果邮件退回率达到 10%，我们可能会暂停您的账户继续发送电子邮件的功能，直到您解决导致高邮件退回率的问题为止。

解决问题的方法

如果您还没有这样做，请设置捕获和管理邮件退回与投诉的流程。所有 Amazon SES 账户都需要设置这些过程。有关更多信息，请参阅 电子邮件程序成功指标。

接下来，确定退回的是哪些电子邮件地址，然后创建并实施减少或消除这些退回邮件的计划。如果您的账户已暂停发送电子邮件的功能，请登录 AWS Management Console 并前往 AWS 支持。回复我们代表您开立的问题。

如果您的账户正在接受审核

审核期结束时，如果账户的邮件退回率仍高于 10%，我们会暂停账户发送电子邮件的功能，直到您解决问题。

如果您已经实施了您认为可以解决问题的更改，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您对案例的回复中，描述您实施的更改。如果我们认可这些更改将降低邮件退回率，我们会将计算调整为仅考虑实施更改后收到的退回邮件。

如果账户发送电子邮件的功能已暂停

您可以要求我们重审这一决定。有关更多信息，请参阅 Amazon SES 发送审核流程 FAQs。

当您实施您认为可以解决问题的更改时，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。包括您为解决此问题而采取的措施的详细信息，以及您确保此问题不会再次发生的计划的详细信息。收到您的请求后，我们会审核您提供的信息，并视需要更改您的账户状态。

投诉率通知

本节包含有关 Amazon SES 声誉指标页面中显示的投诉率通知的其他信息。

您为什么会收到此通知

您会收到此通知是因为您账户的投诉率太高。投诉率是根据您的 Amazon SES 账户所产生的投诉量计算得出的。电子邮件提供商将高投诉率视为发件人对其收件人列表管理不善以及发件人可能在发送未经请求的邮件的标志。

当收件人将您发送的电子邮件标识为垃圾邮件时，就会发生投诉。当收件人在其电子邮件客户端中使用“报告垃圾邮件”按钮时，通常就会发生这种情况。在此计算中不考虑您发送到 Amazon SES 收件箱模拟器的电子邮件所产生的投诉。

我们会根据典型的电子邮件量来计算您的投诉率。典型量是指能够代表您的典型发送操作的电子邮件量。为了对发件量大和发件量小的发件人公平起见，每个账户的典型量都不同并且会随着账户的发送模式的变化而变化。

为了获得最佳结果，请将投诉率保持在 0.1% 以下。较高的投诉率会影响电子邮件的传送。如果投诉率达到 0.1%，我们会自动将您的账户切换到审核状态。如果投诉率达到 0.5%，我们可能会暂停账户继续发送电子邮件的功能，直到您解决导致高投诉率的问题为止。

解决问题的方法

如果您还没有这样做，请设置捕获和管理邮件退回与投诉的流程。所有 Amazon SES 账户都需要设置这些过程。有关更多信息，请参阅 电子邮件程序成功指标。

接下来，确定您发送的哪些邮件将导致投诉，并实施减少这些投诉的计划。如果您的账户已暂停发送电子邮件的权限，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的案例

虽然您应立即停止发送到被投诉的地址，但确定导致收件人发出投诉的因素也很重要。确定这些因素后，针对这些因素调整您的电子邮件发送行为。

如果您的账户正在接受审核

审核期结束时，如果账户的投诉率仍高于 0.5%，我们会暂停账户发送电子邮件的功能，直到您解决问题。

如果您已经实施了您认为可以解决问题的更改，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您对案例的回复中，描述您实施的更改。如果我们认可这些更改将降低投诉率，我们会将计算调整为仅考虑实施更改后收到的投诉。

如果账户发送电子邮件的功能已暂停

您可以要求我们重审这一决定。有关更多信息，请参阅 Amazon SES 发送审核流程 FAQs。

当你实施了你认为可以解决问题的变更后，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。包括您为解决此问题而采取的措施的详细信息，以及您确保此问题不会再次发生的计划的详细信息。收到您的请求后，我们会审核您提供的信息，并视需要更改您的账户状态。

反垃圾邮件组织通知

本节包含有关 Amazon SES 声誉指标页面中显示的反垃圾邮件组织通知的其他信息。

您为什么会收到此通知

一家声誉好的反垃圾邮件组织报告称，您的 Amazon SES 账户发送的一些内容已被其系统标记为未经请求的或有问题的内容。

我们无法提供与导致该反垃圾邮件组织将您的内容标记为有问题的内容的具体邮件相关的信息。我们无法提供发布报告的组织的名称。通常情况下，反垃圾邮件组织会综合考虑以下几个因素：收件人反馈、邮件参与度指标、尝试传送到无效地址的次数、被其垃圾邮件筛选器标记的内容和垃圾邮件陷阱命中数。这里并没有列出全部的因素，还有一些其他因素也可能导致这些组织标记您的内容。

解决问题的方法

要解决此问题，您需要确定您的电子邮件发送程序的哪些方面可能导致反垃圾邮件组织将您的电子邮件标记为有问题。然后，您需要更改您的发送程序来解决这些问题。

如果您的账户正在接受审核

审核期结束时，如果反垃圾邮件组织继续将从您的账户发送的电子邮件标识为问题邮件，我们会暂停账户发送电子邮件的功能，直到您解决问题。

如果您已经实施了您认为可以解决问题的更改，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供所做的更改的详细信息。收到此信息后，我们会延长审核期，以确保仅分析我们在您实施更改之后所收到的反垃圾邮件组织的通知。在此延长的审核期结束时，如果反垃圾邮件组织不再列出您的账户，我们将不再继续审核您的账户。

如果账户发送电子邮件的功能已暂停

您可以要求我们重审这一决定。有关更多信息，请参阅 Amazon SES 发送审核流程 FAQs。

当你实施了你认为可以解决问题的变更后，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。包括您为解决此问题而采取的措施的详细信息，以及您确保此问题不会再次发生的计划的详细信息。收到您的请求后，我们会审核您提供的信息，并视需要更改您的账户状态。

列表轰炸通知

本节包含有关 Amazon SES 声誉指标页面中显示的列表轰炸通知的其他信息。

您为什么会收到此通知

一家反垃圾邮件组织发现，您的电子邮件发送过程易受“列表轰炸”攻击。列表轰炸是一种滥用形式，其中攻击者在基于 Web 的表单上注册了大量的电子邮件地址。对于受影响的电子邮件服务的用户，列表轰炸可能会导致服务中断。它还可能导致您的电子邮件被电子邮件提供商阻止。

反垃圾邮件组织使用专有方法来识别易受列表轰炸的网站。出于这个原因，我们无法提供有关该问题的更多详细信息来说明为什么反垃圾邮件组织认为您的电子邮件发送过程存在问题。此外，我们无法分享识别该问题的组织的名称。

解决问题的方法

您应检查所有基于 Web 的注册表，确保它们不易受到这种滥用。每个表均应包含一个 CAPTCHA，防止自动化脚本提交订阅请求。此外，当新用户注册您的产品或服务时，向他们发送一封电子邮件来确认他们确实希望进行注册。除非客户明确选择接收您的通信邮件，否则不要向客户发送任何其他电子邮件。

最后，您应在电子邮件列表上实施“许可通行证”。在许可通行证中，您向所有客户发送一封电子邮件，询问他们是否仍要接收您发送的电子邮件。只向确认要继续接收您发送的电子邮件的客户发送电子邮件。

如果您的账户正在接受审核

审核期结束时，如果反垃圾邮件组织继续将从您的账户发送的电子邮件标识为问题邮件，我们会暂停账户发送电子邮件的功能，直到您解决问题。

如果您已经实施了您认为可以解决问题的更改，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供所做的更改的详细信息。收到此信息后，我们会延长审核期，以确保仅分析我们在您实施更改之后所收到的反垃圾邮件组织的通知。在此延长的审核期结束时，如果反垃圾邮件组织不再列出您的账户，我们将不再继续审核您的账户。

如果账户发送电子邮件的功能已暂停

您可以要求我们重审这一决定。有关更多信息，请参阅 Amazon SES 发送审核流程 FAQs。

当你实施了你认为可以解决问题的变更后，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。包括您为解决此问题而采取的措施的详细信息，以及您确保此问题不会再次发生的计划的详细信息。收到您的请求后，我们会审核您提供的信息，并视需要更改您的账户状态。

直接反馈通知

本节包含有关 Amazon SES 声誉指标页面中显示的直接反馈通知的其他信息。

您为什么会收到此通知

大量用户直接联系 Amazon SES 以报告他们从与您的 Amazon SES 账户关联的地址或域收到的邮件。这种类型的反馈不会出现在邮箱提供商直接报告的投诉中，并且不会包括在声誉指标页面上显示的退回邮件和投诉指标中。

为了保护报告这些问题的用户的隐私，我们不能提供他们的电子邮件地址。

收件人可能会在以下情况下向 Amazon SES 投诉：收到自己未注册接收的邮件；没有收到应该收到的邮件类型；认为收到的电子邮件没有用或其不感兴趣；发现收到的邮件不是自己注册的邮件；或收到太多邮件。这里并没有列出全部的情况；您的案例中相关的因素取决于您的特定电子邮件发送程序。

解决问题的方法

我们建议您实施双向确认策略 (如构建和维护列表中所述) 来获取新地址，并建议您仅将电子邮件发送到完成双向确认流程的地址。

此外，您应清理最近未与您的电子邮件互动的地址列表。您可以使用打开情况和单击情况跟踪 (如监控您的 Amazon SES 发送活动中所述)，以确定哪些用户正在查看您发送的内容并与之互动。

如果您的账户正在接受审核

审核期结束时，如果 Amazon SES 继续收到大量关于从您的账户发送的邮件的直接投诉，我们会暂停您账户发送电子邮件的功能，直到您解决问题。

如果您已经实施了您认为可以解决问题的更改，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。提供有关您为解决问题所采取的步骤的详细信息，并说明这些步骤将如何防止日后再次出现相同的问题。如果我们认同您所做的更改能够妥善解决问题，我们会取消账户的审核期。

如果账户发送电子邮件的功能已暂停

您可以要求我们重审这一决定。有关更多信息，请参阅 Amazon SES 发送审核流程 FAQs。

当你实施了你认为可以解决问题的变更后，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。包括您为解决此问题而采取的措施的详细信息，以及您确保此问题不会再次发生的计划的详细信息。收到您的请求后，我们会审核您提供的信息，并视需要更改您的账户状态。

域阻止列表通知

本节包含有关 Amazon SES 声誉指标页面中显示的域阻止列表通知的其他信息。

您为什么会收到此通知

从您的 Amazon SES 账户发送的电子邮件包含对已在声誉好的域阻止列表上列出的域的引用。这些列表上的域通常与攻击性或恶意行为关联。相关域可能是也可能不是您从中发送电子邮件的域。包含对阻止列表上的域的引用或链接的邮件或者包含在此类域上托管的图像的邮件也可能会被标记。

我们无法提供导致您的邮件被标记的域的名称，也无法确定被以这种方式标记的电子邮件。

解决问题的方法

首先，创建您通过 Amazon SES 发送的电子邮件中引用的所有域名的列表。接下来，使用 Spamhaus 域查找工具来确定您的电子邮件中的哪些域包含在域阻止列表中。您发送的电子邮件中可能有多个引用的域在阻止列表上。

Spamhaus 域名屏蔽名单不隶属于亚马逊 SES 或. AWS我们不保证此域列表的准确性。Spamhaus 域阻止列表和域查找工具由 Spamhaus Project 拥有、运营和维护。

如果您的账户正在接受审核

我们会在您在审核期间发送的电子邮件中查找对用于恶意目的的域的引用。如果您的电子邮件仍包含大量对于这些域的引用，我们可能会暂停您的账户发送电子邮件的功能，直到您解决此问题为止。

如果您已经实施了您认为可以解决问题的更改，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供所做的更改的详细信息。收到此信息后，我们会延长审核期，以确保仅分析在您实施更改之后您电子邮件中存在的列入阻止列表的域的数量。此延长的审核期结束时，如果域阻止列表通知的数量减少或消除，并且我们认为您已采取措施防止问题再次发生，我们将取消您账户的审核期。

如果账户发送电子邮件的功能已暂停

您可以要求我们重审这一决定。有关更多信息，请参阅 Amazon SES 发送审核流程 FAQs。

当你实施了你认为可以解决问题的变更后，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。包括您为解决此问题而采取的措施的详细信息，以及您确保此问题不会再次发生的计划的详细信息。收到您的请求后，我们会审核您提供的信息，并视需要更改您的账户状态。

内部审查通知

本节包含有关 Amazon SES 声誉指标页面中显示的内部审核通知的其他信息。

您为什么会收到此通知

通过对您的账户的全面审查，发现了可能导致邮箱提供商或收件人将您的邮件识别为垃圾邮件的几个特征。

为了保护我们的滥用检测流程，我们不能显示导致您的账户以这种方式被标记的具体因素。

可能导致此决定的常见因素包括以下方面：

- 商业反垃圾邮件系统标记的邮件。
- 暗示收件人未明确请求电子邮件的邮件内容。
- 消息发件人和电子邮件正文中的品牌之间的不匹配。
- 发件人不清楚的内容。
- 发送用于处理与未经请求的电子邮件关联的内容的消息。
- 与未经请求的电子邮件关联的格式化模式。
- 从声誉不佳的域发送或对声誉不佳的域进行引用。

这个列表并不全面。此通知的具体原因可能是任何这些因素的组合，也可能是没有列出的原因。

解决问题的方法

以下建议可能有助于降低问题的严重性：

- 确保仅您正在联系的收件人是明确要求接收您发送的电子邮件的收件人。
- 请勿购买、出租或借用电子邮件收件人列表。
- 不要试图隐藏您的身份或您发送邮件的沟通目的。
- 创建您通过 Amazon SES 发送的电子邮件中引用的所有域的列表，然后使用 Spamhaus Domain Lookup 工具（https://www.spamhaus.org/lookup/）确定这些域中是否有任何域在 Spamhaus 域阻止列表中。
- 确保您在设计电子邮件时遵守行业最佳实践。

此处并没有列出全部方法，但它应该可以帮助您确定可能导致电子邮件被标记的一些最常见的因素。

Spamhaus 域名屏蔽名单不隶属于亚马逊 SES 或. AWS我们不保证此域列表的准确性。Spamhaus 域阻止列表和域查找工具由 Spamhaus Project 拥有、运营和维护。

如果账户正在接受审核，或者账户发送电子邮件的功能已暂停

当你实施了你认为可以解决问题的变更后，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。提供有关您为解决问题所采取的步骤的详细信息，并说明这些步骤将如何防止日后再次出现相同的问题。如果我们认同您所做的更改能够妥善解决问题，我们将取消您账户的审核期或恢复账户发送电子邮件的功能。

如果在解除对您的账户的审核或恢复了您账户的发送功能后，我们又发现同样的问题，则可能再次审核您的账户或暂停发送电子邮件的功能。在极端情况下，或者如果我们发现同一问题反复出现，我们会永久暂停账户发送电子邮件的功能。

有关当账户接受审核或被暂停发送电子邮件的功能时该做些什么的更多信息，请参阅Amazon SES 发送审核流程 FAQs。

邮箱提供商通知

本节包含有关 Amazon SES 声誉指标页面中显示的邮箱提供商通知的其他信息。

您为什么会收到此通知

某个主流邮箱提供商向我们报告称，与您的 Amazon SES 账户关联的地址或域正在发送未经请求的或恶意的电子邮件。

我们无法分享发布此报告的组织的身份。此外，我们没有有关导致邮箱提供商发布报告的具体原因的信息。通常，邮箱提供商会基于客户反馈、客户参与度指标、尝试传送到无效地址的次数和垃圾邮件筛选器标记的内容进行此类决定。这里并没有列出全部的因素，可能还有一些导致邮箱提供商标记您的内容的其他因素。

解决问题的方法

要解决此问题，您需要确定您的电子邮件发送程序的哪些方面可能导致了邮箱提供商将您的邮件标记为有问题。然后，您必须更改您的发送程序来解决这些问题。

如果您的账户正在接受审核

审核期结束时，如果邮箱提供商继续将从您的账户发送的电子邮件标识为问题邮件，我们会暂停账户发送电子邮件的功能，直到您解决问题。

如果您已经实施了您认为可以解决问题的更改，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供所做的更改的详细信息。收到此信息后，我们会延长审核期，以确保仅分析我们在您实施更改之后所收到的邮箱提供商通知的数量。在此延长的审核期结束时，如果邮箱提供商不再将您的账户报告为有问题，则我们不会继续审核您的账户。

如果账户发送电子邮件的功能已暂停

您可以要求我们重审这一决定。有关更多信息，请参阅 Amazon SES 发送审核流程 FAQs。

当你实施了你认为可以解决问题的变更后，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。包括您为解决此问题而采取的措施的详细信息，以及您确保此问题不会再次发生的计划的详细信息。收到您的请求后，我们会审核您提供的信息，并视需要更改您的账户状态。

收件人反馈通知

本节包含有关 Amazon SES 声誉指标页面中显示的收件人反馈通知的其他信息。

您为什么会收到此通知

某个主流邮箱提供商向我们报告称，他们有大量用户报告称从您的 Amazon SES 账户发送的邮件是未经请求的。这种类型的反馈不会出现在邮箱提供商直接报告的投诉中，也不会包含在 Amazon SES 退回邮件和投诉通知中。

大量投诉可能会对所有 Amazon SES 用户产生负面影响。为了保护您和其他 Amazon SES 客户的声誉，我们将会在某个账户收到一定数量的投诉时立即采取措施。

我们无法提供将您的电子邮件报告为未经请求的具体电子邮件地址的列表。此外，我们也无法分享向我们报告此问题的邮箱提供商的名称。

解决问题的方法

要解决此问题，您需要确定您的电子邮件发送程序的哪些方面可能导致收件人针对他们从您这里收到的电子邮件发出投诉。确定这些因素后，请更改您的电子邮件发送实践以更正这些因素。

为了获取新的地址，我们建议您实施双向确认战略，如构建和维护列表中所述。我们建议仅向完成了双向确认流程的地址发送电子邮件。

此外，您应清理最近未与您的电子邮件互动的地址列表。您可以使用打开情况和单击情况跟踪 (如监控您的 Amazon SES 发送活动中所述)，以确定哪些用户正在查看您发送的内容并与之互动。

如果您的账户正在接受审核

审核期结束时，如果邮箱提供商继续收到大量投诉举报，我们会暂停账户发送电子邮件的功能，直到您解决问题。

如果您已经实施了您认为可以解决问题的更改，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供所做的更改的详细信息。收到此信息后，我们会延长审核期，以确保仅分析我们在您实施更改之后所收到的邮箱提供商投诉的数量。在此延长的审核期结束时，如果邮箱提供商投诉的数量已减少或消除，则我们不会继续审核您的账户。

如果账户发送电子邮件的功能已暂停

您可以要求我们重审这一决定。有关更多信息，请参阅 Amazon SES 发送审核流程 FAQs。

当你实施了你认为可以解决问题的变更后，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。包括您为解决此问题而采取的措施的详细信息，以及您确保此问题不会再次发生的计划的详细信息。收到您的请求后，我们会审核您提供的信息，并视需要更改您的账户状态。

相关账户通知

本节包含有关 Amazon SES 声誉指标页面中显示的相关账户通知的其他信息。

您为什么会收到此通知

我们检测到与其他 Amazon SES 账户发送的电子邮件相关的严重问题。我们认为有问题的账户与您的账户有关 AWS 账户，因此我们已采取措施避免出现类似问题。

解决问题的方法

当我们暂停账户发送电子邮件的功能时，我们始终会将暂停发送功能原因的相关信息发送给此账户的所有者。有关更多信息，请参阅我们发送给相关账户所有者的电子邮件。

您应首先解决相关账户的问题。实施您认为可以解决问题的更改后，登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。提供有关您为解决问题所采取的步骤的详细信息，并说明这些步骤将如何防止日后再次出现相同的问题。如果我们认同您所做的更改能够妥善解决问题，我们将取消您账户的审核期或恢复账户发送电子邮件的功能。

垃圾邮件陷阱通知

本节包含有关 Amazon SES 声誉指标页面中显示的垃圾邮件陷阱通知的其他信息。

您为什么会收到此通知

某个第三方反垃圾邮件组织向我们报告称，他们的垃圾邮件陷阱地址最近收到了来自与您的 Amazon SES 账户关联的已验证地址或域的电子邮件。

垃圾邮件陷阱是一个休眠电子邮件地址，专门用于引诱未经请求的电子邮件 (垃圾邮件)。大量垃圾邮件陷阱报告可能会对所有 Amazon SES 用户产生负面影响。为了保护您和其他 Amazon SES 客户的声誉，我们将会在某个账户向垃圾邮件陷阱地址发送特定量的电子邮件后立即采取措施。

解决问题的方法

我们无法披露与您遇到的垃圾邮件陷阱关联的电子邮件地址。这些地址由拥有它们的组织严密保护，一旦这些地址为众人所知，它们将毫无价值。

将电子邮件发送到垃圾邮件陷阱地址通常表示，您获取客户的电子邮件地址的方式有问题。例如，购买的电子邮件地址列表可能包含垃圾邮件陷阱地址，这正是 Amazon SES 服务条款禁止发送到购买的列表或租赁的列表的原因。为了获取新的地址，我们建议您实施双向确认战略，如构建和维护列表中所述。我们建议仅向完成了双向确认流程的地址发送电子邮件。

此外，您应清理最近未与您的电子邮件互动的地址列表。您可以使用打开情况和单击情况跟踪 (如监控您的 Amazon SES 发送活动中所述)，以确定哪些用户正在查看您发送的内容并与之互动。

如果您的账户正在接受审核

审核期结束时，如果仍有邮件从您的账户发送到垃圾邮件陷阱地址，我们会暂停账户发送电子邮件的功能，直到您解决问题。

如果您已经实施了您认为可以解决问题的更改，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供所做的更改的详细信息。收到此信息后，我们会延长审核期，以确保仅分析我们在您实施更改之后所收到的垃圾邮件陷阱报告的数量。在此延长的审核期结束时，如果垃圾邮件陷阱报告的数量已减少或消除，则我们不会继续审核您的账户。

如果账户发送电子邮件的功能已暂停

您可以要求我们重审这一决定。有关更多信息，请参阅 Amazon SES 发送审核流程 FAQs。

当你实施了你认为可以解决问题的变更后，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。包括您为解决此问题而采取的措施的详细信息，以及您确保此问题不会再次发生的计划的详细信息。收到您的请求后，我们会审核您提供的信息，并视需要更改您的账户状态。

脆弱站点通知

本节包含有关 Amazon SES 声誉指标页面中显示的易受攻击站点通知的其他信息。

您为什么会收到此通知

在一次全面审查中，我们发现从您的账户中发出了一些我们认为不是您有意要发送的邮件。这些邮件非常可能会被邮箱提供商和收件人标记为垃圾邮件。

这些情况中最常见的是第三方滥用您的网站的功能发送不需要的电子邮件。例如，如果您的网站包含“向好友发送电子邮件”、“联系我们”、“邀请好友”或类似功能，则第三方可能使用该功能发送未经请求的电子邮件。

解决问题的方法

首先，确定您的网站或应用程序的哪些功能可能允许第三方在您不知道的情况下使用 Amazon SES 发送电子邮件。在您的支持中心案例中，您可以请求获得我们认为以这种方式发送的邮件的示例。

接下来，修改您的应用程序或网站，以防止未经请求的发送。例如，添加 CAPTCHA、限制电子邮件发送的速率、移除用户提交自定义内容的功能、要求用户登录以发送电子邮件以及移除应用程序同时生成多个通知的功能。

如果账户正在接受审核，或者账户发送电子邮件的功能已暂停

当你实施了你认为可以解决问题的变更后，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。包括您为解决此问题而采取的措施的详细信息，以及您确保此问题不会再次发生的计划的详细信息。收到您的请求后，我们会审核您提供的信息，并视需要更改您的账户状态。

如果在解除对您账户的审核或恢复了您账户的发送功能后，我们又发现了同样的问题，则可能会再次审核您的账户或暂停您账户的电子邮件发送功能。如果发现极端的问题，或者发现同一问题反复出现，我们可能会永久暂停您账户发送电子邮件的功能。

有关当账户接受审核或被暂停发送电子邮件的功能时该做些什么的更多信息，请参阅Amazon SES 发送审核流程 FAQs。

凭证泄露通知

本节包含有关 Amazon SES 声誉指标页面中显示的凭证泄露站点通知的其他信息。

您为什么会收到此通知

在一次全面审查中，我们发现从您的账户中发出了一些我们认为不是您有意要发送的邮件。这些邮件非常可能会被邮箱提供商和收件人标记为垃圾邮件。

一些常见的原因包括 IAM 访问密钥泄露、SMTP 密码泄露或其他安全漏洞。

解决问题的方法

您应该对 SES 利用机制进行全面的安全审查。确保已轮换任何适用的密码或 SMTP 密码，并且已从账户中删除任何未经授权的用户或资源。请勿在第三方网站上或存储库中存储密码或访问密钥等敏感信息。现在建议您不要为用户使用 IAM 访问密钥，也绝不要为根用户使用 IAM 访问密钥。如果您仍在使用此类密钥，则应将它们迁移到提供临时凭证的机制，例如在 AWS IAM Identity Center中创建用户。

如果账户正在接受审核，或者账户发送电子邮件的功能已暂停

当你实施了你认为可以解决问题的变更后，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。包括您为解决此问题而采取的措施的详细信息，以及您确保此问题不会再次发生的计划的详细信息。收到您的请求后，我们会审核您提供的信息，并视需要更改您的账户状态。

如果在解除对您账户的审核或恢复了您账户的发送功能后，我们又发现了同样的问题，则可能会再次审核您的账户或暂停您账户的电子邮件发送功能。如果发现极端的问题，或者发现同一问题反复出现，我们可能会永久暂停您账户发送电子邮件的功能。

有关当账户接受审核或被暂停发送电子邮件的功能时该做些什么的更多信息，请参阅Amazon SES 发送审核流程 FAQs。

其他通知

本节包含有关 Amazon SES 声誉指标页面中显示的其他通知的其他信息。

您为什么会收到此通知

某次自动或人工审查发现了本文档前面几个部分中未列出的问题。

解决问题的方法

有关特定问题的详细信息，请参阅我们代表您开立的支持中心案例。要访问支持中心，请登录 AWS Management Console 并选择支持中心。在您对案例的回复中，描述您实施的更改。根据具体情况和所发现问题的性质，我们可能会结束审核期或恢复账户发送电子邮件的功能。

使用创建信誉监控警报 CloudWatch

Amazon SES 会自动向亚马逊发布一系列与声誉相关的指标。 CloudWatch您可以使用这些指标来创建警报，以便在您的退回邮件率或投诉率达到可能影响您账户发送电子邮件的能力的级别时通知您。

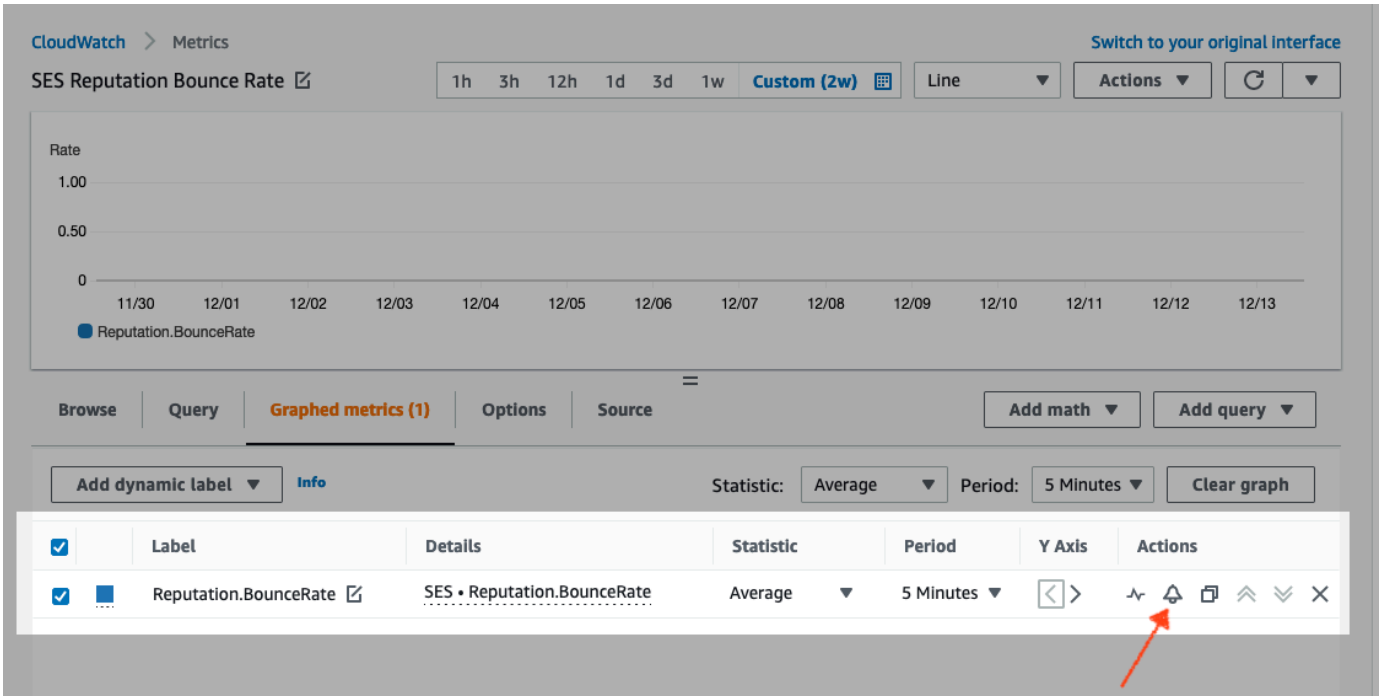
> **Note**
>
> 本节中的 CloudWatch 部分步骤仅介绍设置 CloudWatch 警报以监控您的 SES 发件人信誉的核心步骤。他们不探索有关 CloudWatch 警报可选设置的高级配置。有关配置 CloudWatch 警报的完整信息，请参阅亚马逊 CloudWatch 用户指南中的使用亚马逊 CloudWatch 警报。

先决条件

- 创建一个 Amazon SNS 主题，然后使用您的首选终端节点（如电子邮件或 SMS）订阅该主题。有关更多信息，请参阅 Amazon Simple Notification Service Developer Guide 中的创建 Amazon SNS 主题和订阅 Amazon SNS 主题。
- 如果您从未在当前区域中发送过电子邮件，则可能不会看到 SES 命名空间。要确保您有指标组，请发送测试电子邮件至 mailbox simulator（邮箱模拟器）。

创建 CloudWatch 警报以监控发送信誉

1. 登录 AWS Management Console 并打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在屏幕左侧的导航窗格中，选择声誉指标。
3. 在 “账户级别” 选项卡下的 “信誉指标” 页面上，在 “退回率” 或 “投诉率” 窗格中，选择 “查看范围”，这将打开包含您所选指标的 CloudWatch 控制台。 CloudWatch
4. 在 “图表化指标” 选项卡下，在您选择的指标行上，在本例中为 “声誉”。 BounceRate，选择 “操作” 列中的警钟图标（见下图）-这将打开 “指定指标和条件” 页面。



5. 滚动到 Conditions（条件）窗格，然后在Threshold type（阈值类型）窗格中选择 Static（静态）。

   a. 在 When ***metric***ever is... 字段中，选择 “大于/ 等于”。

   b. 在... 字段中，指定应引起 CloudWatch 警报的值。

      - 如果您要创建警报来监控退回邮件率，请注意，Amazon SES 建议您将退回邮件率保持在 5% 以下。如果您的账户的邮件退回率高于 10%，我们可能会暂停您的账户发送电子邮件的功能。因此，您应配置 CloudWatch 为在账户的跳出率大于或等于 0.05 (5%) 时向您发送通知。
      - 如果您要创建警报来监控投诉率，请注意，Amazon SES 建议您将投诉率保持在 0.1% 以下。如果您的账户的投诉率高于 0.5%，我们可能会暂停您的账户发送电子邮件的功能。因此，您应配置 CloudWatch 为在您账户的投诉率大于或等于 0.001 (0.1%) 时向您发送通知。

   c. 展开 Additional configuration（其他配置），并在 Missing data treatment（缺少数据处理）字段中选择 Treat missing data as ignore (maintain the alarm state)（将缺少数据视为忽略(保持告警状态)）。

   d. 选择下一步。

6. 在 Configure actions（配置操作）窗格上，在 Alarm state trigger（告警状态触发器）中选择 In Alarm（告警中）。

a. 在 Select an SNS topic（选择 SNS 主题）字段中选择 Select an existing SNS topic（选择现有的 SNS 主题）。
b. 在 Send notification to（将通知发送到）搜索框中，选择您在先决条件中创建和订阅的主题。
c. 选择下一步。

7. 在 Add a description（添加描述）下，输入告警的名称和描述，然后选择 Next（下一步）。
8. 在 Preview and create（预览和创建）窗格中，确认您的设置，如果满意，则选择 Create alarm（创建告警）。如果您想要更改某些内容，请在您想要返回和编辑的每个部分中选择 Previous（上一步）按钮。

专用 SNDS 指标 IPs

您可以在每个使用 Amazon SES AWS 区域 的地方查看租用的专用 IP 地址的智能网络数据服务 (SNDS) 数据。此 SNDS 数据可通过 Amazon CloudWatch 控制台获取。

SNDS 是一个 Outlook 程序，它允许 IP 拥有者帮助防止其 IP 空间内出现垃圾邮件。Amazon SES 为租赁专用的用户提供了这些重要数据 IPs。SNDS 数据提供了对 IP 邮件发送行为的洞察，并指出了您的发件人声誉所关注的领域。

> **Note**
>
> 当引用 Outlook 时，这涵盖了他们跟踪的所有域。例如，这可以涵盖 Hotmail.com、Outlook.com 和 Live.com。

查看专用 IP 地址的 SNDS 数据

1. 登录 Amazon CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，展开 Metrics（指标），然后选择 All metrics（所有指标）。

   （提供了新 CloudWatch 控制台界面的说明。）
3. 在 “指标” 容器的 “浏览” 选项卡下，选择您的 AWS 区域，然后选择 SES。
4. 选择 IP 指标，它将向您显示 SNDS IPs 追踪的所有专用数据。

   （注意：如果在所选地区没有与您的账户关联的专用 IP 地址，IP 指标将不会显示在 CloudWatch 控制台中。）
5. 在此列表中查看 SNDS IPs 追踪的所有专用 IP 地址，或者选择单个 IP 地址仅查看其指标。

针对每个专用 IP 地址提供了以下指标，并由 Outlook 定义。有关更多信息，请参阅 Outlook 的 SNDS。FAQs

> **Note**
>
> 这些指标表示每天提供一次更新数据的活动周期。这些指标还具有相应的时间戳，反映 24 小时的时间段。

- SNDS。 RCPTCommands-这是 SNDS 在活动期间感知到的针对特定 IP 地址的 RCPT 命令的数量。RCPT 命令是用于发送邮件的 SMTP 协议的一部分，该协议指定您尝试向其发送电子邮件的收件人地址。
- SNDS。 DATACommands-活动期间，SNDS 感知到的针对特定 IP 地址的 DATA 命令的数量。DATA 命令是用于发送邮件的 SMTP 协议的一部分，特别是实际将邮件传输到先前建立的预期收件人的部分。
- SNDS。 MessageRecipients-活动期间内 SNDS 感知到的针对特定 IP 地址的邮件的收件人数。
- SNDS。 SpamRate-显示在给定活动期间应用于该 IP 地址发送的所有邮件的垃圾邮件筛选的汇总结果。
  - A SpamRate 为 0 表示 IP 地址的垃圾邮件少于 10%。
  - A SpamRate 为 0.5 表示从 IP 地址生成了 10% 到 90% 的垃圾邮件。
  - A SpamRate 为 1 表示 90% 或更多的垃圾邮件来自该 IP 地址。
- SNDS。 ComplaintRate-这是 Outlook 用户在活动期间投诉从 IP 收到的邮件的时间的一小部分。
  - A ComplaintRate 为 1 表示投诉率为 100%。
  - 例如 ComplaintRate ，0.05表示投诉率为5%。
  - A ComplaintRate 为 0 表示该比率小于 0.1%。
- SNDS。 TrapHits-显示发送到 “陷阱账户” 的消息数量。陷阱账户是 Outlook 维护的不请求任何邮件的账户。因此，发送到陷阱账户的任何邮件都很可能是垃圾邮件。

问题排查

问题 1：为什么数据不是每天都填充？ 以下任一情况均适用：

- SNDS 数据取决于 Outlook 的 SNDS 计划。
- SNDS 需要接收才能计算值的最低电子邮件阈值。当某个 IP 的电子邮件量较低时，数据可能不可用。

问题 2：为什么是 SNDS。 SpamRate 和 SNDS。 ComplaintRate 指标发生变化，如果费率变为值为 1 我该怎么办？

这表明您发送行为中的某些内容触发了 Outlook SNDS 计划的负面响应。在这种情况下，您需要查看其他互联网服务提供商 (ISPs) 以及您的参与人数，以确保这不是全球性问题。如果是全球性问题，您可能会看到多个问题 ISPs，这表明存在列表、内容、分发或权限问题。如果问题特定于 Outlook，请参阅如何以最佳方式交付给 Outlook。

问题 3：如果我的 SNDS 会 AWS 支持 采取什么行动。 SpamRate 从 0（或 0.5）的值变为 1？

AWS 对 SNDS 没有任何控制权，因此对 SNDS 没有影响力。所有缓解请求都需要通过新建支持申请表单直接向 Outlook 提出。

自动暂停电子邮件发送

为了保护您的发件人声誉，您可以暂时暂停发送使用特定配置集发送的邮件，或者暂停从特定 AWS 区域的 Amazon SES 账户发送的所有邮件的电子邮件发送。

通过使用 Amazon CloudWatch 和 Lambda，您可以创建一种解决方案，当您的声誉指标（例如退回率或投诉率）超过特定阈值时，该解决方案会自动暂停电子邮件发送。本主题包含设置此解决方案的过程。

本节中的主题：

- 自动为您的整个 Amazon SES 账户暂停电子邮件发送
- 自动暂停配置集电子邮件发送

自动为您的整个 Amazon SES 账户暂停电子邮件发送

本节中的程序说明了在单个 AWS 地区设置亚马逊 SES、Amazon SNS CloudWatch、Amazon 以及 AWS Lambda 自动暂停发送电子邮件的步骤。如果您从多个区域发送电子邮件，请为要在其中实施此解决方案的每个区域重复执行本节中的过程。

本节中的主题：

- 第 1 部分：创建 IAM 角色
- 第 2 部分：创建 Lambda 函数
- 第 3 部分：为您的账户重新启用电子邮件发送
- 第 4 部分：创建 Amazon SNS 主题和订阅
- 第 5 部分：创建 CloudWatch 警报

- 第 6 部分：测试解决方案

第 1 部分：创建 IAM 角色

配置自动暂停电子邮件发送的第一步是创建可执行 `UpdateAccountSendingEnabled` API 操作的 IAM 角色。

创建 IAM 角色

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择角色。
3. 选择 Create role（创建角色）。
4. 在选择可信实体页面中，对于可信实体类型选择 AWS 服务。
5. 在 Use case（使用案例）下，选择 Lambda，然后选择 Next（下一步）。
6. 在 Add permissions（添加权限）页面上，选择以下策略：

   - AWSLambdaBasicExecutionRole
   - 亚马逊SESFull访问权限

   > **Tip**
   >
   > 使用 Permission policies（权限策略）下的搜索框可快速找到这些策略，但请注意，在搜索和选择第一个策略后，必须先选择 Clear filters（清除筛选条件），然后再搜索和选择第二个策略。

   然后选择下一步。
7. 在 Name, review, and create（命名、检查和创建）页面的 Role details（角色详细信息）下，在 Role name（角色名称）字段中为策略输入有意义的名称。
8. 验证您选择的两个策略是否列在 Permissions policy summary（权限策略摘要）表中，然后选择 Create role（创建角色）。

第 2 部分：创建 Lambda 函数

在创建 IAM 角色之后，便可以创建暂停您账户的电子邮件发送的 Lambda 函数。

创建 Lambda 函数

1. 打开 AWS Lambda 控制台，网址为https://console.aws.amazon.com/lambda/。
2. 使用区域选择器来选择要在其中部署此 Lambda 函数的区域。

> **Note**
>
> 此功能仅暂停在此步骤中选择的 AWS 区域发送电子邮件。如果您从多个区域发送电子邮件，请为要在其中自动暂停电子邮件发送的每个区域重复执行本节中的过程。

3. 选择 Create function (创建函数)。
4. 在 Create function（创建函数）下，选择 Author from scratch（从头开始创作）。
5. 在 Basic information（基本信息）下，完成以下步骤：

   - 对于 Function name（函数名称），键入 Lambda 函数的名称。
   - 对于运行时系统，选择 Node.js 18x（或者选择列表中当前提供的版本）。
   - 对于 Architecture（架构），保持预先选择的原定设置值 x86_64。
   - 在 Permissions（权限）下，展开 Change default execution role（更改原定设置执行角色），然后选择 Use an existing role（使用现有角色）。
   - 在 Existing role（现有角色）列表框内单击，然后选择您在the section called “第 1 部分：创建 IAM 角色”中创建的 IAM 角色。

   然后，选择 Create function（创建函数）。
6. 在 Code source（代码源）下，在代码编辑器中粘贴以下代码：

```
'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);
        }
    });
};
```

然后选择部署。

7. 选择测试。如果 Configure test event（配置测试事件）窗口出现，在 Event name（事件名称）字段中键入一个名称，然后选择 Save（保存）。
8. 展开 Test（测试）下拉框并选择刚刚创建的事件的名称，然后选择 Test（测试）。
9. 将出现 Execution results（执行结果）选项卡 - 就在其下方和右侧，确保显示了 `Status: Succeeded`。如果函数执行失败，请执行以下操作：
   - 确认您在the section called “第 1 部分：创建 IAM 角色”中创建的 IAM 角色包含正确的策略。
   - 确认 Lambda 函数中的代码不包含任何错误。Lambda 代码编辑器会自动突出显示语法错误和其他潜在问题。

第 3 部分：为您的账户重新启用电子邮件发送

在the section called “第 2 部分：创建 Lambda 函数”中测试 Lambda 函数的一个副作用是会为您的 Amazon SES 账户暂停电子邮件发送。在大多数情况下，您不想在 CloudWatch 警报触发之前暂停账户的发送。

本节中的过程可为您的 Amazon SES 账户重新启用电子邮件发送。要完成这些过程，您必须安装并配置 AWS Command Line Interface。有关更多信息，请参阅 用户指南。AWS Command Line Interface

重新启用电子邮件发送

1. 在命令行键入以下命令，为账户重新启用电子邮件发送。*`sending_region`*替换为您要重新启用电子邮件发送功能的区域名称。

```
aws ses update-account-sending-enabled --enabled --region sending_region
```

2. 在命令行键入以下命令，查看账户的电子邮件发送状态：

```
aws ses get-account-sending-enabled --region sending_region
```

如果您看到以下输出，说明已成功为您的账户重新启用了电子邮件发送：

```
{
    "Enabled": true
}
```

第 4 部分：创建 Amazon SNS 主题和订阅

CloudWatch 要在触发警报时执行您的 Lambda 函数，您必须先创建一个 Amazon SNS 主题并订阅该主题的 Lambda 函数。

要创建 Amazon SNS 主题并让 Lambda 函数订阅该主题，请执行以下操作：

1. 在 v3/home 上打开亚马逊 SNS 控制台。https://console.aws.amazon.com/sns/
2. 按照《Amazon Simple Notification Service 开发人员指南》中的步骤创建主题。
   - Type（类型）必须是 Standard（标准）（而不是 FIFO）。
3. 按照《Amazon Simple Notification Service 开发人员指南》中的步骤订阅主题。
   a. 对于协议，选择 AWS Lambda。
   b. 对于 Endpoint (端点)，选择您在the section called “第 2 部分：创建 Lambda 函数”中创建的 Lambda 函数。

第 5 部分：创建 CloudWatch 警报

本节包含在中创建警报的过程 CloudWatch ，该警报将在指标达到特定阈值时触发。警报触发后会向在the section called “第 4 部分：创建 Amazon SNS 主题和订阅”中创建的 Amazon SNS 主题发送通知，然后该主题执行在the section called “第 2 部分：创建 Lambda 函数”中创建的 Lambda 函数。

创建 CloudWatch 警报

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 使用区域选择器来选择要在其中自动暂停电子邮件发送的区域。
3. 在导航窗格中，选择 Alarms (告警)。
4. 选择创建告警。
5. 在 Create Alarm（创建告警）窗口上，在 SES Metrics（SES 指标）下选择 Account Metrics（账户指标）。
6. 在 Metric Name（指标名称）下，选择以下选项之一：

   - 声誉。 BounceRate— 如果您想在账户的整体硬退回率超过您定义的阈值时暂停账户电子邮件发送，请选择此指标。
   - 声誉。 ComplaintRate— 如果您想在账户的总体投诉率超过您定义的阈值时暂停向账户发送电子邮件，请选择此指标。

   选择下一步。
7. 完成以下步骤：

   - 在 Alarm Threshold（告警阈值）下，对于 Name（名称），键入警报的名称。
   - 在“随时：声誉”之下。 BounceRate或随时：声誉。 ComplaintRate，指定导致警报触发的阈值。

     > **Note**
     >
     > 如果退信率超过 5% 或投诉率超过 0.1%，账户会自动置于审核状态。当您指定触发 CloudWatch 警报的退回率或投诉率时，我们建议您使用低于这些比率的值，以防止您的账户受到审查。

   - 在 Actions（操作）下，为 Whenever this alarm（每当此告警）选择 State is ALARM（状态为“告警”）。对于 Send notification to (发送通知到)，选择您在the section called “第 4 部分：创建 Amazon SNS 主题和订阅”中创建的 Amazon SNS 主题。

   选择创建警报。

第 6 部分：测试解决方案

您现在可以测试警报以确保它在进入 ALARM 状态时执行 Lambda 函数。您可以使用 SetAlarmState API 操作临时更改警报状态。

这一部分中的过程是可选操作，但我们建议您完成这些过程以确保整个解决方案配置正确。

1. 在命令行键入以下命令，查看账户的电子邮件发送状态。*region*替换为区域名称。

   ```
   aws ses get-account-sending-enabled --region region
   ```

   如果为您的账户启用了发送功能，将显示以下输出：

   ```
   {
       "Enabled": true
   }
   ```

2. 在命令行键入以下命令，将警报状态临时更改为 ALARM：aws cloudwatch set-alarm-state --alarm-name ***MyAlarm*** --state-value ALARM --state-reason "Testing execution of Lambda function" --region ***region***

   将前面的命令替换*MyAlarm*为您在中创建的警报的名称the section called “第 5 部分：创建 CloudWatch 警报”，并*region*替换为要自动暂停电子邮件发送的区域。

   > **Note**
   >
   > 当您执行此命令时，警报状态会从 OK 切换为 ALARM 并在几秒后切换回 OK。您可以在 CloudWatch 控制台中警报的 “历史记录” 选项卡上或使用DescribeAlarmHistory操作来查看这些状态更改。

3. 在命令行键入以下命令，查看账户的电子邮件发送状态。

   ```
   aws ses get-account-sending-enabled --region region
   ```

   如果 Lambda 函数执行成功，您会看到以下输出：

   ```
   {
       "Enabled": false
   }
   ```

4. 完成the section called “第 3 部分：为您的账户重新启用电子邮件发送”中的步骤以为您的账户重新启用电子邮件发送。

自动暂停配置集电子邮件发送

您可以将 Amazon SES 配置为导出特定于使用设置为 Amazon 的特定配置发送的电子邮件的信誉指标 CloudWatch。然后，您可以使用这些指标来创建特定于这些配置集的 CloudWatch 警报。当这些警报超出特定阈值时，您可以自动暂停使用指定配置集的电子邮件发送，而不会影响您的 Amazon SES 账户的整体电子邮件发送功能。

> **Note**
>
> 本节中描述的解决方案暂停发送单个 AWS 区域中特定配置集的电子邮件。如果您从多个区域发送电子邮件，请为要在其中实施此解决方案的每个区域重复执行本节中的过程。

本节中的主题：

- 第 1 部分：启用配置集声誉指标的导出
- 第 2 部分：创建 IAM 角色
- 第 3 部分：创建 Lambda 函数
- 第 4 部分：重新启用配置集的电子邮件发送
- 第 5 部分：创建 Amazon SNS 主题
- 第 6 部分：创建 CloudWatch 警报
- 第 7 部分：测试解决方案

第 1 部分：启用配置集声誉指标的导出

您必须首先启用配置集的声誉指标的导出，然后才能配置 Amazon SES 以自动暂停该配置集的电子邮件发送。

要启用配置集的退回邮件和投诉指标的导出，请完成the section called “查看和导出声誉指标”中的步骤。

第 2 部分：创建 IAM 角色

配置自动暂停电子邮件发送的第一步是创建可执行 `UpdateConfigurationSetSendingEnabled` API 操作的 IAM 角色。

创建 IAM 角色

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择角色。
3. 选择 Create role（创建角色）。
4. 在 选择受信任实体的类型 下，选择 AWS 服务。
5. 在选择将使用此角色的服务下，选择 Lambda。选择 Next: Permissions（下一步: 权限）。
6. 在 Attach permissions policies（附加权限策略）页面上，选择以下策略：

   - AWS Lambda BasicExecutionRole
   - Amazon A SESFull cces s（我们建议您使用根据您的需求量身定制的自定义角色，包括呼叫权限`UpdateConfigurationSetSendingEnabled`。）

   > **Tip**
   >
   > 使用策略列表顶部的搜索框可快速找到这些策略。

   请选择 Next: Review（下一步：审核）。
7. 在 Review（审核）页面上，对于 Name（名称），为角色键入一个名称。选择 Create role（创建角色）。

第 3 部分：创建 Lambda 函数

在创建 IAM 角色之后，便可以创建可暂停配置集的电子邮件发送的 Lambda 函数。

创建 Lambda 函数

1. 打开 AWS Lambda 控制台，网址为https://console.aws.amazon.com/lambda/。
2. 使用区域选择器来选择要在其中部署此 Lambda 函数的区域。

> **Note**
>
> 此函数仅暂停您在此步骤中选择的 AWS 区域中的配置集的电子邮件发送。如果您从多个区域发送电子邮件，请为要在其中自动暂停电子邮件发送的每个区域重复执行本节中的过程。

3. 选择 Create function (创建函数)。
4. 在 Create function（创建函数）下，选择 Author from scratch（从头开始创作）。
5. 在 Author from scratch（从头开始创作）下，完成以下步骤：

   - 对于 Name（名称），键入 Lambda 函数的名称。
   - 对于 Runtime（运行时），选择 Node.js 14（或者选择列表中当前提供的版本）。
   - 对于 Role（角色），选择 Choose an existing role（选择现有角色）。
   - 对于 Existing role (现有角色)，选择您在the section called “第 2 部分：创建 IAM 角色”中创建的 IAM 角色。

   选择 Create function (创建函数)。
6. 在 Function code（函数代码）下，在代码编辑器中粘贴以下代码：

```
'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
            }),
        };
    }
};
```

将上述代码*ConfigSet*中的配置集替换为配置集的名称。选择保存。

7. 选择 Test（测试）。如果 Configure test event（配置测试事件）窗口出现，在 Event name（事件名称）字段中键入一个名称，然后选择 Create（创建）。
8. 确保页面顶部的通知栏显示 `Execution result: succeeded`。如果函数执行失败，请执行以下操作：
   - 确认您在the section called “第 2 部分：创建 IAM 角色”中创建的 IAM 角色包含正确的策略。
   - 确认 Lambda 函数中的代码不包含任何错误。Lambda 代码编辑器会自动突出显示语法错误和其他潜在问题。

第 4 部分：重新启用配置集的电子邮件发送

在the section called “第 3 部分：创建 Lambda 函数”中测试 Lambda 函数的一个副作用是会暂停配置集的电子邮件发送。在大多数情况下，您不希望在触发 CloudWatch 警报之前暂停发送配置集。

这一部分中的过程可为您的配置集重新启用电子邮件发送。要完成这些过程，您必须安装并配置 AWS Command Line Interface。有关更多信息，请参阅 用户指南。AWS Command Line Interface

重新启用电子邮件发送

1. 在命令行键入以下命令，为配置集重新启用电子邮件发送：

```
aws ses update-configuration-set-sending-enabled \
--configuration-set-name ConfigSet \
--enabled
```

在前面的命令中，*ConfigSet*替换为要暂停发送电子邮件的配置集的名称。

2. 在命令行键入以下命令，确保启用了电子邮件发送：

```
aws ses describe-configuration-set \
--configuration-set-name ConfigSet \
--configuration-set-attribute-names reputationOptions
```

该命令生成类似于以下示例的输出：

```
{
    "ConfigurationSet": {
        "Name": "ConfigSet"
    },
    "ReputationOptions": {
        "ReputationMetricsEnabled": true,
        "SendingEnabled": true
    }
}
```

如果 SendingEnabled 的值为 true，说明该配置集的电子邮件发送已成功重新启用。

第 5 部分：创建 Amazon SNS 主题

CloudWatch 要在触发警报时执行 Lambda 函数，您必须先创建一个 Amazon SNS 主题并订阅 Lambda 函数。

创建 Amazon SNS 主题

1. 在 v3/home 上打开亚马逊 SNS 控制台。https://console.aws.amazon.com/sns/

2. 使用区域选择器来选择要在其中自动暂停电子邮件发送的区域。
3. 在导航窗格中，选择 Topics（主题）。
4. 选择 创建新主题。
5. 在 Create new topic（创建新主题）窗口中，对于 Topic name（主题名称），为主题键入一个名称。(可选) 在 Display name（显示名称）字段中键入一个更具描述性的名称。

   选择创建主题。
6. 在主题列表中，选中您在上一步中创建的主题旁的框。在 Actions（操作）菜单上，选择 Subscribe to topic（订阅主题）。
7. 在 Create subscription（创建订阅）窗口中，进行以下选择：

   - 对于 Protocol（协议），选择 AWS Lambda。
   - 对于 Endpoint (终端节点)，选择您在the section called “第 3 部分：创建 Lambda 函数”中创建的 Lambda 函数。
   - 对于 Version or alias（版本或别名），选择 default（原定设置）。
8. 选择创建订阅。

第 6 部分：创建 CloudWatch 警报

本节包含在中创建警报的过程 CloudWatch ，该警报将在指标达到特定阈值时触发。警报触发后会向在the section called “第 5 部分：创建 Amazon SNS 主题”中创建的 Amazon SNS 主题发送通知，然后该主题执行在the section called “第 3 部分：创建 Lambda 函数”中创建的 Lambda 函数。

创建 CloudWatch 警报

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 使用区域选择器来选择要在其中自动暂停电子邮件发送的区域。
3. 在左侧的导航窗格中，选择警报。
4. 选择创建警报。
5. 在创建警报窗口中的 SES 指标 下，选择配置集指标。
6. 在 ses:configuration-set 列中，找到您要为其创建警报的配置集。在 Metric Name（指标名称）下，选择以下选项之一：

   - 声誉。 BounceRate— 如果您想在配置集的总体硬退回率超过您定义的阈值时暂停发送该配置集的电子邮件，请选择此指标。

- 声誉。 ComplaintRate— 如果您想在配置集的总体投诉率超过您定义的阈值时暂停发送该配置集的电子邮件，请选择此指标。

选择下一步。

7. 完成以下步骤：

   - 在 Alarm Threshold（告警阈值）下，对于 Name（名称），键入警报的名称。
   - 在“随时：声誉”之下。 BounceRate或随时：声誉。 ComplaintRate，指定导致警报触发的阈值。

     > **Note**
     >
     > 如果 Amazon SES 账户的总体退回邮件率超过 10% 或 Amazon SES 账户的总体投诉率超过 0.5%，则 Amazon SES 账户会自动置于审核状态。当您指定触发 CloudWatch 警报的退回率或投诉率时，我们建议您使用远低于这些比率的值，以防止您的账户受到审查。

   - 在 Actions（操作）下，为 Whenever this alarm（每当此告警）选择 State is ALARM（状态为“告警”）。对于 Send notification to (发送通知到)，选择您在the section called “第 5 部分：创建 Amazon SNS 主题”中创建的 Amazon SNS 主题。

   选择创建警报。

第 7 部分：测试解决方案

您现在可以测试警报以确保它在进入 ALARM 状态时执行 Lambda 函数。您可以使用 CloudWatch API 中的`SetAlarmState`操作来临时更改警报的状态。

本节中的过程是可选的，但我们建议您完成这些过程来验证整个解决方案的配置是否正确。

测试解决方案

1. 在命令行键入以下命令，查看配置集的电子邮件发送状态：

   ```
   aws ses describe-configuration-set --configuration-set-name ConfigSet
   ```

   如果为配置集启用了发送，您会看到以下输出：

```
{
    "ConfigurationSet": {
        "Name": "ConfigSet"
    },
    "ReputationOptions": {
        "ReputationMetricsEnabled": true,
        "SendingEnabled": true
    }
}
```

如果 SendingEnabled 的值为 true，说明该配置集的电子邮件发送目前已启用。

2. 在命令行键入以下命令，将警报状态临时更改为 ALARM：

```
aws cloudwatch set-alarm-state \
--alarm-name MyAlarm \
--state-value ALARM \
--state-reason "Testing execution of Lambda function"
```

将前面的命令替换为您*MyAlarm*在中创建的警报的名称the section called “第 6 部分：创建 CloudWatch 警报”。

> **Note**
>
> 当您执行此命令时，警报状态会从 OK 切换为 ALARM 并在几秒后切换回 OK。您可以在 CloudWatch 控制台中警报的 “历史记录” 选项卡上或使用DescribeAlarmHistory操作来查看这些状态更改。

3. 在命令行键入以下命令，查看配置集的电子邮件发送状态：

```
aws ses describe-configuration-set \
--configuration-set-name ConfigSet
```

如果 Lambda 函数执行成功，您将看到类似于以下示例的输出：

```
{
    "ConfigurationSet": {
        "Name": "ConfigSet"
    },
    "ReputationOptions": {
```

```
        "ReputationMetricsEnabled": true,
        "SendingEnabled": false
    }
}
```

如果 SendingEnabled 的值为 false，则将禁用配置集的电子邮件发送，指示 Lambda 函数已成功执行。

4. 完成the section called “第 4 部分：重新启用配置集的电子邮件发送”中的步骤，为配置集重新启用电子邮件发送。

使用 Amazon 监控 SES 事件 EventBridge

EventBridge 是一项无服务器服务，它使用事件将应用程序组件连接在一起，使您可以更轻松地构建可扩展的事件驱动应用程序。事件驱动型架构是一种构建松耦合软件系统的风格，这些系统通过发出和响应事件来协同工作。事件是 JSON 格式的消息，通常表示资源或环境的变化，或是其他管理事件。

某些 SES 功能将生成您在创建事件目的地时定义的事件并将其发送到 EventBridge 默认事件总线。事件总线是接收事件并将其传送到零个或多个目的地或目标的路由器。与事件总线关联的规则会在事件到达时进行评估。每条规则都会检查事件是否与规则的模式相匹配。如果事件确实匹配，则将事件 EventBridge 发送到指定的目标。

当功能发生状态更改或状态更新 EventBridge 时，SES 会向发送事件。您可以使用 EventBridge 规则将事件路由到您定义的目标。这些事件将尽最大努力传输，可能会不按顺序传输。

主题

SES 事件

以下事件由 SES 功能生成并发送到中的默认事件总线 EventBridge。有关更多信息，包括每种事件类型的详细数据，请参阅 ???。

虚拟可交付性管理器 Advisor 事件

事件类型	说明
Advisor 推荐状态打开	每当虚拟可交付性管理器 Advisor 中打开新推荐时都会生成事件。
Advisor 推荐状态已解决	每当虚拟可交付性管理器 Advisor 中解决了新推荐时都会生成事件。

SES 电子邮件发送事件

事件类型	说明
电子邮件已退回	收件人的邮件服务器永久拒绝了电子邮件的硬退信。（只有当 SES 重试一段时间后仍无法发送邮件时才包括软退信。）
电子邮件已单击	收件人单击了电子邮件中包含的一个或多个链接。
电子邮件投诉已收到	电子邮件已成功送达收件人的邮件服务器，但收件人将其标记为垃圾邮件。
电子邮件送达	SES 已成功将电子邮件传送到收件人的邮件服务器。
电子邮件传送延迟	无法将电子邮件传送给收件人的邮件服务器，因为临时出现问题。例如，当收件人的收件箱已满，或者当接收电子邮件服务器遇到临时问题时，可能会发生传送延迟。
电子邮件已打开	收件人已收到消息并在其电子邮件客户端中打开了邮件。
电子邮件已拒绝	SES 已接受电子邮件，但确定它包含病毒，而未尝试将其传送到收件人的邮件服务器。
电子邮件呈现失败	由于模板呈现问题，未发送电子邮件。当模板数据丢失或模板参数与数据不匹配时，可能会发生此事件类型。（此事件类型仅在您使用 SendTemplatedEmail 或 SendBulkTemplatedEmail API 操作发送电子邮件时发生。）
电子邮件已发送	发送请求成功，SES 将尝试将消息传送到收件人的邮件服务器。（如果使用账户级别或全局抑制，SES 仍会将其计为发送，但会抑制送达。）
电子邮件已订阅	电子邮件已成功发送，但收件人通过单击电子邮件标头中的 List-Unsubscribe 或脚注中的 Unsubscribe 链接更新了订阅首选项。

SES 事件架构参考

来自 AWS 服务的所有事件都有一组公共字段，其中包含有关事件的元数据，例如作为事件来源的 AWS 服务、事件的生成时间、事件发生的账户和区域等。有关这些常规字段的定义，请参阅《EventBridge 用户指南》中的事件结构参考。

此外，每个事件都有一个 `detail` 字段，其中包该特定事件专有的数据。下面的参考定义了各种 SES 事件的详细信息字段。

使用 EventBridge 选择和管理 SES 事件时，请记住以下几点：

- 来自 SES 的所有事件的 `source` 字段都设置为 `aws.ses`。
- `detail-type` 字段指定事件类型。请参阅the section called “SES 事件”中的事件类型表。
- `detail` 字段包含该特定事件专有的数据。

  对于某些事件类型，例如虚拟可交付性管理器的事件类型，详细信息字段是一个相当简单的数据字符串，该字符串由一组有限的静态值填充而成。相反，电子邮件发送事件的详细信息字段更为复杂，因为它可以由许多详细信息子字段组成，这些子字段是静态和动态值的组合，例如发送电子邮件的时间戳、收件人地址和许多其他电子邮件属性。

主题

虚拟可交付性管理器 Advisor

以下架构参考定义了虚拟可交付性管理器 Advisor 状态事件专有的字段。

所有事件架构（例如`version`、`idaccount`、和其他）中显示的常规字段的定义可在EventBridge 用户指南的事件结构参考中找到。`source` 和 `detail-type` 字段包含在下面的参考中，因为它们包含 SES 事件的 SES 特定值。

`source`

标识生成事件的服务。对于 SES 事件，此值为 `aws.ses`。

`detail-type`

标识事件的类型。

该字段的值列在 the section called “SES 事件”中的虚拟可交付性管理器 Advisor 事件表中。

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

此字段的值可以是：

- `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 例如：虚拟可交付性管理器 Advisor 状态事件

以下是 `Advisor Recommendation Status Open` 事件类型的虚拟可交付性管理器 Advisor 状态事件示例。此示例中的详细信息事件值为 `SPF record was not found.`。

```
{
 "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 电子邮件发送状态架构

以下架构参考定义了 SES 电子邮件发送状态事件专有的字段。

所有事件架构（例如version、idaccount、和其他）中显示的常规字段的定义可在EventBridge 用户指南的事件结构参考中找到。source 和 detail-type 字段包含在下面的参考中，因为它们包含 SES 事件的 SES 特定值。

`source`

标识生成事件的服务。对于 SES 事件，此值为 aws.ses。

`detail-type`

标识事件的类型。

此字段的值列在 the section called “SES 事件”中的 SES 电子邮件发送事件表中。

`detail`

包含关于事件信息的 JSON 对象。生成事件的服务决定该字段的内容。

此处无法列出此字段的所有可能值，因为它们由在任何给定时刻发送的每封唯一电子邮件生成的静态和动态值组成。但是，提供了一个示例，让您大致了解此字段可以包含的数据类型。使用 EventBridge 沙盒可以找到所有电子邮件发送事件类型的详细数据示例，请参阅在中指定示例事件 EventBridge。

以下是为 SES 电子邮件发送事件 Email Rendering Failed 生成的详细信息数据示例：

```
...,
  "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 示例：电子邮件发送状态事件

以下是事件类型 `Email Rendering Failed` 的完整电子邮件发送状态事件示例。此示例中的详细事件值是基于特定电子邮件的电子邮件发送事件的静态和动态值的组合。

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

与 SES 事件 EventBridge 配合使用

默认情况下，SES 将事件发送到 EventBridge 默认事件总线。您可以在默认事件总线上创建规则，以识别要发送 EventBridge 到一个或多个指定目标的特定事件。每条规则都包含一个事件模式， EventBridge 用于在事件到达事件总线时进行匹配。如果事件与给定规则的事件模式匹配，则 EventBridge 会将该事件发送到规则中指定的目标。

在中 EventBridge，定义事件模式通常是创建新规则或编辑现有规则的更大过程的一部分。要了解如何创建 EventBridge 规则，请参阅EventBridge 用户指南中的创建对事件做出反应的 Amazon EventBridge 规则。

通过使用中的沙盒功能 EventBridge，您可以快速定义事件模式并使用示例事件来确认模式与所需事件相匹配，而无需先创建或编辑规则。有关使用沙盒的详细说明，请参阅EventBridge 用户指南中的使用 EventBridge 沙盒测试事件模式。

在 EventBridge 沙盒中指定 SES 示例事件

您可以为 SES 事件选择示例事件，用它们来测试自己创建的事件模式。

在 EventBridge 沙盒中指定 SES 示例事件

1. 打开亚马逊 EventBridge 控制台，网址为https://console.aws.amazon.com/events/。
2. 在导航窗格中选择开发人员资源，然后选择沙盒，再在沙盒页面上选择事件模式选项卡。
3. 对于事件来源，选择AWS 事件或 EventBridge 合作伙伴事件。
4. 在示例事件部分中，为示例事件类型选择 AWS 事件。
5. 对于示例事件，请向下滚动到 SES，然后选择所需的 SES 事件。

   EventBridge 显示该事件类型的示例事件及其所有详细数据。

   然后，您可以使用此事件来测试在事件模式部分创建的事件模式，或者将其用作创建自己的模式测试（将在下一节中介绍）示例事件的基础。

创建并测试 SES 事件的事件模式

按上一节所述选择示例事件后，您可以创建事件模式并使用示例事件来确保它与所需的事件相匹配。

创建和测试与 EventBridge 沙盒中的 SES 事件相匹配的事件模式

1. 打开亚马逊 EventBridge 控制台，网址为https://console.aws.amazon.com/events/。
2. 在导航窗格中选择开发人员资源，然后选择沙盒，再在沙盒页面上选择事件模式选项卡。
3. 对于事件源，选择AWS 事件或 EventBridge 合作伙伴事件，然后选择要测试的示例事件，如上一节所述。
4. 向下滚动到创建方法，然后选择使用模式表单。
5. 在事件模式部分中，为事件源选择 AWS 服务。
6. 在 AWS 服务下，选择 SES。
7. 对于事件类型，选择要匹配的 SES 事件类型。

   EventBridge 显示与所选 SES 事件匹配的最小事件模式，由`source`和`detail-type`字段组成。

   在以下两个示例中，第一个事件模式与所有 `Advisor Recommendation Status Resolved` 事件匹配，在第二个示例中，匹配所有 `Email Bounced` 事件：

   ```
   {
     "source": ["aws.ses"],
     "detail-type": ["Advisor Recommendation Status Resolved"]
   }
   ```

   ```
   {
     "source": ["aws.ses"],
     "detail-type": ["Email Bounced"]
   }
   ```
8. 要更改事件模式，请选择编辑模式，然后在 JSON 编辑器中进行更改。

   您也可以匹配一个或多个详细数据字段中的值。这包括为字段值指定多个可行值。

   在以下示例中，将详细信息字段添加到生成的最小事件模式中，其 `data` 字段值指定为 `DKIM record was not found`，以便查找所有具有相同详细信息值的虚拟可交付性管理器 Advisor 事件：

   ```
   {
   ```

```
  "source": ["aws.ses"],
  "detail-type": ["Advisor Recommendation Status Resolved"],
  "detail": {
    "data": ["DKIM record was not found."]
  }
}
```

在此示例中，添加了详细信息子字段，以报告 2024-08-05 从 noreply@example.com 发送的所有退回电子邮件所生成的事件。（此处使用前缀匹配作为内容筛选的一部分。）：

```
{
  "source": ["aws.ses"],
  "detail-type": ["Email Bounced"],
  "detail": {
    "mail": {
      "timestamp": [{
        "prefix": "2024-08-05"
      }],
      "source": ["noreply@example.com"]
    }
  }
}
```

阅读EventBridge 用户指南中的事件模式很重要，它解释了您在 JSON 编辑器中输入的事件模式值必须用方括号括起来，`[...]`因为它被视为数组。此外还提供了有关如何构造高级事件模式的更多信息。

9. 要测试您的事件模式是否与您在上述示例事件窗格中指定的示例事件相匹配，请选择测试模式。如果匹配，JSON 编辑器底部的绿色横幅将显示“Sample event matched the event pattern”。
10. 要对选择测试模式后的错误进行故障排除，请执行以下操作：

    - 如果存在与 JSON 相关的错误，则消息将指出原因，例如“Event pattern is not valid. Reason: "data" must be an object or an array at line: 5, column: 14”。要解决这个问题，请用方括号 `[...]` 将第 5 行的值括起来。
    - 如果示例事件中的值与您的事件模式之间存在差异，则将显示消息“Sample event did not match the event pattern”。这意味着您要测试的一个或多个值与示例事件生成器生成的示例值不同。要解决此问题，请继续执行其余步骤。
11. 要更改示例事件中的示例值以成功测试您的事件模式，请在示例事件窗格中，选择 JSON 编辑器下的复制。
12. 对于编辑器上方的示例事件类型，选择输入我自己的旁的单选按钮。

13. 将示例事件粘贴到 JSON 编辑器中，对于您在事件模式中使用的任何字段，请替换相同字段的值以匹配您在事件模式中指定的值。
14. 向下滚动到“事件模式”窗格，然后再次选择测试模式。如果所有值都输入正确且匹配，则 JSON 编辑器底部将显示一个绿色横幅，指示“Sample event matched the event pattern”。

其他 EventBridge 资源

有关如何使用 EventBridge 处理和管理事件的更多信息，请参阅 Amazon EventBridge 用户指南中的以下主题。

- 有关事件总线工作原理的详细信息，请参阅 Amazon EventBridge 事件总线。
- 有关事件结构的信息，请参阅 Events
- 有关构造事件模式 EventBridge 以便在将事件与规则进行匹配时使用的信息，请参阅事件模式
- 有关创建规则以指定 EventBridge 处理哪些事件的信息，请参阅规则
- 有关指定向哪些服务或其他目标 EventBridge 发送匹配事件的信息，请参阅 Targets

使用 Amazon SES 的代码示例 AWS SDKs

以下代码示例展示了如何将 Amazon SES 与 AWS 软件开发套件 (SDK) 一起使用。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

使用 Amazon SES 的代码示例 AWS SDKs

以下代码示例展示了如何将 Amazon SES 与 AWS 软件开发套件 (SDK) 一起使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务服务结合来完成特定任务的代码示例。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

    - 与 AWS SDK ListReceiptFilters 配合使用
    - 与 AWS SDK ListTemplates 配合使用
    - 与 AWS SDK SendBulkTemplatedEmail 配合使用
    - SendEmail与 AWS SDK 或 CLI 配合使用
    - SendRawEmail与 AWS SDK 或 CLI 配合使用
    - 与 AWS SDK SendTemplatedEmail 配合使用
    - 与 AWS SDK UpdateTemplate 配合使用
    - VerifyDomainIdentity与 AWS SDK 或 CLI 配合使用
    - VerifyEmailIdentity与 AWS SDK 或 CLI 配合使用
- 使用 Amazon SES 的场景 AWS SDKs
    - 构建 Amazon Transcribe 流式传输应用程序
    - 使用软件开发工具包将 Amazon AWS SES 电子邮件和域名身份从一个 AWS 地区复制到另一个区域
    - 创建 Web 应用程序来跟踪 DynamoDB 数据
    - 创建 Amazon Redshift 项目追踪器
    - 创建 Aurora Serverless 工作项跟踪器
    - 使用软件开发工具包使用 Amazon Rekognition 检测图像中的个人防护装备 AWS
    - 使用软件开发工具包使用 Amazon Rekognition 检测图像中的物体 AWS
    - 使用 Amazon Rekognition 使用软件开发工具包检测视频中的人物和物体 AWS
    - 生成凭证以连接到 Amazon SES SMTP 端点
    - 使用 Step Functions 调用 Lambda 函数
    - 使用软件开发工具包验证电子邮件身份并通过 Amazon AWS SES 发送消息

使用 Amazon SES 的基本示例 AWS SDKs

以下代码示例展示了如何使用 Amazon 简单电子邮件服务的基础知识 AWS SDKs。

示例

- 使用 Amazon SES 执行的操作 AWS SDKs
    - 与 AWS SDK CreateReceiptFilter 配合使用
    - 与 AWS SDK CreateReceiptRule 配合使用
    - 与 AWS SDK CreateReceiptRuleSet 配合使用

- 与 AWS SDK CreateTemplate 配合使用
- DeleteIdentity与 AWS SDK 或 CLI 配合使用
- 与 AWS SDK DeleteReceiptFilter 配合使用
- 与 AWS SDK DeleteReceiptRule 配合使用
- 与 AWS SDK DeleteReceiptRuleSet 配合使用
- 与 AWS SDK DeleteTemplate 配合使用
- 与 AWS SDK DescribeReceiptRuleSet 配合使用
- GetIdentityVerificationAttributes与 AWS SDK 或 CLI 配合使用
- GetSendQuota与 AWS SDK 或 CLI 配合使用
- 将 GetSendStatistics 与 CLI 配合使用
- 与 AWS SDK GetTemplate 配合使用
- ListIdentities与 AWS SDK 或 CLI 配合使用
- 与 AWS SDK ListReceiptFilters 配合使用
- 与 AWS SDK ListTemplates 配合使用
- 与 AWS SDK SendBulkTemplatedEmail 配合使用
- SendEmail与 AWS SDK 或 CLI 配合使用
- SendRawEmail与 AWS SDK 或 CLI 配合使用
- 与 AWS SDK SendTemplatedEmail 配合使用
- 与 AWS SDK UpdateTemplate 配合使用
- VerifyDomainIdentity与 AWS SDK 或 CLI 配合使用
- VerifyEmailIdentity与 AWS SDK 或 CLI 配合使用

使用 Amazon SES 执行的操作 AWS SDKs

以下代码示例演示了如何使用执行单个 Amazon SES 操作 AWS SDKs。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 Amazon SES API，是必须在上下文中运行的较大型程序的代码节选。您可以在使用 Amazon SES 的场景 AWS SDKs 中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Simple Email Service API 参考。

示例

- 与 AWS SDK CreateReceiptFilter 配合使用
- 与 AWS SDK CreateReceiptRule 配合使用
- 与 AWS SDK CreateReceiptRuleSet 配合使用
- 与 AWS SDK CreateTemplate 配合使用
- DeleteIdentity与 AWS SDK 或 CLI 配合使用
- 与 AWS SDK DeleteReceiptFilter 配合使用
- 与 AWS SDK DeleteReceiptRule 配合使用
- 与 AWS SDK DeleteReceiptRuleSet 配合使用
- 与 AWS SDK DeleteTemplate 配合使用
- 与 AWS SDK DescribeReceiptRuleSet 配合使用
- GetIdentityVerificationAttributes与 AWS SDK 或 CLI 配合使用
- GetSendQuota与 AWS SDK 或 CLI 配合使用
- 将 GetSendStatistics 与 CLI 配合使用
- 与 AWS SDK GetTemplate 配合使用
- ListIdentities与 AWS SDK 或 CLI 配合使用
- 与 AWS SDK ListReceiptFilters 配合使用
- 与 AWS SDK ListTemplates 配合使用
- 与 AWS SDK SendBulkTemplatedEmail 配合使用
- SendEmail与 AWS SDK 或 CLI 配合使用
- SendRawEmail与 AWS SDK 或 CLI 配合使用
- 与 AWS SDK SendTemplatedEmail 配合使用
- 与 AWS SDK UpdateTemplate 配合使用
- VerifyDomainIdentity与 AWS SDK 或 CLI 配合使用
- VerifyEmailIdentity与 AWS SDK 或 CLI 配合使用

与 AWS SDK **`CreateReceiptFilter`** 配合使用

以下代码示例演示如何使用 `CreateReceiptFilter`。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考CreateReceiptFilter中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考CreateReceiptFilter中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用CreateReceiptFilter于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **CreateReceiptRule** 配合使用

以下代码示例演示如何使用 CreateReceiptRule。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考CreateReceiptRule中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考CreateReceiptRule中的。

Python

适用于 Python 的 SDK（Boto3）

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

创建可供 Amazon S3 放置传入电子邮件副本的 Amazon S3 存储桶，并创建规则以便为特定收件人列表复制传入电子邮件到该存储桶。

```
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
```

- 有关 API 的详细信息，请参阅适用CreateReceiptRule于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **CreateReceiptRuleSet** 配合使用

以下代码示例演示如何使用 CreateReceiptRuleSet。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考CreateReceiptRuleSet中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考CreateReceiptRuleSet中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用CreateReceiptRuleSet于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **CreateTemplate** 配合使用

以下代码示例演示如何使用 CreateTemplate。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 验证电子邮件身份与发送消息

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考CreateTemplate中的。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考CreateTemplate中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考CreateTemplate中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用CreateTemplate于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteIdentity`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteIdentity`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 验证电子邮件身份与发送消息

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考DeleteIdentity中的。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考DeleteIdentity中的。

CLI

AWS CLI

删除身份

以下示例使用 `delete-identity` 命令从通过 Amazon SES 验证的身份列表中删除身份：

```
aws ses delete-identity --identity user@example.com
```

有关已验证身份的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址和域”。

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

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考DeleteIdentity中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用DeleteIdentity于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **DeleteReceiptFilter** 配合使用

以下代码示例演示如何使用 DeleteReceiptFilter。

C++

SDK for C++

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

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

```
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考DeleteReceiptFilter中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考DeleteReceiptFilter中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用DeleteReceiptFilter于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **DeleteReceiptRule** 配合使用

以下代码示例演示如何使用 DeleteReceiptRule。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考DeleteReceiptRule中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考DeleteReceiptRule中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用DeleteReceiptRule于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **DeleteReceiptRuleSet** 配合使用

以下代码示例演示如何使用 DeleteReceiptRuleSet。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考DeleteReceiptRuleSet中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考DeleteReceiptRuleSet中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用DeleteReceiptRuleSet于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **DeleteTemplate** 配合使用

以下代码示例演示如何使用 DeleteTemplate。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 验证电子邮件身份与发送消息

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考DeleteTemplate中的。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考DeleteTemplate中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考DeleteTemplate中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用DeleteTemplate于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **DescribeReceiptRuleSet** 配合使用

以下代码示例演示了如何使用 DescribeReceiptRuleSet。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用DescribeReceiptRuleSet于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

GetIdentityVerificationAttributes与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetIdentityVerificationAttributes。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 验证电子邮件身份与发送消息

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考GetIdentityVerificationAttributes中的。

CLI

AWS CLI

获取身份列表的 Amazon SES 验证状态

以下示例使用 `get-identity-verification-attributes` 命令检索身份列表的 Amazon SES 验证状态：

```
aws ses get-identity-verification-attributes --
identities "user1@example.com" "user2@example.com"
```

输出：

```
{
   "VerificationAttributes": {
       "user1@example.com": {
           "VerificationStatus": "Success"
       },
       "user2@example.com": {
           "VerificationStatus": "Pending"
       }
   }
}
```

如果调用此命令时，列表包含从未提交验证的身份，则该身份将不会出现在输出中。

有关已验证身份的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址和域”。

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

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用GetIdentityVerificationAttributes于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

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

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考GetIdentityVerificationAttributes中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

GetSendQuota与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetSendQuota。

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考GetSendQuota中的。

CLI

AWS CLI

获取 Amazon SES 发送限制

以下示例使用 `get-send-quota` 命令返回 Amazon SES 发送限制：

```
aws ses get-send-quota
```

输出：

```
{
   "Max24HourSend": 200.0,
   "SentLast24Hours": 1.0,
   "MaxSendRate": 1.0
}
```

Max24 HourSend 是您的发送配额，这是您在 24 小时内可以发送的最大电子邮件数量。发送配额反映一个滚动的时段。每当您尝试发送电子邮件时，Amazon SES 都会检查您在过去 24 小时内发送的电子邮件数量。只要您发送电子邮件总数小于您的配额，发送请求就会被接受，并发送您的电子邮件。

SentLast24 小时是您在过去 24 小时内发送的电子邮件数量。

MaxSendRate 是您每秒可以发送的最大电子邮件数。

请注意，发送限制基于收件人而不是消息。例如，一封包含 10 个收件人的电子邮件占用 10 份发送配额。

有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“管理您的 Amazon SES 发送限制”。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回用户的当前发送限制。

```
Get-SESSendQuota
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) GetSendQuota中的。

适用于 PowerShell V5 的工具

示例 1：此命令返回用户的当前发送限制。

```
Get-SESSendQuota
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) GetSendQuota中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetSendStatistics** 与 CLI 配合使用

以下代码示例演示如何使用 GetSendStatistics。

CLI

AWS CLI

获取 Amazon SES 发送统计信息

以下示例使用 get-send-statistics 命令返回 Amazon SES 发送统计信息

```
aws ses get-send-statistics
```

输出：

```
{
   "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
        }
    ]
}
```

结果是数据点的列表，表示过去两周的发送活动。列表中的每个数据点都包含 15 分钟间隔的统计信息。

在此示例中，只有两个数据点，因为用户在过去两周内发送的唯一电子邮件是在两个 15 分钟间隔内发送的。

有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“监控您的 Amazon SES 使用情况统计信息”。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回用户的发送统计信息。结果是数据点的列表，表示过去两周的发送活动。列表中的每个数据点都包含 15 分钟间隔的统计信息。

```
Get-SESSendStatistic
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) GetSendStatistics中的。

适用于 PowerShell V5 的工具

示例 1：此命令返回用户的发送统计信息。结果是数据点的列表，表示过去两周的发送活动。列表中的每个数据点都包含 15 分钟间隔的统计信息。

```
Get-SESSendStatistic
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) GetSendStatistics中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **GetTemplate** 配合使用

以下代码示例演示如何使用 GetTemplate。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 验证电子邮件身份与发送消息

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考GetTemplate中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考GetTemplate中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用GetTemplate于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListIdentities`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListIdentities`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 跨区域复制电子邮件和域身份

- 验证电子邮件身份与发送消息

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考ListIdentities中的。

C++

SDK for C++

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

```
//! 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;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考ListIdentities中的。

CLI

AWS CLI

列出特定 AWS 账户的所有身份（电子邮件地址和域名）

以下示例使用 `list-identities` 命令列出已提交到 Amazon SES 进行验证的所有身份：

```
aws ses list-identities
```

输出：

```
{
    "Identities": [
      "user@example.com",
      "example.com"
    ]
}
```

返回的列表包含所有身份，无论验证状态如何（已验证、待验证、失败等）。

在此示例中，由于未指定 identity-type 参数，因此返回了电子邮件地址和 域。

有关验证的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址和域”。

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

Java

适用于 Java 的 SDK 2.x

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

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListIdentities中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考ListIdentities中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：无论验证状态如何，此命令都会返回包含特定 AWS 账户的所有身份（电子邮件地址和域名）的列表。

```
Get-SESIdentity
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) ListIdentities中的。

适用于 PowerShell V5 的工具

示例 1：无论验证状态如何，此命令都会返回包含特定 AWS 账户的所有身份（电子邮件地址和域名）的列表。

```
Get-SESIdentity
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) ListIdentities中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用ListIdentities于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

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

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考ListIdentities中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **ListReceiptFilters** 配合使用

以下代码示例演示如何使用 ListReceiptFilters。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考ListReceiptFilters中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考ListReceiptFilters中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用ListReceiptFilters于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **ListTemplates** 配合使用

以下代码示例演示如何使用 ListTemplates。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 验证电子邮件身份与发送消息

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考ListTemplates中的。

Java

适用于 Java 的 SDK 2.x

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

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListTemplates中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考ListTemplates中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用ListTemplates于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **SendBulkTemplatedEmail** 配合使用

以下代码示例演示了如何使用 SendBulkTemplatedEmail。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考SendBulkTemplatedEmail中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

SendEmail与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 SendEmail。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 验证电子邮件身份与发送消息

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考SendEmail中的。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考SendEmail中的。

CLI

AWS CLI

使用 Amazon SES 发送格式化的电子邮件

以下示例使用 `send-email` 命令发送格式化的电子邮件：

```
aws ses send-email --from sender@example.com --destination file://
destination.json --message file://message.json
```

输出：

```
{
   "MessageId": "EXAMPLEf3a5efcd1-51adec81-d2a4-4e3f-9fe2-5d85c1b23783-000000"
}
```

目标和消息是 JSON 数据结构，保存在当前目录下的 .json 文件中。这些文件如下所示：

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"
       }
   }
}
```

请将发件人和收件人的电子邮件地址替换为您要使用的电子邮件地址。请注意，发件人的电子邮件地址必须已通过 Amazon SES 验证。在您获得 Amazon SES 的生产访问权限之前，您还必须验证每个收件人的电子邮件地址，除非收件人是 Amazon SES 邮箱模拟器。有关验证的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址和域”。

输出中的消息 ID 表示 send-email 调用成功。

如果您没有收到电子邮件，请检查垃圾邮件。

有关发送格式化电子邮件的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“使用 Amazon SES API 发送格式化的电子邮件”。

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

Java

适用于 Java 的 SDK 2.x

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

```
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.printStackTrace();
        }
    }

    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.printStackTrace();
        }
    }

    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");
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendEmail中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考SendEmail中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用SendEmail于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

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

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考SendEmail中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

SendRawEmail与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 SendRawEmail。

CLI

AWS CLI

使用 Amazon SES 发送原始电子邮件

以下示例使用 send-raw-email 命令发送带有 TXT 附件的电子邮件：

```
aws ses send-raw-email --raw-message file://message.json
```

输出：

```
{
   "MessageId": "EXAMPLEf3f73d99b-c63fb06f-d263-41f8-a0fb-d0dc67d56c07-000000"
}
```

原始消息是一个 JSON 数据结构，保存在当前目录下名为 message.json 的文件中。其中包含以下内容：

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

如您所见，“Data”是一个长字符串，包含 MIME 格式的全部原始电子邮件内容，其中包括一个名为 attachment.txt 的附件。

请将 sender@example.com 和 recipient@example.com 替换为您要使用的地址。请注意，发件人的电子邮件地址必须已通过 Amazon SES 验证。在您获得 Amazon SES 的生产访问权限之前，您还必须验证收件人的电子邮件地址，除非收件人是 Amazon SES 邮箱模拟器。有关验证的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址和域”。

输出中的消息 ID 表示对的调用 send-raw-email成功。

如果您没有收到电子邮件，请检查垃圾邮件。

有关发送原始电子邮件的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“使用 Amazon SES API 发送原始电子邮件”。

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

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

使用 nodemailer 发送带附件的电子邮件。

```
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);
        }
      },
    );
  });
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考SendRawEmail中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **SendTemplatedEmail** 配合使用

以下代码示例演示如何使用 SendTemplatedEmail。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 验证电子邮件身份与发送消息

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考SendTemplatedEmail中的。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考SendTemplatedEmail中的。

Java

适用于 Java 的 SDK 2.x

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

```
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);
```

```
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendTemplatedEmail中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考SendTemplatedEmail中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用SendTemplatedEmail于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **UpdateTemplate** 配合使用

以下代码示例演示如何使用 UpdateTemplate。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 验证电子邮件身份与发送消息

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考UpdateTemplate中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考[UpdateTemplate](#)中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用UpdateTemplate于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`VerifyDomainIdentity`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `VerifyDomainIdentity`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 跨区域复制电子邮件和域身份
- 验证电子邮件身份与发送消息

CLI

AWS CLI

使用 Amazon SES 验证域

以下示例使用 `verify-domain-identity` 命令验证域：

```
aws ses verify-domain-identity --domain example.com
```

输出：

```
{
   "VerificationToken": "eoEmxw+YaYhb3h3iVJHuXMJXqeu1q1/wwmvjuEXAMPLE"
}
```

要完成域验证，您必须在域的 DNS 设置中添加一条包含返回的验证令牌的 TXT 记录。有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证域”。

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

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

```
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;
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考VerifyDomainIdentity中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用VerifyDomainIdentity于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

VerifyEmailIdentity与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 VerifyEmailIdentity。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 跨区域复制电子邮件和域身份
- 验证电子邮件身份与发送消息

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考VerifyEmailIdentity中的。

C++

SDK for C++

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

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

- 有关 API 的详细信息，请参阅 适用于 C++ 的 AWS SDK API 参考VerifyEmailIdentity中的。

CLI

AWS CLI

使用 Amazon SES 验证电子邮件地址

以下示例使用 `verify-email-identity` 命令验证电子邮件地址：

```
aws ses verify-email-identity --email-address user@example.com
```

在使用 Amazon SES 发送电子邮件之前，您必须验证从中发送电子邮件的地址或域，以证明您拥有该地址或域。如果您尚未获得生产访问权限，则还需要验证所有收件电子邮件地址，Amazon SES 邮箱模拟器提供的电子邮件地址除外。

接 verify-email-identity到电话后，该电子邮件地址将收到一封验证电子邮件。用户必须单击此电子邮件中的链接才能完成验证过程。

有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的“在 Amazon SES 中验证电子邮件地址”。

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

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

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

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

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考VerifyEmailIdentity中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用VerifyEmailIdentity于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

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

```
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
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考VerifyEmailIdentity中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon SES 的场景 AWS SDKs

以下代码示例向您展示了如何使用在 Amazon SES 中实现常见场景 AWS SDKs。这些场景向您展示了如何通过调用 Amazon SES 中的多个函数或与其他 AWS 服务结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 构建 Amazon Transcribe 流式传输应用程序
- 使用软件开发工具包将 Amazon AWS SES 电子邮件和域名身份从一个 AWS 地区复制到另一个区域
- 创建 Web 应用程序来跟踪 DynamoDB 数据
- 创建 Amazon Redshift 项目追踪器
- 创建 Aurora Serverless 工作项跟踪器
- 使用软件开发工具包使用 Amazon Rekognition 检测图像中的个人防护装备 AWS
- 使用软件开发工具包使用 Amazon Rekognition 检测图像中的物体 AWS
- 使用 Amazon Rekognition 使用软件开发工具包检测视频中的人物和物体 AWS
- 生成凭证以连接到 Amazon SES SMTP 端点
- 使用 Step Functions 调用 Lambda 函数
- 使用软件开发工具包验证电子邮件身份并通过 Amazon AWS SES 发送消息

构建 Amazon Transcribe 流式传输应用程序

以下代码示例展示如何构建可实时录制、转录与翻译实时音频，并通过电子邮件发送结果的应用程序。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

演示了如何使用 Amazon Transcribe 构建可实时录制、转录与翻译实时音频，并通过 Amazon Simple Email Service (Amazon SES) 以电子邮件发送结果的应用程序。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Comprehend
- Amazon SES

- Amazon Transcribe
- Amazon Translate

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包将 Amazon AWS SES 电子邮件和域名身份从一个 AWS 地区复制到另一个区域

以下代码示例显示如何将 Amazon SES 电子邮件和域名身份从一个 AWS 区域复制到另一个区域。由 Route 53 管理域身份时，会为目标区域将验证记录复制到域。

Python

适用于 Python 的 SDK（Boto3）

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

```
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()
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的以下主题。
  - ListIdentities
  - SetIdentityNotificationTopic
  - VerifyDomainDkim
  - VerifyDomainIdentity
  - VerifyEmailIdentity

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 Web 应用程序来跟踪 DynamoDB 数据

以下代码示例显示如何创建一个 Web 应用程序，来跟踪 Amazon DynamoDB 表中的工作项，并使用 Amazon Simple Email Service (Amazon SES) 来发送报告。

.NET

适用于 .NET 的 SDK

展示如何使用 Amazon DynamoDB .NET API 创建用于跟踪 DynamoDB 工作数据的动态 Web 应用程序。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- DynamoDB
- Amazon SES

Java

适用于 Java 的 SDK 2.x

展示如何使用 Amazon DynamoDB API 创建用于跟踪 DynamoDB 工作数据的动态 Web 应用程序。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- DynamoDB
- Amazon SES

Kotlin

适用于 Kotlin 的 SDK

展示如何使用 Amazon DynamoDB API 创建用于跟踪 DynamoDB 工作数据的动态 Web 应用程序。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- DynamoDB
- Amazon SES

Python

适用于 Python 的 SDK（Boto3）

演示如何使用创建 REST 服务，通过亚马逊简单电子邮件服务 (Amazon SES) 跟踪亚马逊 DynamoDB 中的工作项目并通过电子邮件发送报告。 适用于 Python (Boto3) 的 AWS SDK 此示例使用 Flask Web 框架处理 HTTP 路由，并且与 React 网页集成来呈现完整功能的 Web 应用程序。

- 构建与 AWS 服务集成的 Flask REST 服务。
- 读取、写入和更新存储在 DynamoDB 表中的工作项。
- 使用 Amazon SES 发送工作项的电子邮件报告。

有关完整的源代码以及如何设置和运行的说明，请参阅上的 “AWS 代码示例存储库” 中的完整示例 GitHub。

本示例中使用的服务

- DynamoDB
- Amazon SES

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 Amazon Redshift 项目追踪器

以下代码示例展示如何使用 Amazon Redshift 数据库创建用于跟踪和报告工作项的 Web 应用程序。

Java

适用于 Java 的 SDK 2.x

展示如何创建 Web 应用程序来跟踪与报告存储与 Amazon Redshift 数据库的工作项。

有关如何设置查询 Amazon Redshift 数据的 Spring REST API 以及供 React 应用程序使用的完整源代码和说明，请参阅上的完整示例。GitHub

本示例中使用的服务

- Amazon Redshift
- Amazon SES

Kotlin

适用于 Kotlin 的 SDK

展示如何创建 Web 应用程序来跟踪与报告存储与 Amazon Redshift 数据库的工作项。

有关如何设置查询 Amazon Redshift 数据的 Spring REST API 以及供 React 应用程序使用的完整源代码和说明，请参阅上的完整示例。GitHub

本示例中使用的服务

- Amazon Redshift
- Amazon SES

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 Aurora Serverless 工作项跟踪器

以下代码示例演示了如何创建用于跟踪 Amazon Aurora Serverless 数据库中的工作项，以及使用 Amazon Simple Email Service (Amazon SES) 发送报告的 Web 应用程序。

.NET

适用于 .NET 的 SDK

演示如何使用创建一个 Web 应用程序，该 适用于 .NET 的 AWS SDK 应用程序可跟踪 Amazon Aurora 数据库中的工作项目，并使用亚马逊简单电子邮件服务 (Amazon SES) 通过电子邮件发送报告。此示例使用使用 React.js 构建的前端与 RESTful .NET 后端进行交互。

- 将 React Web 应用程序与 AWS 服务集成。
- 列出、添加、更新和删除 Aurora 表中的项目。
- 使用 Amazon SES 以电子邮件形式发送已筛选工作项的报告。
- 使用随附的 AWS CloudFormation 脚本部署和管理示例资源。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Aurora
- Amazon RDS
- Amazon RDS 数据服务
- Amazon SES

C++

SDK for C++

演示了如何创建用于跟踪和报告存储在 Amazon Aurora Serverless 数据库中的工作项的 Web 应用程序。

有关如何设置查询 Amazon Aurora Serverless 数据的 C++ REST API 以及如何由 React 应用程序使用的完整源代码和说明，请参阅上的GitHub完整示例。

本示例中使用的服务

- Aurora
- Amazon RDS
- Amazon RDS 数据服务
- Amazon SES

Java

适用于 Java 的 SDK 2.x

展示如何创建 Web 应用程序来跟踪与报告存储与 Amazon RDS 数据库的工作项。

有关如何设置查询 Amazon Aurora Serverless 数据的 Spring REST API 以及如何让 React 应用程序使用的完整源代码和说明，请参阅上的GitHub完整示例。

有关如何设置和运行使用 JDBC API 的示例的完整源代码和说明，请参阅上的完整示例。GitHub

本示例中使用的服务

- Aurora
- Amazon RDS
- Amazon RDS 数据服务
- Amazon SES

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

演示如何使用 适用于 JavaScript 的 AWS SDK (v3) 创建一个 Web 应用程序，该应用程序使用亚马逊简单电子邮件服务 (Amazon SES) Service 跟踪亚马逊 Aurora 数据库中的工作项目并通过电子邮件发送报告。此示例使用由 React.js 构建的前端与 Express Node.js 后端进行交互。

- 将 React.js 网络应用程序与集成 AWS 服务。
- 列出、添加以及更新 Aurora 表中的项目。
- 使用 Amazon SES 以电子邮件形式发送已筛选工作项的报告。
- 使用随附的 AWS CloudFormation 脚本部署和管理示例资源。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Aurora
- Amazon RDS
- Amazon RDS 数据服务

- Amazon SES

Kotlin

适用于 Kotlin 的 SDK

展示如何创建 Web 应用程序来跟踪与报告存储与 Amazon RDS 数据库的工作项。

有关如何设置查询 Amazon Aurora Serverless 数据的 Spring REST API 以及如何让 React 应用程序使用的完整源代码和说明，请参阅上的GitHub完整示例。

本示例中使用的服务

- Aurora
- Amazon RDS
- Amazon RDS 数据服务
- Amazon SES

PHP

适用于 PHP 的 SDK

演示如何使用创建一个 Web 应用程序，该 适用于 PHP 的 AWS SDK 应用程序通过亚马逊简单电子邮件服务 (Amazon SES) Semple Service 跟踪亚马逊 RDS 数据库中的工作项目并通过电子邮件发送报告。此示例使用使用 React.js 构建的前端与 RESTful PHP 后端进行交互。

- 将 React.js 网络应用程序与 AWS 服务集成。
- 列出、添加、更新和删除 Amazon RDS 表中的项目。
- 使用 Amazon SES 以电子邮件发送已筛选工作项的报告。
- 使用随附的 AWS CloudFormation 脚本部署和管理示例资源。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Aurora
- Amazon RDS
- Amazon RDS 数据服务
- Amazon SES

Python

适用于 Python 的 SDK（Boto3）

演示如何使用创建 REST 服务，该服务通过亚马逊简单电子邮件服务 (Amazon SES) 跟踪亚马逊Aurora无服务器数据库中的工作项目并通过电子邮件发送报告。 适用于 Python (Boto3) 的 AWS SDK 此示例使用 Flask Web 框架处理 HTTP 路由，并且与 React 网页集成来呈现完整功能的 Web 应用程序。

- 构建与 AWS 服务集成的 Flask REST 服务。
- 读取、写入和更新存储在 Aurora Serverless 数据库中的工作项。
- 创建包含数据库凭据的 AWS Secrets Manager 密钥，并使用它来验证对数据库的调用。
- 使用 Amazon SES 发送工作项的电子邮件报告。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Aurora
- Amazon RDS
- Amazon RDS 数据服务
- Amazon SES

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包使用 Amazon Rekognition 检测图像中的个人防护装备 AWS

以下代码示例展示如何构建采用 Amazon Rekognition 来检测图像中的个人防护设备（PPE）的应用程序。

Java

适用于 Java 的 SDK 2.x

演示如何创建使用个人防护设备检测图像的 AWS Lambda 功能。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- DynamoDB

- Amazon Rekognition
- Amazon S3
- Amazon SES

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包使用 Amazon Rekognition 检测图像中的物体 AWS

以下代码示例展示如何构建采用 Amazon Rekognition 来按类别检测图像中对象的应用程序。

.NET

适用于 .NET 的 SDK

展示如何使用 Amazon Rekognition .NET API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

演示如何使用 Amazon Rekogn 适用于 JavaScript 的 AWS SDK ition 和，创建一款应用程序，该应用程序使用 Amazon Rekognition 按类别识别位于亚马逊简单存储服务 (Amazon S3) Simple S3 存储桶中的图像中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

了解如何：

- 使用 Amazon Cognito 创建未经身份验证的用户。
- 使用 Amazon Rekognition 分析包含对象的图像。
- 为 Amazon SES 验证电子邮件地址。
- 使用 Amazon SES 发送电子邮件通知。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

适用于 Kotlin 的 SDK

展示如何使用 Amazon Rekognition Kotlin API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

适用于 Python 的 SDK（Boto3）

向您展示如何使用创建 适用于 Python (Boto3) 的 AWS SDK 允许您执行以下操作的 Web 应用程序：

- 将照片上载到 Amazon Simple Storage Service (Amazon S3) 存储桶。
- 使用 Amazon Rekognition 来分析和标注照片。
- 使用 Amazon Simple Email Service (Amazon SES) 发送图像分析的电子邮件报告。

此示例包含两个主要组件：使用 React 构建的 JavaScript 网页和使用 Flask-RESTful 构建的用 Python 编写的 REST 服务。

可以使用 React 网页执行以下操作：

- 显示存储在 S3 存储桶中的图像列表。
- 将计算机中的图像上载到 S3 存储桶。
- 显示图像和用于识别图像中检测到的物品的标注。
- 获取 S3 存储桶中所有图像的报告并发送报告电子邮件。

该网页调用 REST 服务。该服务将请求发送到 AWS 以执行以下操作：

- 获取并筛选 S3 存储桶中的图像列表。
- 将照片上载到 S3 存储桶。
- 使用 Amazon Rekognition 分析各张照片并获取标注列表，这些标注用于识别在照片中检测到的物品。
- 分析 S3 存储桶中的所有照片，然后使用 Amazon SES 通过电子邮件发送报告。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Rekognition

- Amazon S3
- Amazon SES

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon Rekognition 使用软件开发工具包检测视频中的人物和物体 AWS

以下代码示例展示如何使用 Amazon Rekognition 检测视频中的人物和对象。

Java

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，以检测位于 Amazon Simple Storage Service (Amazon S3) 存储桶的视频当中的人脸和对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

适用于 Python 的 SDK（Boto3）

通过启动异步检测任务，使用 Amazon Rekognition 来检测视频中的人脸、对象和人物。此示例还将 Amazon Rekognition 配置为在任务完成时通知 Amazon Simple Notification Service (Amazon SNS) 主题，并订阅该主题的 Amazon Simple Queue Service (Amazon SQS) 队列。当队列收到有关任务的消息时，将检索该任务并输出结果。

最好在上查看此示例 GitHub。有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

生成凭证以连接到 Amazon SES SMTP 端点

以下代码示例展示如何生成连接以连接到 Amazon SES SMTP 端点。

Python

适用于 Python 的 SDK（Boto3）

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

```
#!/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()
```

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Step Functions 调用 Lambda 函数

以下代码示例说明如何创建按顺序调用 AWS Lambda 函数的 AWS Step Functions 状态机。

Java

适用于 Java 的 SDK 2.x

演示如何使用 AWS Step Functions 和创建 AWS 无服务器工作流程。 AWS SDK for Java 2.x每个工作流程步骤都是使用 AWS Lambda 函数实现的。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包验证电子邮件身份并通过 Amazon AWS SES 发送消息

以下代码示例展示了如何：

- 使用 Amazon SES 添加和验证电子邮件地址。
- 发送标准电子邮件。
- 创建模板和发送模板化电子邮件。
- 使用 Amazon SES SMTP 服务器发送邮件。

Python

适用于 Python 的 SDK（Boto3）

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

使用 Amazon SES 验证电子邮件地址，然后发送邮件。

```
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)
```

创建函数来封装 Amazon SES 身份操作。

```
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
```

创建函数来封装 Amazon SES 模板操作。

```
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
```

创建函数来封装 Amazon SES 电子邮件操作。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的以下主题。
  - CreateTemplate
  - DeleteIdentity
  - DeleteTemplate
  - GetIdentityVerificationAttributes
  - GetTemplate
  - ListIdentities
  - ListTemplates
  - SendEmail
  - SendTemplatedEmail
  - UpdateTemplate
  - VerifyDomainIdentity
  - VerifyEmailIdentity

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon SES API v2 的代码示例 AWS SDKs

以下代码示例展示了如何将 Amazon SES API v2 与 AWS 软件开发套件 (SDK) 一起使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务服务结合来完成特定任务的代码示例。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- 使用 Amazon SES API v2 的基本示例 AWS SDKs

使用 Amazon SES API v2 的基本示例 AWS SDKs

以下代码示例展示了如何使用亚马逊简单电子邮件服务 API v2 的基础知识。 AWS SDKs

示例

- SendEmail与 AWS SDK 一起使用

使用 Amazon SES API v2 的操作 AWS SDKs

以下代码示例演示了如何使用执行单个 Amazon SES API v2 操作。 AWS SDKs每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 Amazon SES API v2 API，是必须在上下文中运行的较大型程序的代码节选。您可以在使用 Amazon SES API v2 的场景 AWS SDKs 中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Simple Email Service API v2 API 参考。

示例

- CreateContact与 AWS SDK 一起使用
- CreateContactList与 AWS SDK 一起使用
- CreateEmailIdentity与 AWS SDK 一起使用
- CreateEmailTemplate与 AWS SDK 一起使用
- DeleteContactList与 AWS SDK 一起使用
- DeleteEmailIdentity与 AWS SDK 一起使用
- DeleteEmailTemplate与 AWS SDK 一起使用
- GetEmailIdentity与 AWS SDK 一起使用
- ListContactLists与 AWS SDK 一起使用
- ListContacts与 AWS SDK 一起使用
- SendEmail与 AWS SDK 一起使用

`CreateContact`与 AWS SDK 一起使用

以下代码示例演示如何使用 `CreateContact`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 时事通讯场景

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考CreateContact中的。

Java

适用于 Java 的 SDK 2.x

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

```
    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;
      }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateContact中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用CreateContact于 Python 的AWS SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

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

```
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(())
}
```

- 有关 API 的详细信息，请参阅适用CreateContact于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateContactList`与 AWS SDK 一起使用

以下代码示例演示如何使用 `CreateContactList`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 时事通讯场景

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考CreateContactList中的。

Java

适用于 Java 的 SDK 2.x

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

```
    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;
        }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateContactList中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用CreateContactList于 Python 的AWS SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

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

```
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(())
}
```

- 有关 API 的详细信息，请参阅适用CreateContactList于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateEmailIdentity`与 AWS SDK 一起使用

以下代码示例演示如何使用 `CreateEmailIdentity`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 时事通讯场景

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考CreateEmailIdentity中的。

Java

适用于 Java 的 SDK 2.x

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

```
   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;
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateEmailIdentity中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用CreateEmailIdentity于 Python 的AWS SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

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

```
        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)),
            },
```

```
        }
```

- 有关 API 的详细信息，请参阅适用CreateEmailIdentity于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateEmailTemplate`与 AWS SDK 一起使用

以下代码示例演示如何使用 `CreateEmailTemplate`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 时事通讯场景

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考CreateEmailTemplate中的。

Java

适用于 Java 的 SDK 2.x

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

```
    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;
        }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateEmailTemplate中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用CreateEmailTemplate于 Python 的AWS SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

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

```
        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)),
            },
        }
```

- 有关 API 的详细信息，请参阅适用CreateEmailTemplate于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DeleteContactList与 AWS SDK 一起使用

以下代码示例演示如何使用 DeleteContactList。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 时事通讯场景

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考DeleteContactList中的。

Java

适用于 Java 的 SDK 2.x

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

```
    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();
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteContactList中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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)
```

- 有关 API 的详细信息，请参阅适用DeleteContactList于 Python 的AWS SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

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

```
        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}")),
        }
```

- 有关 API 的详细信息，请参阅适用DeleteContactList于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteEmailIdentity`与 AWS SDK 一起使用

以下代码示例演示如何使用 `DeleteEmailIdentity`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 时事通讯场景

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考DeleteEmailIdentity中的。

Java

适用于 Java 的 SDK 2.x

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

```
     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.");
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteEmailIdentity中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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)
```

- 有关 API 的详细信息，请参阅适用DeleteEmailIdentity于 Python 的AWS SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

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

```
            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));
                }
            }
```

- 有关 API 的详细信息，请参阅适用DeleteEmailIdentity于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteEmailTemplate`与 AWS SDK 一起使用

以下代码示例演示如何使用 `DeleteEmailTemplate`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 时事通讯场景

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考DeleteEmailTemplate中的。

Java

适用于 Java 的 SDK 2.x

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

```
    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();
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteEmailTemplate中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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)
```

- 有关 API 的详细信息，请参阅适用DeleteEmailTemplate于 Python 的AWS SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

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

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

- 有关 API 的详细信息，请参阅适用DeleteEmailTemplate于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetEmailIdentity`与 AWS SDK 一起使用

以下代码示例演示了如何使用 `GetEmailIdentity`。

Rust

适用于 Rust 的 SDK

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

确定电子邮件地址是否已验证。

```
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(())
}
```

- 有关 API 的详细信息，请参阅适用GetEmailIdentity于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListContactLists`与 AWS SDK 一起使用

以下代码示例演示了如何使用 `ListContactLists`。

Rust

适用于 Rust 的 SDK

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

```
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(())
}
```

- 有关 API 的详细信息，请参阅适用ListContactLists于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListContacts`与 AWS SDK 一起使用

以下代码示例演示如何使用 `ListContacts`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 时事通讯场景

.NET

适用于 .NET 的 SDK

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

```
    /// <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>();
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考ListContacts中的。

Java

适用于 Java 的 SDK 2.x

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

```
     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;
      }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListContacts中的。

Python

适用于 Python 的 SDK（Boto3）

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

```
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
```

- 有关 API 的详细信息，请参阅适用ListContacts于 Python 的AWS SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

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

```
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(())
```

```
}
```

- 有关 API 的详细信息，请参阅适用ListContacts于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

SendEmail与 AWS SDK 一起使用

以下代码示例演示如何使用 SendEmail。

.NET

适用于 .NET 的 SDK

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

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

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考SendEmail中的。

Java

适用于 Java 的 SDK 2.x

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

发送邮件。

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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);
        }
    }
}
```

使用模板发送消息。

```
      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());
      }
```

发送带有标题信息的消息。

```
public class SendwithHeader {

    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();

        String bodyHTML = """
                <html>
                    <head></head>
                    <body>
                        <h1>Hello!</h1>
                        <p>See the list of customers.</p>
                    </body>
                </html>
                """;

        sendWithHeader(sesv2Client, sender, recipient, subject, bodyHTML);
        sesv2Client.close();
    }

    /**
     * Sends an email using the AWS SES V2 client.
     *
     * @param sesv2Client the SES V2 client to use for sending the email
     * @param sender the email address of the sender
     * @param recipient the email address of the recipient
     * @param subject the subject of the email
     * @param bodyHTML the HTML content of the email body
     */
    public static void sendWithHeader(SesV2Client sesv2Client,
                                      String sender,
                                      String recipient,
                                      String subject,
                                      String bodyHTML) {
        EmailContent emailContent = EmailContent.builder()
                .simple(Message.builder()
                        .body(b -> b.html(c ->
c.charset(UTF_8.name()).data(bodyHTML))
                                .text(c ->
c.charset(UTF_8.name()).data(bodyHTML)))
                        .subject(c -> c.charset(UTF_8.name()).data(subject))
```

```
                        .headers(List.of(
                                MessageHeader.builder()
                                        .name("List-Unsubscribe")
                                        .value("<https://nutrition.co/?
address=x&topic=x>, <mailto:unsubscribe@nutrition.co?subject=TopicUnsubscribe>")
                                        .build(),
                                MessageHeader.builder()
                                        .name("List-Unsubscribe-Post")
                                        .value("List-Unsubscribe=One-Click")
                                        .build()))
                        .build())
                .build();

        SendEmailRequest request = SendEmailRequest.builder()
                .fromEmailAddress(sender)
                .destination(d -> d.toAddresses(recipient))
                .content(emailContent)
                .build();

        try {
            SendEmailResponse response = sesv2Client.sendEmail(request);
            System.out.println("Email sent! Message ID: " +
 response.messageId());
        } catch (SesV2Exception e) {
            System.err.println("Failed to send email: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendEmail中的。

Python

适用于 Python 的 SDK（Boto3）

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

向联系人列表中的所有成员发送消息。

```
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}'.")
```

使用模板向联系人列表中的所有成员发送消息。

```
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},
            )
```

- 有关 API 的详细信息，请参阅适用SendEmail于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

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

```
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)
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 AWS SDK API 参考SendEmail中的。

Rust

适用于 Rust 的 SDK

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

向联系人列表中的所有成员发送消息。

```
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(())
}
```

使用模板向联系人列表中的所有成员发送消息。

```
            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)),
            }
```

- 有关 API 的详细信息，请参阅适用SendEmail于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon SES API v2 的场景 AWS SDKs

以下代码示例向您展示了如何使用在 Amazon SES API v2 中实现常见场景。 AWS SDKs这些场景向您展示了如何通过调用 Amazon SES API v2 中的多个函数或与其他 AWS 服务结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 使用软件开发工具包的完整亚马逊 SES API v2 新闻简报场景 AWS

使用软件开发工具包的完整亚马逊 SES API v2 新闻简报场景 AWS

以下代码示例展示了如何运行 Amazon SES API v2 新闻简报场景。

.NET

适用于 .NET 的 SDK

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

运行场景。

```
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);
    }
}
```

适用于服务操作的包装器。

```
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;
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 AWS SDK API 参考》中的以下主题。
  - CreateContact
  - CreateContactList

- CreateEmailIdentity
- CreateEmailTemplate
- DeleteContactList
- DeleteEmailIdentity
- DeleteEmailTemplate
- ListContacts
- SendEmail。simple
- SendEmail。模板

Java

适用于 Java 的 SDK 2.x

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

```
    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();
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的以下主题。
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate

- DeleteContactList
- DeleteEmailIdentity
- DeleteEmailTemplate
- ListContacts
- SendEmail。simple
- SendEmail。模板

Python

适用于 Python 的 SDK（Boto3）

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

```
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)
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的以下主题。
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail。simple
  - SendEmail。模板

Rust

适用于 Rust 的 SDK

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

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

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的以下主题。
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate

- DeleteContactList
- DeleteEmailIdentity
- DeleteEmailTemplate
- ListContacts
- SendEmail。simple
- SendEmail。模板

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 Amazon SES 与 AWS 软件开发工具包配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon Simple Email Service 中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将此描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 Amazon Simple Email AWS 服务的合规计划，请参阅按合规计划AWS 划分的范围内服务（划分）。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

该文档帮助您了解如何在使用 Amazon Simple Email Service 时应用责任共担模式。它说明了如何配置 Amazon Simple Email Service 以实现您的安全性和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的亚马逊简单电子邮件服务资源。

> **Note**
>
> 如果您需要举报滥用 AWS 资源的情况，包括垃圾邮件和恶意软件分发，请不要使用本开发者指南任何页面上的反馈链接，因为该表格是由 AWS 文档团队而不是 AWS 信任与安全部门收到的。相反，在 “如何举报滥用 AWS 资源？” 页面上，按照说明联系 AWS 信任与安全团队，举报任何类型的亚马逊 AWS 滥用行为。

内容

Amazon Simple Email Service 中的数据保护

分 AWS 担责任模型适用于亚马逊简单电子邮件服务中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与资源通信。 AWS 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API 或 AWS 服务 使用亚马逊简单电子邮件服务或其他服务时 AWS SDKs。 AWS CLI在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

内容

Amazon SES 的静态数据加密

默认情况下，Amazon SES 会加密所有静态数据。默认加密有助于降低保护数据的操作开销和复杂性。加密还支持创建符合严格加密合规性和法规要求的 Mail Manager 存档。

SES 提供以下加密选项：

- AWS 自有密钥 — SES 默认使用这些密钥。您无法查看、管理或使用 AWS 自有密钥，也无法审核其使用情况。但是，无需采取任何措施或更改任何计划即可保护用于加密数据的密钥。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的 AWS 自有密钥。
- 客户自主管理型密钥 – SES 支持使用由您自己创建、拥有和管理的对称客户自主管理型密钥。由于您可以完全控制此加密，因此可以执行以下任务：
  - 制定和维护关键策略
  - 建立和维护 IAM 策略和授权
  - 启用和禁用密钥策略
  - 轮换加密材料
  - 添加标签
  - 创建密钥别名
  - 安排密钥删除

  要使用您自己的密钥，请在创建 SES 资源时选择客户自主管理型密钥。

  有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户托管密钥。

> **Note**
>
> SES 使用 AWS 自有密钥自动启用静态加密，不收取任何费用。
> 但是，使用客户管理的密钥需要 AWS KMS 付费。有关定价的更多信息，请参阅 AWS Key Management Service 定价。

创建客户托管密钥

您可以使用 AWS Management Console、或，创建对称的客户托管密钥。 AWS KMS APIs

创建对称的客户托管密钥

请按照《AWS Key Management Service 开发人员指南》中创建对称加密 KMS 密钥的步骤操作。

> **Note**
>
> 对于存档，您的密钥必须满足以下要求：
>
> - 密钥必须是对称的。
> - 密钥材料来源必须是 AWS_KMS。
> - 密钥的用法必须是 ENCRYPT_DECRYPT。

密钥策略

密钥策略控制对客户自主管理型密钥的访问。每个客户托管式密钥必须只有一个密钥策略，其中包含确定谁可以使用密钥以及如何使用密钥的声明。创建客户托管式密钥时，可以指定密钥策略。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的管理对客户托管密钥的访问。

要将客户自主管理型密钥与 Mail Manager 存档一起使用，您的密钥策略必须允许以下 API 操作：

- kms: DescribeKey — 提供客户托管的密钥详细信息，允许 SES 验证密钥。
- kms: GenerateDataKey — 允许 SES 生成用于加密静态数据的数据密钥。
- kms:Decrypt – 允许 SES 在将存储的数据返回给 API 客户端之前对其进行解密。

以下示例显示了典型的密钥策略：

```
{
            "Sid": "Allow SES to encrypt/decrypt",
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
```

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的在策略中指定权限。

有关问题排查的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的密钥访问问题排查。

为 Mail Manager 存档指定客户自主管理型密钥

您可以指定客户管理的密钥作为使用 AWS 自有密钥的替代方法。创建存档时，您可以通过输入 KMS 密钥 ARN 来指定数据密钥，Mail Manager 存档会使用该密钥来加密存档中的所有客户数据。

- KMS 密钥 ARN： AWS KMS 客户自主管理型密钥的密钥标识符。输入密钥 ID、密钥 ARN、别名名称或别名 ARN。

Amazon SES 加密上下文

加密上下文是一组可选的键值对，包含有关数据的其他上下文信息。

AWS KMS 使用加密上下文作为其他经过身份验证的数据来支持经过身份验证的加密。当您在加密数据的请求中包含加密上下文时，会将加密上下文 AWS KMS 绑定到加密数据。要解密数据，您必须在请求中包含相同的加密上下文。

> **Note**
>
> Amazon SES 不支持用于存档创建的加密上下文。相反，您可以使用 IAM 或 KMS 策略。有关策略示例，请参阅本节后面的存档创建策略。

Amazon SES 加密上下文

SES 在所有 AWS KMS 加密操作中使用相同的加密上下文，其中密钥为`aws:ses:arn`，值为资源 Amazon 资源名称 (ARN)。

Example

```
"encryptionContext": {
    "aws:ses:arn": "arn:aws:ses:us-west-2:111122223333:ExampleResourceName/
ExampleResourceID"
}
```

使用加密上下文进行监控

使用对称的客户自主管理型密钥来加密您的 SES 资源时，您还可以使用审计记录和日志中的加密上下文，来识别客户自主管理型密钥的使用情况。加密上下文还会显示在AWS CloudTrail 或 Amazon Logs 生成的 CloudWatch 日志中。

使用加密上下文控制对客户托管式密钥的访问

您可以使用密钥策略和 IAM 策略中的加密上下文作为 conditions 来控制对您的对称客户托管密钥的访问。您还可以在授权中使用加密上下文约束。

SES 在授权中使用加密上下文约束来控制对您账户或区域中客户自主管理型密钥的访问。授权约束要求授权允许的操作使用指定的加密上下文。

Example

以下是密钥政策声明示例，用于授予对特定加密上下文的客户托管密钥的访问权限。此策略语句中的条件要求授权具有指定加密上下文的加密上下文约束。

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

存档创建策略

以下策略示例演示了如何启用存档创建。这些策略适用于所有资产。

IAM 策略

```
{
            "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 策略

```
{
            "Sid": "Allow SES to encrypt/decrypt",
            "Effect": "Allow",
            "Principal": {
                "Service": "ses.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
```

```
            "Resource": "*"
        },
```

为 Amazon SES 监控您的加密密钥

当您将 AWS KMS 客户托管密钥与您的 Amazon SES 资源一起使用时，您可以使用AWS CloudTrail或Amazon CloudWatch Logs 来跟踪 SES 发送到的请求 AWS KMS。

以下示例是GenerateDataKeyDecrypt、和DescribeKey监控 SES 为访问由您的客户托管密钥加密的数据而调用的 KMS 操作 AWS CloudTrail 的事件：

GenerateDataKey

当您为资源启用 AWS KMS 客户托管密钥时，SES 会创建一个唯一的表密钥。它向发送GenerateDataKey请求 AWS KMS ，指定资源的 AWS KMS客户托管密钥。

当您为 Mail Manager 存档资源启用 AWS KMS 客户托管密钥GenerateDataKey时，它将在加密静态存档数据时使用。

以下示例事件记录了 GenerateDataKey 操作：

```
{
    "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

当您访问加密的资源时，SES 会调用 Decrypt 操作，以使用存储的加密数据密钥来访问加密数据。

以下示例事件记录了 Decrypt 操作：

```
{
    "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 使用 DescribeKey 操作来验证与您的资源关联的 AWS KMS 客户自主管理型密钥是否存在于账户和区域中。

以下示例事件记录了 DescribeKey 操作：

```
{
    "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"
}
```


了解更多

以下资源提供有关静态数据加密的更多信息。

- 有关 AWS Key Management Service 基本概念的更多信息，请参阅 《AWS Key Management Service 开发人员指南》。

- 有关 AWS Key Management Service的安全最佳实操的更多信息，请参阅 《AWS Key Management Service 开发人员指南》。

传输中加密

默认情况下，Amazon SES 使用操作 TLS。这意味着 Amazon SES 始终尝试与接收邮件服务器建立安全连接。如果无法建立安全连接，它将发送未加密邮件。您可以更改此行为，以便 Amazon SES 只有在能够建立安全连接的情况下，才会将邮件发送到接收电子邮件服务器。有关更多信息，请参阅 Amazon SES 和安全协议。

从 Amazon SES 中删除个人数据

根据您使用 Amazon SES 的方式，它可能会存储某些被视为个人信息的数据。例如，要使用 Amazon SES 发送电子邮件，您必须提供至少一个已验证的身份（电子邮件地址或域）。您可以使用 Amazon SES 控制台或者 Amazon SES API 永久删除这些个人数据。

本章提供了删除可能被视为个人信息的各种类型的数据的过程。

内容

从账户级别黑名单中删除电子邮件地址

Amazon SES 包括一个可选的账户级别黑名单。启用此功能后，导致退回或投诉的电子邮件地址会自动添加到黑名单中。电子邮件地址在删除之前会一直保留在名单中。有关账户级别黑名单的更多信息，请参阅使用 Amazon SES 账户级黑名单。

您可以使用 Amazon SES API v2 中的 DeleteSuppressedDestination 操作，从账户级别黑名单中删除电子邮件地址。此部分包括使用 AWS CLI删除电子邮件地址的过程。有关安装和配置 AWS CLI 的更多信息，请参阅 AWS Command Line Interface 用户指南。

要使用 AWS CLI从账户级黑名单中删除地址，请执行以下操作：

- 在命令行输入以下命令：

```
aws sesv2 delete-suppressed-destination --email-address recipient@example.com
```

在前面的命令中，将*recipient@example.com*替换为要从账户级禁止列表中删除的电子邮件地址。

删除有关使用 Amazon SES 发送的电子邮件的数据

当您使用 Amazon SES 发送电子邮件时，您可以将有关该电子邮件的信息发送给其他 AWS 服务。例如，您可以将有关电子邮件事件的信息（如传送、打开和单击）发送到 Firehose。该事件数据通常包含您的电子邮件地址以及从中发送电子邮件的 IP 地址。它还包含将电子邮件发送到的所有收件人的电子邮件地址。

您可以使用 Firehose 将电子邮件事件数据流式传输到多个目的地，包括亚马逊简单存储服务、亚马逊服务和亚马逊 Redshift。 OpenSearch 要删除该数据，您应该先停止将数据流式传输到 Firehose，然后删除已流式传输的数据。要停止将 Amazon SES 事件数据流式传输到 Firehose，您必须删除 Firehose 事件目标。

使用 Amazon SES 控制台删除 Firehose 事件目标

1. 打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在电子邮件发送下面，选择配置集。
3. 在配置集列表中，选择包含 Firehose 事件目标的配置集。
4. 在要删除的 Firehose 事件目标旁边，选择删除（ ）按钮。
5. 如有必要，请删除 Firehose 写入到其他服务的数据。有关更多信息，请参阅 the section called “删除存储的事件数据”。

您也可以使用 Amazon SES API 删除事件目标。以下过程使用 AWS Command Line Interface (AWS CLI) 与 Amazon SES API 进行交互。您也可以使用 AWS 软件开发工具包或直接发出 HTTP 请求与 API 进行交互。

要移除 Firehose 事件目的地，请使用 AWS CLI

1. 在命令行处，键入以下命令：

   ```
   aws sesv2 delete-configuration-set-event-destination --configuration-set-
   name configSet \
   --event-destination-name eventDestination
   ```

   在此命令中，*configSet*替换为包含 Firehose 事件目标的配置集的名称。*eventDestination*替换为 Firehose 事件目标的名称。
2. 如有必要，请删除 Firehose 写入到其他服务的数据。有关更多信息，请参阅 the section called “删除存储的事件数据”。

删除存储的事件数据

有关从其他 AWS 服务中删除信息的更多信息，请参阅以下文档：

- 《Amazon Simple Storage Service 用户指南》中的删除对象和存储桶
- 在《亚马逊 OpenSearch 服务开发者指南》中删除 OpenSearch 服务域
- 《Amazon Redshift 集群管理指南》中的删除集群

您还可以使用 Firehose 将电子邮件数据流式传输到 Splunk，Splunk 是一项不受支持 AWS 或管理的第三方服务。 AWS Management Console有关从 Splunk 中删除数据的更多信息，请咨询您的系统管理员或参阅 Splunk 网站上的文档。

删除有关身份的数据

身份包括用于通过 Amazon SES 发送电子邮件的电子邮件地址和域。在某些司法管辖区中，电子邮件地址或域可能会被视为个人身份数据。

使用 Amazon SES 控制台删除身份

1. 打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在身份管理下面，执行以下操作之一：

- 如果要删除域，请选择域。
- 如果要删除电子邮件地址，请选择电子邮件地址。

3. 选择要删除的身份，然后选择删除。
4. 在确认对话框中，选择是，删除身份。

您也可以使用 Amazon SES API 删除身份。以下过程使用 AWS Command Line Interface（AWS CLI）与 Amazon SES API 进行交互。您也可以使用 AWS 软件开发工具包或直接发出 HTTP 请求与 API 进行交互。

要删除身份，请使用 AWS CLI

- 在命令行处，键入以下命令：

```
aws ses delete-identity --identity sender@example.com
```

在此命令中，*sender@example.com*替换为要删除的身份。

删除发件人身份验证数据

发件人身份验证是指，配置 Amazon SES 以便其他用户可以代表您发送电子邮件的过程。要启用发件人授权，您必须创建一个策略，如使用 Amazon SES 的发送授权中所述。除了（与代表您发送电子邮件的个人或群组相关联）之外，这些政策还包含身份 AWS IDs（属于您）。您可以修改或删除发件人身份验证策略以删除该个人数据。以下过程介绍了如何删除这些策略。

使用 Amazon SES 控制台删除发件人身份验证策略

1. 打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在身份管理下面，执行以下操作之一：
   - 如果要删除的发件人身份验证策略与某个域关联，请选择域。
   - 如果要删除的发件人身份验证策略与某个电子邮件地址关联，请选择电子邮件地址。
3. 在身份策略下面，选择要删除的策略，然后选择删除策略。

您也可以使用 Amazon SES API 删除发件人身份验证策略。以下过程使用 AWS Command Line Interface (AWS CLI) 与 Amazon SES API 进行交互。您也可以使用 AWS 软件开发工具包或直接发出 HTTP 请求与 API 进行交互。

要删除发件人身份验证策略，请使用 AWS CLI

- 在命令行处，键入以下命令：

```
aws ses delete-identity-policy --identity example.com --policy-name samplePolicy
```

在此命令中，*example.com*替换为包含发件人身份验证策略的身份。*samplePolicy*替换为发件人身份验证策略的名称。

删除与接收规则相关的数据

如果您使用 Amazon SES 接收传入电子邮件，您可以创建应用于一个或多个身份（电子邮件地址或域）的接收规则。这些规则确定 Amazon SES 如何处理发送到指定身份的传入邮件。

使用 Amazon SES 控制台删除接收规则

1. 打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在电子邮件接收下面，选择规则集。
3. 如果接收规则是活动规则集的一部分，请选择查看活动规则集。否则，选择包含要删除的接收规则的规则集。
4. 在接收规则列表中，选择要删除的规则。
5. 在操作 菜单上，选择删除。
6. 在确认对话框中，选择删除。

您也可以使用 Amazon SES API 删除接收规则。以下过程使用 AWS Command Line Interface (AWS CLI) 与 Amazon SES API 进行交互。您也可以使用 AWS 软件开发工具包或直接发出 HTTP 请求与 API 进行交互。

要删除接收规则，请使用 AWS CLI

- 在命令行处，键入以下命令：

```
aws ses delete-receipt-rule --rule-set myRuleSet --rule-name myReceiptRule
```

在此命令中，*myRuleSet*替换为包含接收规则的接收规则集的名称。*myReceiptRule*替换为要删除的接收规则的名称。

删除与 IP 地址筛选器相关的数据

如果使用 Amazon SES 接收传入电子邮件，您可以创建筛选器以明确接受或阻止从特定 IP 地址发送的邮件。

使用 Amazon SES 控制台删除 IP 地址筛选器

1. 打开 Amazon SES 控制台，网址为https://console.aws.amazon.com/ses/。
2. 在电子邮件接收下面，选择 IP 地址筛选器。
3. 在 IP 地址筛选器列表中，选择要删除的筛选器，然后选择删除。

您也可以使用 Amazon SES API 删除 IP 地址筛选器。以下过程使用 AWS Command Line Interface (AWS CLI) 与 Amazon SES API 进行交互。您也可以使用 AWS 软件开发工具包或直接发出 HTTP 请求与 API 进行交互。

要删除 IP 地址过滤器，请使用 AWS CLI

- 在命令行处，键入以下命令：

  ```
  aws ses delete-receipt-filter --filter-name IPfilter
  ```

  在此命令中，*IPfilter*替换为要删除的 IP 地址过滤器的名称。

删除电子邮件模板中的数据

如果您使用电子邮件模板发送电子邮件，这些模板可能包含个人数据，具体取决于您如何配置这些模板。例如，您可能在模板中添加了一个电子邮件地址，收件人可以通过该地址进行联系以了解更多信息。

您只能使用 Amazon SES API 删除电子邮件模板。

要删除电子邮件模板，请使用 AWS CLI

- 在命令行处，键入以下命令：

  ```
  aws ses delete-template --template-name sampleTemplate
  ```

  在此命令中，*sampleTemplate*替换为要删除的电子邮件模板的名称。

删除自定义验证电子邮件模板中的数据

如果您使用自定义的模板验证新的电子邮件发送地址，这些模板可能包含个人数据，具体取决于您如何配置这些模板。例如，您可能在验证电子邮件模板中添加了一个电子邮件地址，收件人可以通过该地址进行联系以了解更多信息。

您只能使用 Amazon SES API 删除自定义验证电子邮件模板。

要删除自定义验证电子邮件模板，请使用 AWS CLI

- 在命令行处，键入以下命令：

```
aws ses delete-custom-verification-email-template --template-
name verificationEmailTemplate
```

在此命令中，*verificationEmailTemplate*替换为要删除的自定义验证电子邮件模板的名称。

通过关闭 AWS 账户删除所有个人数据

您也可以通过关闭您的 AWS 账户来删除 Amazon SES 中存储的所有个人数据。但是，此操作还会删除您在所有其他服务中存储的所有其他数据（个人或非个人数据）。 AWS

当您关闭 AWS 账户时，您的账户中的数据将 AWS 保留 90 天。在保留期过后，将永久删除该数据并且不可撤销。

要关闭您的 AWS 账户

关闭账户中介绍了如何关闭 AWS 账户的完整说明。 AWS

Amazon SES 中的 Identity and Access Management

您可以将 AWS Identity and Access Management (IAM) 与亚马逊简单电子邮件服务 (Amazon SES) 配合使用，以指定用户、群组或角色可以执行的 SES API 操作。(在本主题中，我们将这些实体统称为用户。) 您还可以控制用户可对电子邮件的“发件人”、收件人和“退回路径”地址使用的电子邮件地址。

例如，您可以创建一个 IAM 策略，允许组织中的用户发送电子邮件，但是不允许其执行管理操作（例如检查发送统计数据）。又例如，您可以编写一个策略，允许用户通过 SES 从您的账户发送电子邮件，但只在他们使用特定的“发件人”地址时才能这样做。

要使用 IAM，您可以定义一个 IAM 策略（一个用于显式定义权限的文档），然后将该策略附加到用户。要了解如何创建 IAM 策略，请参阅 IAM 用户指南。除了应用您在策略中设定的限制之外，用户与 SES 交互的方式或 SES 执行请求的方式没有变化。

> **Note**
>
> - 如果您的账户在 SES 沙盒中，其限制可能会阻止实施其中一些策略 – 请参阅请求生产环境访问权限。
> - 您还可以使用发送授权策略控制对 SES 的访问。不过，IAM 策略限制各个用户可执行的操作，发送授权策略限制可以使用各个经验证的身份的方式。此外，只有发送授权策略可以授予跨账户访问权限。有关发送授权的更多信息，请参阅使用 Amazon SES 的发送授权。

如果您正在查找有关如何为现有用户生成 SES SMTP 凭证的信息，请参阅获取 Amazon SES SMTP 凭证。

创建用于访问 SES 的 IAM 策略

本节介绍如何将 IAM 策略专门用于 SES。要了解创建 IAM 策略的常规方式，请参阅 IAM 用户指南。

有三个理由可能让您将 IAM 与 SES 结合使用：

- 限制电子邮件发送操作。
- 限制用户发送的电子邮件的“发件人”、收件人和“退回路径”地址。
- 控制 API 使用的一般方面，例如允许用户调用他们有权使用的时间段。 APIs

限制操作

要控制用户可执行的 SES 操作，您可以使用 IAM 策略的 `Action` 元素。您可以通过使用小写字符串 `Action` 作为 API 名称的前缀来将 `ses:` 元素设置为任何 SES API 操作。例如，您可以将 `Action` 设置为 `ses:SendEmail`、`ses:GetSendStatistics` 或 `ses:*`（适用于所有操作）。

然后，根据 `Action` 来指定 `Resource` 元素，如下所示：

如果该**`Action`**元素仅允许访问电子邮件发送 APIs（即**`ses:SendEmail`**和/或**`ses:SendRawEmail`**）：

- 要允许用户使用您的任何身份发送 AWS 账户，请`Resource`将其设置为`*`

- 要限制允许用户发送的身份，请Resource将其设置为允许用户使用的身份。 ARNs

如果该**Action**元素允许所有人访问 APIs：

- 如果您不希望限制用户发送邮件所用的身份，请将 Resource 设置为 *
- 如果您希望限制用户发送邮件时可用的身份，则需要创建两个策略（或位于一个策略中的两个语句）：
  - 其中一个Action设置为允许的明确列表 non-email-sending APIs ，Resource设置为*
  - 一个Action设置为电子邮件发送 APIs（和 ses:SendEmail /或ses:SendRawEmail）之一，并Resource设置为您允许用户使用的身份的 ARN。

有关可用的 SES 操作的列表，请参阅 Amazon Simple Email Service API 参考。如果该用户将使用 SMTP 接口，您必须至少允许对 ses:SendRawEmail 的访问。

限制电子邮件地址

如果您要将用户限制到特定电子邮件地址，则可以使用一个 Condition 数据块。在 Condition 数据块中，您将使用条件键来指定条件，如 IAM 用户指南中所述。通过使用条件键，您可以控制以下电子邮件地址：

> **Note**
>
> 这些电子邮件地址条件键仅适用于下表中 APIs 注明的。

条件键	描述	API
ses:Recipients	限制收件人地址，这包括“收件人”、“抄送”和“密件抄送”地址。	SendEmail ，SendRawEmail
ses:FromAddress	限制“发件人”地址。	SendEmail ，SendRawEmail ，SendBounce
ses:FromDisplayName	限制用作显示名称的“发件人”地址。	SendEmail ，SendRawEmail

条件键	描述	API
ses:FeedbackAddress	限制“退回路径”地址，这是供退回邮件和投诉通过电子邮件反馈转发发送给您的地址。有关电子邮件反馈转发的信息，请参阅通过电子邮件接收 Amazon SES 通知。	SendEmail ，SendRawEmail
ses:MultiRegionEndpointId	允许您控制发送电子邮件时使用的端点 ID	SendEmail ，SendBulkEmail

通过 SES API 版本进行限制

通过在条件中使用 ses:ApiVersion 键，您可以根据 SES API 的版本限制对 SES 的访问。

> **Note**
>
> SES SMTP 接口使用 SES API 版本 2 的 ses:SendRawEmail。

限制常规 API 使用

通过在条件中使用 AWS-wide 密钥，您可以根据允许用户访问的日期和时间等方面限制对 SES 的访问 APIs。SES 仅实现以下 AWS范围的策略密钥：

- aws:CurrentTime
- aws:EpochTime
- aws:SecureTransport
- aws:SourceIp
- aws:SourceVpc
- aws:SourceVpce
- aws:UserAgent
- aws:VpcSourceIp

有关这些键的更多信息，请参阅 IAM 用户指南。

SES 的 IAM 策略示例

本主题提供允许用户仅在特定条件下访问 SES 的策略的示例。

此节中的策略示例：

- 允许对所有 SES 操作的完全访问
- 允许仅访问 SES API 版本 2
- 仅允许对电子邮件发送操作的访问
- 限制发送的时间段
- 限制收件人地址
- 限制“发件人”地址
- 限制电子邮件发件人的显示名称
- 限制退回邮件和投诉反馈的目标

允许对所有 SES 操作的完全访问

以下策略允许用户调用任何 SES 操作。

JSON

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

允许仅访问 SES API 版本 2

以下策略允许用户只调用 API 版本 2 的 SES 操作。

JSON

```
{
                    "Version":"2012-10-17",
                    "Statement":[
                        {
                            "Effect":"Allow",
                            "Action":[
                            "ses:*"
                            ],
                            "Resource":"*",
                            "Condition": {
                                "StringEquals" : {
                                "ses:ApiVersion" : "2"
                                }
                            }
                        }
                    ]
                }
```

仅允许对电子邮件发送操作的访问

以下策略允许用户使用 SES 发送电子邮件，但不允许用户执行管理操作（如访问 SES 发送统计数据）。

JSON

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

限制发送的时间段

以下政策 APIs 仅允许用户在 2018 年 9 月致电 SES 发送电子邮件。

JSON

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

限制收件人地址

以下策略允许用户致电 SES 发送电子邮件 APIs，但仅限于域名 e xample.com 中的收件人地址（*StringLike*区分大小写）。

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
```

```
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Resource":"*",
      "Condition":{
        "ForAllValues:StringLike":{
          "ses:Recipients":[
            "*@example.com"
          ]
        }
      }
    }
  ]
}
```

限制“发件人”地址

以下政策允许用户致电 SES 发送电子邮件 APIs，但前提是 “发件人” 地址为 marketing@example.com。

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Resource":"*",
      "Condition":{
        "StringEquals":{
          "ses:FromAddress":"marketing@example.com"
        }
      }
    }
  ]
}
```

以下政策允许用户调用 SendBounceAPI，但前提是 “发件人” 地址为 bounce@example.com。

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "ses:SendBounce"
      ],
      "Resource":"*",
      "Condition":{
        "StringEquals":{
          "ses:FromAddress":"bounce@example.com"
        }
      }
    }
  ]
}
```

限制电子邮件发件人的显示名称

以下政策允许用户致电 SES 发送电子邮件 APIs，但前提`StringLike`是 “发件人” 地址的显示名称包括市场营销（区分大小写）。

JSON

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

```
        "StringLike":{
          "ses:FromDisplayName":"Marketing"
        }
      }
    }
  ]
}
```

限制退回邮件和投诉反馈的目标

以下政策允许用户致电 SES 发送电子邮件 APIs，但前提是电子邮件的 “返回路径” 设置为 feedback@example.com。

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "ses:SendEmail",
        "ses:SendRawEmail"
      ],
      "Resource":"*",
      "Condition":{
        "StringEquals":{
          "ses:FeedbackAddress":"feedback@example.com"
        }
      }
    }
  ]
}
```

AWS Amazon 简单电子邮件服务的托管策略

要向用户、群组和角色添加权限，使用 AWS 托管策略比自己编写策略要容易得多。创建仅为团队提供所需权限的 IAM 客户管理型策略需要时间和专业知识。要快速入门，您可以使用我们的 AWS 托管

策略。这些政策涵盖常见用例，可在您的 AWS 账户中使用。有关 AWS 托管策略的更多信息，请参阅 IAM 用户指南中的AWS 托管策略。

AWS 服务维护和更新 AWS 托管策略。您无法更改 AWS 托管策略中的权限。服务偶尔会向 AWS 托管策略添加其他权限以支持新功能。此类更新会影响附加策略的所有身份（用户、组和角色）。当推出新功能或有新操作可用时，服务最有可能更新 AWS 托管策略。服务不会从 AWS 托管策略中移除权限，因此策略更新不会破坏您的现有权限。

此外，还 AWS 支持跨多个服务的工作职能的托管策略。例如，ReadOnlyAccess AWS 托管策略提供对所有 AWS 服务和资源的只读访问权限。当服务启动一项新功能时， AWS 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅 IAM 用户指南中的适用于工作职能的AWS 托管式策略。

AWS 托管策略：Amazon A SESFull ccess

您可以将 `AmazonSESFullAccess` 策略附加到 IAM 身份。提供对 Amazon SES 的完全访问权限。

要查看此策略的权限，请参阅《AWS 托管策略参考》中的 Amazon A SESFull cces s。

AWS 托管策略：Amazon SESRead OnlyAccess

您可以将 `AmazonSESReadOnlyAccess` 策略附加到 IAM 身份。提供对 Amazon SES 的只读访问权限。

要查看此策略的权限，请参阅《AWS 托管策略参考》SESReadOnlyAccess中的 Amazon。

AWS 托管策略：Amazon SESService RolePolicy

无法将 `AmazonSESServiceRolePolicy` 策略附加到 IAM 实体。此附加到服务相关角色的策略允许 Amazon SES 代表您执行操作。有关更多信息，请参阅 Amazon SES 的服务相关角色权限。

要查看此策略的权限，请参阅《AWS 托管策略参考》SESServiceRolePolicy中的 Amazon。

Amazon 简单电子邮件服务更新了 AWS 托管政策

查看 Amazon Simple Email Service AWS 托管政策自该服务开始跟踪这些变更以来的详情和更新。

更改	描述	日期
Amazon Simple Email Service 添加了新的托管式策略	Amazon Simple Email Service 向服务相关角色 `AWSServic`	2024 年 5 月 13 日

更改	描述	日期
	eRoleForAmazonSES 添加了 AmazonSESServiceRolePolicy ，使 SES 可以代表您执行操作	
Amazon Simple Email Service 更新了一个策略定义	Amazon Simple Email Service 将此表（下行）中的前一个条目澄清为：亚马逊简单电子邮件服务已ses:BatchGetMetricData 添加到亚马逊SESReadOnlyAccess托管政策中——这将允许访问 SES API BatchGetMetricData	2024 年 4 月 30 日
Amazon Simple Email Service 更新了一个策略定义	Amazon Simple Email 服务已ses:BatchGet* 添加到亚马逊SESReadOnlyAccess托管策略中，这将允许访问 SES API BatchGetMetricData	2024 年 2 月 16 日
Amazon Simple Email Service 更改了两个策略定义	Amazon Simple Email Service 从 Amazon A SESFullccess 和 Amazon SESReadOnlyAccess 定义末尾删除了“通过 AWS 管理控制台”	2023 年 5 月 3 日
Amazon Simple Email Service 已开始跟踪更改	Amazon Simple Email Service 开始跟踪其 AWS 托管政策的更改	2023 年 4 月 5 日

对 Amazon SES 使用服务相关角色

亚马逊简单电子邮件服务 (SES) Simple Servic AWS Identity and Access Management e 使用 (IAM) 服务相关角色。服务相关角色是一种独特类型的 IAM 角色，它与 Amazon SES 直接相关。服务相关角色由 SES 预定义，包括该服务代表您调用其他 AWS 服务所需的所有权限。

服务相关角色可让您更轻松地设置 SES，因为您不必手动添加必要的权限。SES 定义其服务相关角色的权限，除非另外定义，否则只有 SES 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务相关角色。这可以保护您的 SES 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的AWS 服务，并在服务相关角色列中查找标有 “是” 的服务。选择是和链接，查看该服务的服务相关角色文档。

Amazon SES 的服务相关角色权限

SES 使用名为 AWSServiceRoleForAmazonSES 的服务相关角色 — 允许 SES 代表您的 SES 资源发布 Amazon CloudWatch 基本监控指标。

S AWSService RoleForAmazon ES 服务相关角色信任以下服务来代入该角色：

- `ses.amazonaws.com`

名为 Amazon 的角色权限策略SESServiceRolePolicy 是一种AWS 托管策略，允许 SES 对指定资源完成以下操作：

- 操作：`AWS/SES` CloudWatch 命名空间中的 `cloudwatch:PutMetricData`。此操作授予 SES 将指标数据放入 CloudWatch `AWS/SES`命名空间的权限。有关中可用的 SES 指标的更多信息 CloudWatch，请参阅Amazon SES 中的日志记录和监控。
- 操作：`AWS/SES/MailManager` CloudWatch 命名空间中的 `cloudwatch:PutMetricData`。此操作授予 SES 将指标数据放入 CloudWatch`AWS/SES/MailManager`命名空间的权限。有关中可用的 SES 指标的更多信息 CloudWatch，请参阅Amazon SES 中的日志记录和监控。
- 操作：`AWS/SES/Addons` CloudWatch 命名空间中的 `cloudwatch:PutMetricData`。此操作授予 SES 将指标数据放入 CloudWatch `AWS/SES/Addons`命名空间的权限。有关中可用的 SES 指标的更多信息 CloudWatch，请参阅Amazon SES 中的日志记录和监控。

您必须配置使用户、组或角色能够创建、编辑或删除服务相关角色的权限。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

为 Amazon SES 创建服务相关角色

您无需手动创建服务相关角色。当您在 AWS Management Console、或 AWS API 中创建 SES 资源时，SES 会为您创建服务相关角色。 AWS CLI

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您创建 SES 资源时，SES 会再次为您创建服务相关角色。

为 Amazon SES 编辑服务相关角色

SES 不允许您编辑 AWSService RoleForAmazon SES 服务相关角色。创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。

删除 SES 的服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，您必须先清除您的服务相关角色，然后才能手动删除它。

清除服务相关角色

必须先删除所有 SES 资源，然后才能使用 IAM 删除服务相关角色。

> **Note**
>
> 如果在您尝试删除资源时 SES 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

手动删除 服务相关角色

使用 IAM 控制台 AWS CLI、或 AWS API 删除 AWSService RoleForAmazon SES 服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的删除服务相关角色。

Amazon SES 服务相关角色支持的区域

SES 并非在提供该服务的每个区域中都支持使用服务相关角色。您可以在以下区域使用 AWSService RoleForAmazon SES 角色。

区域名称	区域标识	SES 支持
美国东部（弗吉尼亚州北部）	us-east-1	是
美国东部（俄亥俄州）	us-east-2	是
亚太地区（悉尼）	ap-southeast-2	是

区域名称	区域标识	SES 支持
亚太地区（东京）	ap-northeast-1	是
欧洲（法兰克福）	eu-central-1	是
欧洲地区（爱尔兰）	eu-west-1	是

Amazon SES 中的日志记录和监控

监控是维护 Amazon SES 和您的 AWS 解决方案的可靠性、可用性和性能的重要组成部分。 AWS 提供了帮助您监控 Amazon SES 和应对潜在事件的工具。

- Amazon 会实时 CloudWatch监控您的 AWS 资源和您运行 AWS 的应用程序。您可以收集和跟踪指标，创建自定义的控制平面，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。有关更多信息，请参阅正在从中检索 Amazon SES 事件数据 CloudWatch 和使用创建信誉监控警报 CloudWatch。
- AWS CloudTrail捕获由您或代表您发起的 API 调用和相关事件， AWS 账户 并将日志文件传输到您指定的 Amazon S3 存储桶。您可以识别哪些用户和帐户拨打了电话 AWS、发出呼叫的源 IP 地址以及呼叫发生的时间。有关更多信息，请参阅 使用记录 Amazon SES API 调用 AWS CloudTrail。
- Amazon SES 电子邮件发送事件可以帮助您优化电子邮件发送策略。Amazon SES 可捕获详细信息，包括发送数、送达数、打开数、单击数、退回邮件数、投诉数以及拒绝数。有关更多信息，请参阅 监控发送活动。
- Amazon SES 声誉指标可跟踪您账户的退回邮件和投诉率。有关更多信息，请参阅 监控发件人声誉。

使用记录 Amazon SES API 调用 AWS CloudTrail

Amazon SES 与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在 SES 中执行的操作的记录。 CloudTrail 将 SES 的 API 调用捕获为事件。捕获的调用包括来自 SES 控制台的调用和对 SES API 操作的代码调用。如果您创建跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括 SES 的事件。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向 SES 发出的请求、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，包括如何配置和启用它，请参阅《AWS CloudTrail 用户指南》。

SES 信息在 CloudTrail

CloudTrail 在您创建账户 AWS 账户 时已在您的账户上启用。当 SES 中出现支持的事件活动时，该活动将与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在中查看、搜索和下载最近发生的事件 AWS 账户。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您的 AWS 账户事件（包括 SES 的事件），请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅下列内容：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail日志文件

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根证书还是 AWS Identity and Access Management (IAM) 用户凭证发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

中的 SES 数据事件 CloudTrail

数据事件提供有关对在资源上或资源内执行的资源操作的信息。这些也称为数据层面操作。数据事件通常是高容量活动。默认情况下， CloudTrail 不记录数据事件。 CloudTrail 事件历史记录不记录数据事件。

记录数据事件将收取额外费用。有关 CloudTrail 定价的更多信息，请参阅AWS CloudTrail 定价。

> **Note**
>
> 通过 SES SMTP 接口发送电子邮件的活动不会记录到 CloudTrail 事件中。要获取全面的活动记录，请使用 SES API 参考和 SES API v2 参考 APIs 中的最新 SES。

下表列出了您可以记录数据事件的 SES 资源类型。数据事件类型（控制台）列显示要从控制 CloudTrail 台上的数据事件类型列表中选择的值。res ources.type 值列显示该`resources.type`值，该值是您在使用列配置高级事件选择器时指定的值，该列显示或。 AWS CLI CloudTrail APIs“ APIs 记录到的数据 CloudTrail” 列显示了 CloudTrail 针对该资源类型记录的 API 调用。

数据事件的 SES 资源类型

数据事件类型（控制台）	resources.type 值	数据 APIs 已记录到 CloudTrail
SES 身份	AWS:: SES:: EmailIdentity	SES: SendEmail SendRawEmail SendTemplatedEmail SendBulkTemplatedEmail SES v2： SendEmail SendBulkEmail
SES 配置集	AWS:: SES:: ConfigurationSet	SES: SendTemplatedEmail SendBulkTemplatedEmail SES v2： SendEmail SendBulkEmail
SES 模板	AWS:: SES:: Template	

以下示例说明如何使用`--advanced-event-selectors`参数记录所有 SES 电子邮件身份的所有数据事件：

```
aws cloudtrail put-event-selectors \
--region Region \
```

```
--trail-name TrailName \
--advanced-event-selectors
'[
    {
        "Name": "Log SES data plane actions for all email identities",
        "FieldSelectors": [
        { "Field": "eventCategory", "Equals": ["Data"] },
        { "Field": "resources.type", "Equals": ["AWS::SES::EmailIdentity"] }
       ]
    }
]'
```

您可以进一步细化高级事件选择器，以筛选`eventName`、`readOnly`、和`resources.ARN`字段，从而仅记录那些对您很重要的事件。有关这些字段的更多信息，请参阅《AWS CloudTrail API 参考》中的AdvancedFieldSelector。有关如何记录数据事件的更多示例，请参阅记录跟踪的数据事件。

CloudTrail SES 日志记录的日志传输方案

CloudTrail 根据账户和资源所有权、身份类型和地区等因素提供日志。以下矩阵根据这些因素的特定组合解释了将日志传送给谁和向何处传送。

场景类型	账户角色	资源	请求流程	日志传输
单一跨账户	账户 A：资源所有者 账户 B：请求者	电子邮件身份	B → A 的电子邮件身份	日志同时传送到 A 和 B
		反馈转发电子邮件	B → A 的反馈电子邮件	日志同时传送到 A 和 B
多个跨账户	账户 A：反馈电子邮件所有者 账户 B：电子邮件身份所有者 账户 C：请求者	反馈电子邮件 (A) 电子邮件身份 (B)	C → A 的反馈电子邮件 + B 的电子邮件身份	已传送到 A、B 和 C 的日志
全球终端节点（单一账户）	账户 A：所有者和申请者	全局终点（主终点：欧盟西部-1 和次要终点：us-west-2）	A → 全球终端节点	在处理请求的区域（e u-west-1 或 us-west- 2）

场景类型	账户角色	资源	请求流程	日志传输
				中交付给 A 的日志
全局终端节点（跨账户）	账户 A：电子邮件身份所有者 账户 B：请求者	电子邮件身份 (A) 全球端点 (B)（eu-west-1 和 us-west-2）	B → A 通过全局端点的电子邮件身份	发送给处理请求的区域（eu-west-1 或 us-west- 2）的 A 和 B 的日志

> **Note**
>
> - CloudTrail 始终将日志传送到请求者帐户。
> - 即使资源所有者没有执行操作，他们也会收到日志。
> - 对于全球终端节点，两个账户都需要在所有配置的区域进行 CloudTrail 订阅。
> - 在区域损伤期间，所有日志都显示在健康区域。

中的 SES 管理事件 CloudTrail

SES 将管理事件传递给 CloudTrail。管理事件包括与在您的内部创建和管理资源相关的操作 AWS 账户。在 Amazon SES 中，管理事件包括创建和删除身份或接收规则等操作。有关 SES API 操作的更多信息，请参阅 SES API 参考和 SES API v2 参考。

CloudTrail SES 的日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

以下示例演示了这些事件类型的 CloudTrail 日志：

事件类型

- DeleteIdentity
- VerifyEmailIdentity

- SendEmail 内容简单
- SendEmail 包含模板化内容

DeleteIdentity


```
{
  "Records":[
    {
        "eventVersion": "1.11",
        "userIdentity": {
            "type": "AssumedRole",
            "principalId": "AROA4DO2KAWIPZEXAMPLE:myUserName",
            "arn": "arn:aws:sts::111122223333:assumed-role/users/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "sessionContext": {
                "sessionIssuer": {
                    "type": "Role",
                    "principalId": "AROA4DO2KAWIPZEXAMPLE",
                    "arn": "arn:aws:iam::111122223333:role/admin-role",
                    "accountId": "111122223333",
                    "userName": "myUserName"
                },
                "attributes": {
                    "creationDate": "2025-02-27T09:53:35Z",
                    "mfaAuthenticated": "false"
                }
            }
        },
        "eventTime": "2025-02-27T09:54:31Z",
        "eventSource": "ses.amazonaws.com",
        "eventName": "DeleteIdentity",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "aws-cli/2.23.4",
        "requestParameters": {
            "identity": "sender@example.com"
        },
        "responseElements": null,
        "requestID": "50b87bfe-ab23-11e4-9106-5b36376f9d12",
        "eventID": "0ffa308d-1467-4259-8be3-c749753be325",
        "readOnly": false,
```

```
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management",
        "tlsDetails": {
            "tlsVersion": "TLSv1.3",
            "cipherSuite": "TLS_AES_128_GCM_SHA256",
            "clientProvidedHostHeader": "email.us-east-1.amazonaws.com"
        }
    }
  ]
}
```

VerifyEmailIdentity

```
{
  "Records":[
    {
        "eventVersion": "1.11",
        "userIdentity": {
            "type": "AssumedRole",
            "principalId": "AROA4DO2KAWIPZEXAMPLE:myUserName",
            "arn": "arn:aws:sts::111122223333:assumed-role/users/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "sessionContext": {
                "sessionIssuer": {
                    "type": "Role",
                    "principalId": "AROA4DO2KAWIPZEXAMPLE",
                    "arn": "arn:aws:iam::111122223333:role/admin-role",
                    "accountId": "111122223333",
                    "userName": "myUserName"
                },
                "attributes": {
                    "creationDate": "2025-02-27T09:53:35Z",
                    "mfaAuthenticated": "false"
                }
            }
        },
        "eventTime": "2025-02-27T09:56:20Z",
        "eventSource": "ses.amazonaws.com",
        "eventName": "VerifyEmailIdentity",
        "awsRegion": "us-east-1",
```

```
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "aws-cli/2.23.4",
        "requestParameters": {
            "emailAddress": "sender@example.com"
        },
        "responseElements": null,
        "requestID": "eb2ff803-ac09-11e4-8ff5-a56a3119e253",
        "eventID": "5613b0ff-d6c6-4526-9b53-a603a9231725",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management",
        "tlsDetails": {
            "tlsVersion": "TLSv1.3",
            "cipherSuite": "TLS_AES_128_GCM_SHA256",
            "clientProvidedHostHeader": "email.us-east-1.amazonaws.com"
        }
    }
  ]
}
```

SendEmail 内容简单

```
{
  "Records":[{
       "eventTime": "2025-01-24T11:43:00Z",
        "eventSource": "ses.amazonaws.com",
        "eventName": "SendEmail",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "aws-cli/2.23.4 md/awscrt#0.23.4",
        "requestParameters": {
            "destination": {
                "bccAddresses": ["HIDDEN_DUE_TO_SECURITY_REASONS"],
                "toAddresses": ["HIDDEN_DUE_TO_SECURITY_REASONS"],
                "ccAddresses": ["HIDDEN_DUE_TO_SECURITY_REASONS"]
            },
            "message": {
                "subject": {
                    "charset": "UTF-8",
                    "data": "HIDDEN_DUE_TO_SECURITY_REASONS"
                },
```

```
                "body": {
                    "html": {
                        "charset": "UTF-8",
                        "data": "HIDDEN_DUE_TO_SECURITY_REASONS"
                    },
                    "text": {
                        "charset": "UTF-8",
                        "data": "HIDDEN_DUE_TO_SECURITY_REASONS"
                    }
                }
            },
            "source": "sender@example.com"
        },
        "responseElements": null,
        "additionalEventData": {
            "sesMessageId": "01000100a11a11aa-00aa0a00-00a0-48a8-aaa7-
a174a83b456a-000000"
        },
        "requestID": "ab2cc803-ac09-11d7-8bb8-a56a3119e476",
        "eventID": "eb834e01-f168-435f-92c0-c36278378b6e",
        "readOnly": true,
        "resources": [{
            "accountId": "111122223333",
            "type": "AWS::SES::EmailIdentity",
            "ARN": "arn:aws:ses:us-east-1:111122223333:identity/sender@example.com"
        }],
        "eventType": "AwsApiCall",
        "managementEvent": false,
        "recipientAccountId": "111122223333",
        "eventCategory": "Data",
        "tlsDetails": {
            "tlsVersion": "TLSv1.3",
            "cipherSuite": "TLS_AES_128_GCM_SHA256",
            "clientProvidedHostHeader": "email.us-east-1.amazonaws.com"
        }
    }
  ]
}
```


SendEmail 包含模板化内容


```
{
        "eventVersion": "1.11",
```

```
        "userIdentity": {
            "type": "AssumedRole",
            "principalId": "AROA4DO2KAWIPZEXAMPLE:myUserName",
            "arn": "arn:aws:sts::111122223333:assumed-role/users/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "sessionContext": {
                "sessionIssuer": {
                    "type": "Role",
                    "principalId": "AROA4DO2KAWIPZEXAMPLE",
                    "arn": "arn:aws:iam::111122223333:role/admin-role",
                    "accountId": "111122223333",
                    "userName": "admin-role"
                },
                "attributes": {
                    "creationDate": "2025-03-05T18:51:06Z",
                    "mfaAuthenticated": "false"
                }
            }
        },
        "eventTime": "2025-03-05T19:16:29Z",
        "eventSource": "ses.amazonaws.com",
        "eventName": "SendEmail",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "aws-cli/2.23.4",
        "requestParameters": {
            "fromEmailAddress": "sender@example.com",
            "destination": {
                "toAddresses": ["HIDDEN_DUE_TO_SECURITY_REASONS"],
                "bccAddresses": ["HIDDEN_DUE_TO_SECURITY_REASONS"],
                "ccAddresses": ["HIDDEN_DUE_TO_SECURITY_REASONS"]
            },
            "emailTags": [{
                "value": "test",
                "name": "campaign"
            }, {
                "value": "cli-test",
                "name": "sender"
            }],
            "replyToAddresses": ["HIDDEN_DUE_TO_SECURITY_REASONS"],
            "content": {
                "template": {
                    "templateData": "HIDDEN_DUE_TO_SECURITY_REASONS",
```

```
                    "templateName": "TestTemplate"
                }
            }
        },
        "responseElements": null,
        "additionalEventData": {
            "sesMessageId": "01000100a11a11aa-00aa0a00-00a0-48a8-aaa7-
a174a83b456a-000000"
        },
        "requestID": "50b87bfe-ab23-11e4-9106-5b36376f9d12",
        "eventID": "0ffa308d-1467-4259-8be3-c749753be325",
        "readOnly": true,
        "resources": [{
            "accountId": "111122223333",
            "type": "AWS::SES::EmailIdentity",
            "ARN": "arn:aws:ses:us-east-1:111122223333:identity/sender@example.com"
        }, {
            "accountId": "111122223333",
            "type": "AWS::SES::Template",
            "ARN": "arn:aws:ses:us-east-1:111122223333:template/TestTemplate"
        }],
        "eventType": "AwsApiCall",
        "managementEvent": false,
        "recipientAccountId": "111122223333",
        "eventCategory": "Data",
        "tlsDetails": {
            "tlsVersion": "TLSv1.3",
            "cipherSuite": "TLS_AES_128_GCM_SHA256",
            "clientProvidedHostHeader": "email.us-east-1.amazonaws.com"
        }
    }
```


Amazon Simple Email Service 的合规性验证

作为多项合规计划的一部分，第三方审计师评估亚马逊简单电子邮件服务的安全 AWS 性和合规性。其中包括 SOC、PCI、FedRAMP、HIPAA 及其他。

有关特定合规计划范围内的 AWS 服务列表，请参阅合规计划范围内的AWS 服务按合规计划。有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 Amazon Simple Email Service 时的合规性责任取决于您数据的敏感度、贵公司的合规性目标以及适用的法律法规。 AWS 提供以下资源来帮助您满足合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了在上部署以安全性和合规性为重点的基准环境的步骤。 AWS
- AWS 合AWS 规资源 — 此工作簿和指南集可能适用于您的行业和所在地区。
- 使用AWS Config 开发人员指南中的规则评估资源 — AWS Config; 评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 此 AWS 服务可全面了解您的安全状态 AWS ，帮助您检查是否符合安全行业标准和最佳实践。

Amazon Simple Email Service 的弹性

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。各区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础结构相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅AWS 全球基础设施。

Amazon Simple Email Service 中的基础设施安全性

作为一项托管服务，Amazon 简单电子邮件服务受 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络访问亚马逊简单电子邮件服务。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

为 Amazon SES 设置 VPC 端点

很多 Amazon SES 客户制定了公司策略，以限制其内部系统连接到公有 Internet。这些策略禁止使用公有 Amazon SES 端点。

如果您有类似的策略，您可以使用 Amazon Virtual Private Cloud 以确保不超出这些限制。借助 Amazon VPC，您可以将 AWS 资源部署到位于隔离区域中的虚拟网络中 AWS Cloud。有关 Amazon VPC 的更多信息，请参阅《Amazon VPC 用户指南》。

您可以通过安全且可扩缩的方式，通过 VPC 端点直接从 Amazon VPC 连接到 SES。当您使用接口 VPC 端点时，它提供更好的安全态势，因为您无需打开出站流量防火墙，同时还提供使用 Amazon VPC 端点的其他好处。

使用 VPC 端点时，流向 SES 的流量不会通过互联网传输，也从不会离开 Amazon 网络，以便在不存在可用性风险或网络流量带宽限制的情况下安全地将您的 VPC 连接到 SES。您可以在多账户基础设施中集中管理 SES，并将其作为服务提供给您的账户，而无需使用互联网网关。

限制

- SES 在以下可用区中不支持 VPC 端点：`use1-az2`、`use1-az3`、`use1-az5`、`usw1-az2`、`usw2-az4`、`apne2-az4`、`cac1-az3` 和 `cac1-az4`。
- VPC 中使用的 SMTP 端点仅限于当前用于您账户的 AWS 区域 。

在 Amazon VPC 中设置 SES 的演练示例

先决条件

在完成本节中的过程之前，您必须完成以下步骤：

- 拥有现有的虚拟私有云（VPC）或创建新的 VPC。有关过程，请参阅开始使用 Amazon VPC。
- 在您的 VPC 中启动一个 Amazon EC2 实例，以测试与稍后步骤中创建的 VPC 终端节点的连接。有关更多信息，请参阅默认 VPCs。

Note

虽然 SES 的 VPC 终端节点可以与任何资源一起使用，但为了便于测试方法，本示例将使用 EC2 实例作为资源。由于亚马逊默认 EC2 限制通过端口 25 发送电子邮件的流量，因此您必

须使用 TCP 25 以外的其他端口，例如 TCP 465、587、2465 或 2587，有关更多信息，请参阅限制使用端口 25 发送电子邮件。

在 Amazon VPC 中设置 SES

设置与 SES 一起使用的 VPC 端点的过程包括几个单独的步骤。首先，您必须创建一个允许实例与 SMTP 端口通信的安全组，然后为 Amazon SES 创建 VPC 端点，最后，测试与 VPC 端点的连接以确保其配置正确。

步骤 1：创建安全组

在此步骤中，您将创建一个安全组，允许 Amazon EC2 实例与您将要创建的 VPC 接口终端节点进行通信。

创建安全组

1. 在 Amazon EC2 控制台的导航窗格中，在 “网络与安全” 下，选择 “安全组”。
2. 选择 Create security group（创建安全组）。
3. 在 Basic details (基本详细信息) 下面，执行以下操作：
   - 对于 Security group name (安全组名称)，输入标识安全组的唯一名称。
   - 对于 Description (描述)，输入一些描述安全组用途的文本。
   - 对于 VPC，选择要在其中使用 Amazon SES 的 VPC。
4. 在 Inbound rules (入站规则)下面，选择 Add rule (添加规则)。
5. 对于新的入站规则，请执行以下操作：
   - 对于类型，选择自定义 TCP。
   - 对于 Port range (端口范围)，输入要用于发送电子邮件的端口号。您可以使用以下任何端口号：**465**、**587**、**2465** 或 **2587**。
   - 对于 Source type (源类型)，选择 Custom (自定义)。
   - 对于源，输入私有 IP CIDR 范围或其他安全组 IDs ，其中包含将使用 VPC 终端节点与 SES 服务通信的资源。
   - （对您希望从中进行访问的每个 CIDR 范围或安全组重复步骤 4 - 5。）
6. 完成后，选择 Create security group (创建安全组)。

步骤 2：创建 VPC 端点

在 Amazon VP C 中，VPC 终端节点允许您将自己的 VPC 连接到支持的 AWS 服务。在本示例中，您将配置亚马逊 VPC，以便您的亚马逊 EC2 安全组可以连接到 Amazon SES。

创建 VPC 端点

1. 打开位于 https://console.aws.amazon.com/vpc/ 的 Amazon VPC 控制台。
2. 在 “PrivateLink 和莱迪思” 下，选择 “端点”。
3. 选择 Create Endpoint（创建端点），打开 Create Endpoint（创建端点）页面。
4. （可选）在 Endpoint settings（端点设置）面板中，在 Name tag（命名标签）字段中创建一个标签。
5. 对于服务类别，请选择 AWS 服务。
6. 在 Services（服务）面板中，在搜索栏中筛选 smtp，然后选择其单选按钮。
7. 在 VPC 面板中，在搜索栏内单击，然后从列表框中选择 VPC（参见the section called “先决条件”）。
8. 在子网面板中，选择可用区和子网 IDs。

   > **Note**
   >
   > Amazon SES 不支持以下可用性区域中的 VPC 端点：`use1-az2`、`use1-az3`、`use1-az5`、`usw1-az2`、`usw2-az4`、`apne2-az4`、`cac1-az3` 和 `cac1-az4`。

9. 在 Security groups（安全组）面板中，选择以前创建的安全组。
10. （可选）在标签面板中，可以创建一个或多个标签。
11. 选择创建端点。在 Amazon VPC 创建端点时，请等待大约 5 分钟。在端点可供使用时，Status（状态）列中的值将变为 Available（可用。

（可选）步骤 3：测试到 VPC 端点的连接

在完成配置 VPC 端点的过程后，您可以测试连接以确保正确配置了 VPC 端点。您可以使用大多数操作系统附带的命令行工具以测试连接。

测试到 VPC 端点的连接

1. 在您刚刚创建电子邮件-smtp VPC 终端节点的同一 VPC 中启动一个 Amazon EC2 实例。

有关连接到 Linux 实例的信息，请参阅亚马逊 EC2 用户指南中的连接到您的 Linux 实例。

有关连接到 Windows 实例的信息，请参阅亚马逊 EC2 用户指南中的入门教程。

2. 例如，使用 SES SMTP 接口发送测试电子邮件。

> **Note**
>
> 您必须先验证电子邮件地址或域，然后才能通过 Amazon SES 发送电子邮件。有关验证身份的更多信息，请参阅在 Amazon SES 中创建和验证身份。

Amazon SES 问题排查

本部分包含以下主题，可以帮助您在遇到问题：

- 有关可能遇到的域验证问题的信息，请参阅域和电子邮件地址验证问题。
- 有关 DKIM 相关问题的解决方案，请参阅Amazon SES 中的 DKIM 问题排查。
- 有关您在发送电子邮件时可能遇到的常见送达问题列表，以及您可以采取的纠正措施，请参阅Amazon SES 送达问题。
- 有关收件人在收到经由 Amazon SES 发送的电子邮件时可能看到的问题描述，请参阅从 Amazon SES 接收电子邮件时遇到的问题。
- 有关退回邮件、投诉和送达通知问题的解决方案，请参阅Amazon SES 通知问题。
- 有关使用 Amazon SES 发送电子邮件时可能发生的错误的列表，请参阅Amazon SES 电子邮件发送错误。
- 有关使用 API 或 SMTP 接口对 Amazon SES 进行多次调用时如何加快电子邮件发送速度的提示，请参阅增加 Amazon SES 吞吐量。
- 有关在通过 Amazon SES 的简单邮件传输协议（SMTP）接口使用它时可能遇到的常见问题的解决方案，以及 Amazon SES 返回的 SMTP 响应代码的列表，请参阅 Amazon SES SMTP 问题。
- 有关由 Amazon SES API v2 返回的常见错误代码的列表，请参阅常见错误。
- 有关与发送审核流程相关的常见问题以及如何处理这些问题的说明，请参阅Amazon SES 发送审核流程 FAQs。
- 有关基于 DNS 的黑洞名单 (DNSBLs) 如何影响您通过 Amazon SES 发送内容的讨论，请参阅。DNS 黑洞名单 (DNSBL) FAQs

如果您直接调用 Amazon SES API，请参阅 Amazon Simple Email Service API 参考，以了解您可能收到的 HTTP 错误。

> **Note**
>
> 如果您需要请求技术支持，请不要使用本开发者指南任何页面上的反馈链接，因为表单是由 AWS 文档团队而不是 AWS Support收到的。而是在联系我们页面上，浏览可用的不同支持选项。

内容

一般性 Amazon SES 问题

本页的信息将解释和帮助诊断在使用 Amazon SES 时可能遇到的问题。

我所做的更改不会立即可见

作为一项通过世界各地数据中心的计算机访问的服务，Amazon SES 使用一种称为最终一致性的分布式计算模型。您在 Amazon SES（或其他 AWS 服务）中所做的任何更改都需要一段时间才能从所有可能的终端节点中看到。它在服务器与服务器之间以及全球的区域与区域之间发送数据需要时间，这会造成一定的延迟。在大多数情况下，此延迟最多需要几分钟时间。

可能会注意到延迟的领域包括：

- 创建和修改配置集 – 当您创建或修改配置集时 (例如，如果您将专用 IP 地址池与现有配置集相关联)，您创建或修改配置集的时间与更改生效的时间之间可能会有简短的延迟。
- 创建和修改事件目的地 — 当您创建或修改事件目的地（例如，告诉 Amazon SES 将您的电子邮件发送到其他 AWS 服务）时，您创建或修改事件目标的时间与电子邮件发送事件实际到达指定目标的时间之间可能会有一段延迟。

域和电子邮件地址验证问题

要使用 Amazon SES 验证域或电子邮件地址，可使用 Amazon SES 控制台或 Amazon SES API 启动该过程。本节包含有可能帮助解决验证过程问题的信息。

> **Note**
>
> 在以下过程中，对 DNS 记录的引用可以指向 CNAME 记录或 TXT 记录，具体取决于您使用的 DKIM 的形式。Easy DKIM 使用 CNAME 记录，而自带 DKIM (BYODKIM) 使用 TXT 记录。我们针对每个 Easy DKIM 或 BYODKIM 提供了详细的验证程序。

常见的域验证问题

如果您尝试使用the section called “验证域身份”中的步骤验证域时遇到问题，请查看下面的可能原因和解决方案。

- 您正在尝试验证不归您所有的域 – 您无法验证不归自己所有的域。例如，如果您想通过 Amazon SES 从 gmail.com 域上的地址发送电子邮件，您需要专门验证该电子邮件地址。您无法验证整个 gmail.com 域。
- 您正在尝试验证私有域 – 如果 DNS 记录无法通过公共 DNS 解析，则无法验证域。
- DNS 提供商不允许 DNS 记录名称中有下划线 – 少数 DNS 提供商不允许在记录名称中包含下划线 (_)。但是，DKIM 记录名称中的下划线是必需的。如果您的 DNS 提供商不允许您在记录名称中输入下划线，请联系提供商的客户支持团队以获取帮助。
- DNS 提供商将域名附加到 DNS 记录的结尾 – 某些 DNS 提供商会自动将您的域名附加到 DNS 记录的属性名称中。例如，如果您创建一条属性名为 _domainkey.example.com 的记录，提供商可能会附加域名，最终的属性名称将为 _domainkey.example.com.example.com。要避免域名重复，请在输入 DNS 记录时向域名结尾添加句点。此步骤告知 DNS 提供商没有必要将域名附加到记录。
- 您的 DNS 提供商修改了 DNS 记录值 – 某些提供商会自动修改 DNS 记录值以仅使用小写字母。Amazon SES 仅在以下情况下才验证您的域：您的域检测到其属性值与您启动域验证流程时 Amazon SES 提供的值完全匹配的验证记录。如果域的 DNS 提供商将 DNS 记录值更改为仅使用小写字母，请与 DNS 提供商联系以获取更多帮助。
- 您想多次验证同一个域名 — 您可能需要多次验证您的域名，因为您在不同的地区发送邮件，或者因为您使用同一个域名从多个 AWS 账户发送邮件。如果 DNS 提供商不允许您拥有多条具有相同属性名称的 DNS 记录，您仍可以验证两个域。如果 DNS 提供商允许，您可以将多个属性值分配到同一条 DNS 记录。例如，如果 DNS 由 Amazon Route 53 管理，您可以完成以下步骤为同一条 CNAME 记录设置多个值：

  1. 在 Route 53 控制台中，选择在验证第一个区域中的域时创建的 CNAME 记录。
  2. 在 Value (值) 框中，转到现有属性值的末尾，然后按 Enter。
  3. 添加附加区域的属性值，然后保存记录集。

如果 DNS 提供商不允许为同一条 DNS 记录分配多个值，则可以在 DNS 记录属性名称中包含 _domainkey 来验证域一次，然后从属性名称中删除 _domainkey 并再验证一次。此解决方案的缺点是只能对同一个域验证两次。

检查域验证设置

您可以使用以下过程验证您的 Amazon SES 域验证 DNS 记录是否已正确地发布到您的 DNS 服务器。此过程使用 nslookup 工具，目前支持的平台有 Windows 和 Linux。在 Linux 上，您也可以使用 dig。

这些说明中的命令在 Windows 7 中执行，我们使用的示例域为 ses-example.com，这是通过使用 CNAME 记录的 Easy DKIM 配置的。

在此过程中，您首先要查找适用于您的域的 DNS 服务器，然后查询这些服务器以查看 CNAME 记录。您要查询为您的域名提供服务的 DNS 服务器，因为这些服务器包含的域 up-to-date信息最多，可能需要一段时间才能传播到其他 DNS 服务器。

验证您的域验证 CNAME 记录是否已发布到您的 DNS 服务器

1. 通过采取以下步骤查找您的域的名称服务器。

   a. 进入命令行。要进入 Windows 7 中的命令行，请选择 Start，然后键入 cmd。在基于 Linux 的操作系统中，打开终端窗口。

   b. 在命令提示符处，键入以下命令，其中 <domain> 是您的域。此操作将列出所有可用于您的域的名称服务器。

      ```
      nslookup -type=NS <domain>
      ```

      如果您的域为 ses-example.com，此命令将类似于：

      ```
      nslookup -type=NS ses-example.com
      ```

      命令的输出将列出可用于您的域的名称服务器。您将在下一步骤中查询这些服务器之一。

2. 通过执行以下步骤，验证 CNAME 记录是否已正确发布。请记住，Amazon SES 将生成三条 CNAME 记录以进行 Easy DKIM 身份验证，因此，需要分别对这三条记录重复以下过程。

   a. 在命令提示符处，键入以下命令，其中 <random string> 是 SES 生成的 CNAME 名称，<domain> 是您的域，<name server> 是您在步骤 1 中找到的其中一个名称服务器。

```
nslookup -type=CNAME <random string>_domainkey.<domain> <name server>
```

在我们的 ses-example.com 示例中，如果我们在步骤 1 中找到的名称服务器名为 ns1.name-server.net，并且 SES 生成的 <random string> 为 4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz，则我们将键入以下内容：

```
nslookup -type=CNAME 4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz_domainkey.ses-example.com
 ns1.name-server.net
```

b. 在命令的输出中，请验证 `canonical name =` 后的字符串是否与在 Amazon SES 控制台的身份列表中选择域时看到的 CNAME 值匹配。

在我们的示例中，我们正在 4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz_domainkey.ses-example.com 下寻找值为 4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz.dkim.amazonses.com 的 CNAME 记录。如果记录已正确发布，我们希望命令具有以下输出：

```
4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz_domainkey.ses-example.com canonical name =
 "4hzwn5lmznmmjyl2pqf2agr3uzzzzxyz.dkim.amazonses.com"
```

常见电子邮件验证问题

- 验证电子邮件未送达 – 如果完成验证电子邮件地址身份中的步骤，但在几分钟内没有收到验证电子邮件，请完成以下步骤：
  - 检查垃圾邮件或垃圾邮件文件夹中是否有要验证的电子邮件地址。
  - 确认您尝试验证的地址能够接收电子邮件。使用单独的电子邮件地址（例如您的个人电子邮件地址），将测试电子邮件发送到您要验证的地址。
  - 检查 Amazon SES 控制台中已验证地址的列表。确保您尝试验证的电子邮件地址没有任何错误。

Amazon SES 中的 DKIM 问题排查

本节列出了在 Amazon SES 中配置 DKIM 身份验证时可能会遇到的一些问题。如果您尝试设置 DKIM 并且遇到问题，请查看下面的可能原因和解决方法。

您已成功设置 DKIM，但是您的邮件未经过 DKIM 签名

如果您使用了 Easy DKIM 或 BYODKIM 为域配置 DKIM，但是您发送的邮件未经过 DKIM 签名，请执行以下操作：

- 请确保为适当的身份启用了 DKIM。要在 Amazon SES 控制台中为某个身份启用 DKIM，请在 Identities (身份) 列表中选择电子邮件域。在域的详细信息页面上，展开 DKIM，然后选择 Enable (启用) 以启用 DKIM。
- 确保您没有从同一域上的经过验证的电子邮件地址发送邮件。如果您已为某个域设置 DKIM，则从该域发送的所有邮件都经过 DKIM 签名，但您单独验证的电子邮件地址除外。单独验证的电子邮件地址使用单独的设置。例如，如果您已为域 example.com 配置 DKIM，并且您单独验证了电子邮件地址 mary@example.com（但没有为该地址配置 DKIM），则您从 mary@example.com 发送的电子邮件将在不经过 DKIM 身份验证的情况下发送。您可以通过从自己账户的身份列表中删除电子邮件地址身份，来解决此问题。
- 如果您在多个 AWS 区域使用相同的身份，则必须分别为每个区域配置 DKIM。同样，如果您使用具有多个账户的同一个域名，则必须为每个 AWS 账户配置 DKIM。如果您删除某个特定区域或账户所需的 DNS 记录，则 Amazon SES 将在该区域或账户中禁用 DKIM 签名。如果 DKIM 签名变为禁用状态，Amazon SES 会通过电子邮件向您发送通知。

在 Amazon SES 控制台中，域的 DKIM 详细信息显示：DKIM: waiting on sender verification...(DKIM: 正在等待发件人验证...) DKIM 验证状态：等待验证。

如果您完成 Easy DKIM 或 BYODKIM - 自带 DKIM 中的过程来为域配置 DKIM，但 Amazon SES 控制台仍指示 DKIM 验证处于等待状态，请执行以下操作：

- 等待最多 72 小时。在极少数情况下，DNS 记录可能需要一些时间才能对 Amazon SES 可见。
- 确认 CNAME 记录（对于 Easy DKIM）或 TXT 记录（对于 BYODKIM）使用了正确的名称。某些 DNS 提供商会自动将域名附加到您创建的记录。例如，如果您创建的记录中带有 `example._domainkey.example.com` 名称，则 DNS 提供商可能会将您的域名添加到此字符串的结尾，从而生成 `example._domainkey.example.com.example.com`。有关更多信息，请参阅您的 DNS 提供商的文档。

您收到一封来自 Amazon SES 的电子邮件，其中声明您的 DKIM 设置已被（或将被）撤销。

这意味着 Amazon SES 无法再在您的 DNS 服务器上找到所需的 CNAME 记录（如果您使用了 Easy DKIM）或所需的 TXT 记录（如果您使用了 BYODKIM）。通知电子邮件将告诉您在撤销 DKIM 设置状态和禁用 DKIM 签名之前必须重新发布 DNS 记录的时间。如果 DKIM 设置已撤销，您必须从头重新开始 DKIM 设置过程。

尝试设置 BYODKIM 时，DKIM 验证过程失败。

确保您的私有密钥使用正确的格式。私有密钥必须为 PKCS #1 或 PKCS #8 两种格式之一，并使用 1024 位或 2048 位 RSA 加密。此外，私有密钥必须采用 base64 编码。

设置 BYODKIM 时，当尝试为域指定公有密钥时，您收到 **`BadRequestException`** 错误。

如果收到 `BadRequestException` 错误，请执行以下操作：

- 确保为公有密钥指定的选择器至少包含 1 个字母数字字符并且小于等于 63 个字母数字字符。选择器不能包含句点，也不能包含其他符号或标点符号。
- 请确保已从公有密钥中删除页眉和页脚行，并且已从公有密钥中删除所有换行符。

在使用 Easy DKIM 时，您的 DNS 服务器成功返回了 Amazon SES DKIM CNAME 记录，但为域验证 TXT 记录返回了 **`SERVFAIL`**。

您的 DNS 提供商可能无法重定向 CNAME 记录。亚马逊 SES 并 ISPs查询 TXT 记录。为遵守 DKIM 规范，您的 DNS 服务器必须能够响应 TXT 记录查询以及 CNAME 记录查询。如果您的 DNS 提供商无法响应 TXT 记录查询，替代方法是使用 Route 53 作为您的 DNS 托管提供商。

您的电子邮件包含两个 DKIM 签名

包含 `d=amazonses.com` 的额外 DKIM 签名是 Amazon SES 自动添加的。您可以忽略它。

Amazon SES 送达问题

在您成功向 Amazon SES 发送请求后，您的邮件通常会立即发送。有时候可能有短暂延迟。在任何情况下，您都可以确信您的电子邮件将发送。

但是，当 Amazon SES 发送您的邮件时，有几个因素可能阻止其成功发送，并在某些情况下，您将仅在发送的邮件未送达时才知道送达失败。要解决此情况，请使用以下过程。

如果未收到电子邮件，请尝试以下操作：

- 确认您已对有问题的电子邮件发出 `SendEmail` 或 `SendRawEmail` 请求，并且已收到成功响应。如果您以编程方式发出这些请求，请检查您的软件日志，以确保该程序已发出请求并收到成功响应。
- 阅读博客文章通过 SES 发送电子邮件时可能出现延迟的三个位置，因为该问题可能实际上只是延迟而不是传送失败。
- 请检查发件人的电子邮件地址 (“From”地址) 以验证其是否有效。此外，请检查 Return-Path 地址，此地址是发送退回邮件的地址。如果您的邮件被退回，则将提供一条说明性错误消息。

- 检查 AWS Service Health Dashboard，以确认 Amazon SES 不存在已知问题。
- 联系电子邮件收件人或收件人的 ISP。验证收件人使用正确的电子邮件地址，并询问收件人的 ISP 是否存在任何已知的送达问题。此外，确定电子邮件是否确实已到达，而不是作为垃圾邮件被筛选。
- 如果您已注册付费的 AWS 支持 方案，您可以创建新的技术支持案例。在您与我们的通信中，请提供任何相关的收件人地址，以及从 IDs或回SendRawEmail复中 IDs 返回的任何请求SendEmail或消息。
- 请稍等片刻，以确定该问题是实际延迟，而不是永久的传输失败。为了打击垃圾邮件发送者，有些人会 ISPs 暂时拒绝来自未知发送邮件服务器的传入邮件。此过程称为灰名单，可能会导致送达延迟。Amazon SES 将重新发送这些消息。如果问题原因是灰名单，ISP 可能会在这些重试中接受电子邮件。
- 即使您已从客户的最佳利益考虑，仍可能遇到影响消息送达能力的情况。请参阅the section called “保持良好的发件人声誉”，以帮助确保您的电子邮件通信成功送达目标受众。

从 Amazon SES 接收电子邮件时遇到的问题

本节介绍了您在收到发送自 Amazon SES 的电子邮件时，可能会遇到的一些常见问题。

电子邮件客户端显示“通过 amazonses.com 发送”作为电子邮件的来源

当发件人的域与发送电子邮件的域不匹配时，某些电子邮件客户端会显示“通过”域（在本示例中为 amazonses.com）。有关更多信息，请参阅 Gmail 支持网站上的发件人姓名旁边的额外信息。或者，您可以设置DomainKeys 识别邮件 (DKIM)。使用 DKIM 验证您的电子邮件时，电子邮件客户端通常不会显示“通过”，因为 DKIM 签名会显示该电子邮件来自声明的所属域。有关设置 DKIM 的信息，请参阅在 Amazon SES 中使用 DKIM 对电子邮件进行身份验证。

> **Note**
>
> 如果您收到来自 SES 用户的垃圾邮件或其他未经请求的电子邮件，请使用电子邮件客户端中的垃圾邮件举报工具，并按照联系我们下列出的步骤举报 SES 电子邮件滥用行为。

邮件包含乱码或无意义的字符

如果您的邮件包含不在 ASCII 字符集中的字符（例如带重音的拉丁字符、中文字符或阿拉伯字符），则必须使用 HTML 字符编码方式对这些字符进行编码。可以使用基于 Web 的工具对您的电子邮件中的字符进行编码，例如 Email On Acid 网站上的 HTML 字符转换器。

此外，您可以自行组装 MIME 消息。在 MIME 邮件中，您可以指定邮件应使用 UTF-8 编码。使用 UTF-8 编码时，可以直接在您的邮件中使用非 ASCII 字符。创建 MIME 消息后，可以使用 SendRawEmailAPI 或 API v2 进行发送。SendMail

此问题的一个常见原因是 Microsoft Word 的智能引号功能。如果您经常从 Word 复制内容并粘贴到您的电子邮件中，可能会遇到此问题。智能引号功能会将直引号字符（"..."）替换为弯引号字符（“...”）。弯引号字符不是标准的 ASCII 字符。因此，它们可能会在某些电子邮件客户端中显示为“??”，或显示为一组字符，例如“â€œ”。要更正此问题，您可以禁用 Word 中的智能引号功能。或者，您可以使用上一段中的 SendRawEmail 解决方案。要了解如何禁用此功能，请参阅 Microsoft Office 支持网站上的 Word 中的智能引号。

Amazon SES 通知问题

如果您遇到与退回邮件、投诉或送达通知相关的问题，请查看下面的可能原因和解决方案。

- 您通过 Amazon SNS 收到退回邮件通知，但不知道通知对应哪些收件人 – 将来，要将退回邮件通知与给定收件人相关联，您有以下选项：
  - 由于 Amazon SES 不会保留您添加的任何自定义消息 IDs ，因此请存储标识符与 Amazon SES 在接受电子邮件时传回给您的亚马逊 SES 消息编号之间的映射。
  - 每次调用 Amazon SES 只向单个收件人发送电子邮件，而不是向多个收件人发送一封电子邮件。
  - 您可以启用通过电子邮件进行反馈转发，这会将完整的退回邮件转发给您。
- 您通过 Amazon SNS 或电子邮件反馈转发收到投诉或送达通知，但您不知道这些通知对应于哪些收件人——有些 ISPs 人会在将投诉通知传递给 Amazon SES 之前先删除投诉收件人的电子邮件地址。为使您能够找到收件人的电子邮件地址，最好是存储标识符与 Amazon SES 在接受电子邮件时传回的 Amazon SES 邮件 ID 之间的映射。请注意，Amazon SES 不会保留您添加的任何自定义消息 IDs 。
- 您要设置通知以进入不属于您的 Amazon SNS 主题 – 该主题的所有者必须配置允许您的账户在他们的主题中调用 `SNS:Publish` 操作的 Amazon SNS 访问策略。有关如何使用 IAM 策略控制对 Amazon SNS 主题的访问权限的信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的管理对 Amazon SNS 主题的访问权限。

Amazon SES 电子邮件发送错误

本主题介绍您在通过 Amazon SES 发送电子邮件时可能遇到的各种类型的电子邮件发送错误。如果您尝试通过 Amazon SES 发送电子邮件且调用 Amazon SES 失败，Amazon SES 将向您的应用程序返回错误消息，并且不会发送电子邮件。您看到此错误消息的方式取决于您调用 Amazon SES 的方式。

- 如果您直接调用 Amazon SES API，查询操作将返回错误。该错误可能是 `MessageRejected` 或 Amazon Simple Email Service API 参考的常见错误主题中指定的错误之一。
- 如果您使用使用支持异常的编程语言的 AWS 软件开发工具包调用 Amazon SES，Amazon SES 可能会抛出异常。异常的类型取决于软件开发工具包和错误。例如，例外可能是 Amazon SES`MessageRejectedException`（实际名称可能因软件开发工具包而异），也可能是一般 AWS 异常。无论是哪种类型的异常，异常中的错误类型和错误消息将为您提供更多信息。
- 如果您通过 SMTP 接口调用 Amazon SES，您遇到错误的方式取决于应用程序。某些应用程序可能会显示特定的错误消息，而其他应用程序则可能不会显示。有关 Amazon SES 返回的 SMTP 响应代码的列表，请参阅由 Amazon SES 返回的 SMTP 响应代码。

> **Note**
>
> 当您调用 Amazon SES 发送电子邮件失败时，您无需为该电子邮件付费。

当您尝试发送电子邮件时，可能会导致 Amazon SES 返回错误的 Amazon SES 特定问题的类型如下。这些错误是对一般 AWS 错误的补充，`MalformedQueryString`例如在《亚马逊简单电子邮件服务 API 参考》的 “常见错误” 主题中指出的错误。

- 电子邮件地址未经验证。以下身份无法签入区域 region：identity1、identity2、identity3 – 您正在尝试从未通过 Amazon SES 验证的电子邮件地址或域发送电子邮件。此错误可能发生于“From”、“Source”、“Sender”或“Return-Path”地址。如果您的账户仍位于 Amazon SES 沙盒中，您还必须验证每个收件人电子邮件地址，但 Amazon SES 邮箱模拟器提供的收件人除外。如果 Amazon SES 无法显示所有失败的身份，错误消息将以省略号结束。

  > **Note**
  >
  > Amazon SES 有多个终端节点 AWS 区域，每个终端节点的电子邮件地址验证状态是不同的 AWS 区域。您必须完成要使用的每个发件人的验证过程。 AWS 区域

- Account is paused (账户已暂停) – 已暂停您的账户发送电子邮件的功能。您仍然可以访问 Amazon SES 控制台并执行大多数操作。但是，如果您尝试发送电子邮件，则会收到此消息。

  如果我们暂停了您的账户发送电子邮件的功能，将自动将通知发送到与您的 AWS 账户关联的电子邮件地址。有关更多信息，请参阅 the section called “发送评论流程 FAQs”。
- Throttling (限制) – 您的应用程序可能尝试每秒发送了过多的邮件，或者您可能在过去 24 小时内发送了过多的电子邮件。在这些情况下，错误消息可能类似于以下示例：

- Daily message quota exceeded (已超出每日邮件发送配额) – 您已超出在 24 小时内允许发送的最大邮件数量。如果您已超出每日配额，您必须等到下一个 24 小时时段内，然后才能发送更多电子邮件。
- Maximum sending rate exceeded (已超出最大发送速率) – 您尝试每秒发送的电子邮件数已超过允许的最大发送速率。如果您已超出发送速率，您可以继续发送电子邮件，但需要降低发送速率。有关更多信息，请参阅 AWS 消息和定位博客上的 “如何处理 “限制-已超过最大发送速率” 错误。
- 已超出 Sigv2 SMTP 最大发送速率-您正在尝试使用 2019 年 1 月 10 日之前创建的 SMTP 凭据发送邮件；您的 SMTP 凭据是使用旧版本的签名创建的。 AWS 出于安全考虑，您应删除在此日期之前创建的凭证，并将其替换为较新的凭证。您可以使用 IAM 控制台删除较早的凭证。有关创建凭证的更多信息，请参阅 the section called “获取 SMTP 凭证”。

您应定期监视您的发送活动，以了解您离发送配额还有多远。有关更多信息，请参阅 监控您的 Amazon SES 发送配额。有关发送配额的基本信息，请参阅管理您的 Amazon SES 发送限制。有关如何提高发送配额的信息，请参阅提升您的 Amazon SES 发送配额。

> **Important**
>
> 如果说明限制错误的错误文本与您超出每日配额或最大发送速率无关，则可能存在导致降低发送功能的系统级问题。有关服务状态的信息，请转至 AWS Service Health Dashboard。

- 未指定收件人 – 没有提供收件人。
- There are non-ASCII characters in the email address (电子邮件地址中有非 ASCII 字符) – 电子邮件地址字符串必须是 7 位 ASCII 字符。如果您希望向或从某个地址的域部分中包含 Unicode 字符的电子邮件地址发送邮件，则必须使用 Punycode 对域进行编码。不允许在电子邮件地址的本地部分 (@ 符号前面的部分) 中使用 Punycode，也不允许在“易记发件人”名称中使用。如果您想要在“易记发件人”名称中使用 Unicode 字符，您必须使用 MIME encoded-word 语法编码“易记发件人”名称，如使用 Amazon SES API v2 发送原始电子邮件中所述。有关 Punycode 的更多信息，请参阅 RFC 3492。
- Mail FROM domain is not verified (发件人域未验证) – Amazon SES 无法读取使用指定 MAIL FROM 域所需的 MX 记录。有关设置自定义 MAIL FROM 域的信息，请参阅 使用自定义 MAIL FROM 域。
- Configuration set does not exist (配置集不存在) – 您指定的配置集不存在。配置集是可选参数，您可以使用它来发布电子邮件发送事件。有关更多信息，请参阅 使用 Amazon SES 事件发布监控电子邮件发送。

增加 Amazon SES 吞吐量

在发送电子邮件时，您调用 Amazon SES 的频率可以等于允许的最大发送速率。(有关最大发送速率的更多信息，请参阅管理您的 Amazon SES 发送限制。) 但是，每次调用 Amazon SES 需要花费一定时间才能完成。

如果使用 Amazon SES API 或 SMTP 接口进行多次 Amazon SES 调用，可能需要考虑以下提示，以帮助提高吞吐量：

- 测量当前性能以确定瓶颈 – 性能测试的方法包括在应用程序的代码循环中以尽可能快的速度发送多个测试电子邮件。策略每个 `SendEmail` 请求的往返操作延迟。然后，以增量方式在同一台计算机上启动应用程序的其他实例，并监视是否对网络延迟有影响。您可能还需要在多台计算机上和不同网络中运行此测试，以帮助查明可能存在的任何机器资源瓶颈或网络瓶颈。
- （仅 API）考虑使用持久 HTTP 连接 – 使用持久 HTTP 连接，而不是为每个 API 请求建立单独的新 HTTP 连接并因此产生开销。也就是说，为多个 API 请求重复使用相同的 HTTP 连接。
- 考虑使用多个线程 – 在应用程序使用单个线程时，应用程序代码调用 Amazon SES API，然后同步等待 API 响应。发送电子邮件通常是 I/O 密集型操作，使用多个线程工作可提供更高的吞吐量。您可以根据需要使用多个线程并发执行发送。
- 考虑使用多个进程 – 使用多个进程可帮助提高吞吐量，因为您将获得更多与 Amazon SES 的并发活跃连接。例如，您可以将预期的电子邮件分配到多个存储桶，然后同时运行电子邮件发送脚本的多个实例。
- 考虑使用本地邮件中继 – 您的应用程序可以快速将邮件传送到本地邮件服务器，这可以帮助缓冲该邮件并将它们异步传送给 Amazon SES。有些邮件服务器支持送达并发，这意味着即使您的应用程序以单线程的方式生成传送至邮件服务器的电子邮件，邮件服务器在发送至 Amazon SES 时也将使用多个线程。有关更多信息，请参阅 将 Amazon SES 与您的现有电子邮件服务器集成。
- 考虑将您的应用程序托管在靠近 Amazon SES API 终端节点的地方 — 您不妨考虑将您的应用程序托管在靠近 Amazon SES API 终端节点的数据中心，或者托管在与 Amazon SES API 终端节点相同 AWS 区域的亚马逊 EC2 实例上。这有助于减少您的应用程序和 Amazon SES 之间的网络延迟并提高吞吐量。有关已推出 Amazon SES 的区域的列表，请参阅《AWS 一般参考》中的 Amazon Simple Email Service（Amazon SES）。
- 考虑使用多台计算机 – 根据主机的系统配置，与单个 IP 地址的同步 HTTP 连接数量可能会有限制，一旦您在单个计算机上超过特定的并发连接数，就会限制并行机制的优势。如果这是瓶颈，您可能需要考虑使用多台计算机进行并发 Amazon SES 请求。
- 考虑使用 Amazon SES 查询 API 代替 SMTP 终端节点 – 使用 Amazon SES 查询 API 让您可以使用单次网络调用提交电子邮件发送请求，而与 SMTP 终端节点相连接涉及包含多个网络请求（例如

EHLO、MAIL FROM、RCPT TO、DATA、QUIT）的 SMTP 对话。有关 Amazon SES 查询 API 的更多信息，请参阅使用 Amazon SES API 发送电子邮件。

- 使用 Amazon SES 邮箱模拟器测试您的最高吞吐量 – 您可以使用邮箱模拟器测试您实施的任何更改。邮箱模拟器可帮助您确定系统的最大吞吐量，并且不会用完您的日发送配额。有关邮箱模拟器的信息，请参阅手动使用邮箱模拟器。

如果您通过 Amazon SES 的 SMTP 接口访问它，请参阅Amazon SES SMTP 问题，以了解可能会影响吞吐量的特定 SMTP 相关问题。

Amazon SES SMTP 问题

本节包含与通过 Amazon SES 简单邮件传输协议（SMTP）接口发送电子邮件相关的几个常见问题的解决方案。它还包含 Amazon SES 返回的 SMTP 响应代码列表。

要了解有关通过 Amazon SES SMTP 接口发送电子邮件的更多信息，请参阅使用 Amazon SES SMTP 接口发送电子邮件。

- 无法连接到 Amazon SES SMTP 端点。

  Amazon SES SMTP 端点连接问题通常与以下问题有关：

  - 凭据不正确-用于连接到 SMTP 终端节点的凭据与您的 AWS 凭据不同。要获取 SMTP 凭证，请参阅 获取 Amazon SES SMTP 凭证。有关凭证的更多信息，请参阅Amazon SES 凭证的类型。
  - 网络或防火墙问题 – 您的网络可能会在您尝试发送电子邮件时使用的端口上阻止出站连接。要确定本地网络上是否有问题导致出现连接问题，请在命令行中键入以下命令，并将 *port* 替换为您尝试使用的端口（通常为 465、587、2465 或 2587）：`telnet email-smtp.us-west-2.amazonaws.com port`

    如果您能够使用此命令连接 SMTP 服务器，并且您尝试使用 TLS Wrapper 或 STARTTLS连接 Amazon SES，请完成使用命令行来测试与 Amazon SES SMTP 接口的连接中所述的过程。

    如果您无法使用 `telnet` 或 `openssl` 连接到 Amazon SES SMTP 端点，则表明您的网络在您尝试使用的端口上阻止了出站连接（例如，有防火墙）。请向您的网络管理员咨询，以诊断和解决该问题。

- 您正在使用端口 25 从亚马逊 EC2 实例向 Amazon SES 发送数据，但收到了超时错误。

  亚马逊默认 EC2 限制端口 25。要取消这些限制，请提交 A mazon 取消电子邮件发送限制的 EC2 请求。您也可以使用端口 465 或 587（这两个端口不受限制）连接到 Amazon SES。

- 网络错误导致电子邮件丢失。

  确保您的应用程序在连接到 Amazon SES SMTP 端点时使用重试逻辑，且您的应用程序可以检测到网络错误，并在出错时重新尝试邮件传输。SMTP 是一个详细的协议，使用此协议发送电子邮件需要几次网络往返操作。由于 SMTP 的性质，潜在的网络错误增加。

- 您失去与 SMTP 端点的连接。

  丢失连接通常由以下问题引起：

  - MTU 大小 – 如果您收到超时错误消息，则在用于连接到 Amazon SES SMTP 接口的计算机的网络接口上，最大传输单位（MTU）可能过大。要解决此问题，请将此计算机的 MTU 大小设置为 1500 字节。

    有关在 Windows、Linux 和 macOS 操作系统上设置 MTU 大小的更多信息，请参阅《Amazon Redshift 管理指南》中的查询似乎在客户端挂起且未到达集群。

    有关为亚马逊实例设置 MTU 大小的更多信息，请参阅亚马逊 EC2 用户指南中的您的 EC2 EC2 实例的网络最大传输单位 (MTU)。

  - 长寿命连接 — Amazon SES SMTP 终端节点在弹性负载均衡器 (ELB) 后面的亚马逊 EC2 实例队列上运行。为了确保系统具有容错能力，活动的 up-to-date Amazon EC2 实例会定期终止并替换为新实例。由于您的应用程序通过 ELB 连接到 Amazon EC2 实例，因此当该亚马逊 EC2 实例终止时，该连接将失效。您应该在通过单一 SMTP 连接传输固定数量的消息后，或者该 SMTP 连接已活动特定时间后，建立新的 SMTP 连接。您需要通过实验才能找到适当的阈值，具体取决于应用程序的托管位置以及向 Amazon SES 提交电子邮件的方式。

- 您需要知道 Amazon SES SMTP 邮件服务器的 IP 地址，以便将此 IP 地址加入网络允许列表。

  Amazon SES SMTP 端点的 IP 地址位于负载均衡器之后。因此，这些 IP 地址会频繁更改。所以，我们无法提供 Amazon SES 端点各个 IP 地址的最终列表。我们建议您将 `amazonses.com` 域列入允许列表，而不是将单个 IP 地址列入允许列表。

由 Amazon SES 返回的 SMTP 响应代码

此部分包含 Amazon SES SMTP 接口返回的响应代码的列表。

您应重试收到 400 错误的 SMTP 请求。我们建议您实现一个重试请求的系统，该系统的等待时间逐渐变长（例如，在重试前等待 5 秒，然后等待 10 秒，接下去等待 30 秒）。如果第三次重试没有成功，请等待 20 分钟，然后重复此过程。要查看使用指数重试策略的实现示例，请参阅 AWS 消息收发和目标博客上的How to handle a "Throttling - Maximum sending rate exceeded" error。

> **Note**
>
> AWS SDKs 自动实现重试逻辑，但他们使用 HTTPS 接口而不是 SMTP。

如果您收到 500 错误，则必须先修改您的请求以更正问题，然后重新提交请求。例如，如果您的 AWS 身份验证凭证无效，则必须更新您的应用程序以使用正确的凭证，然后才能再次提交请求。

描述	响应代码	更多信息
身份验证成功	`235 Authentication successful`	您的 SMTP 客户端已成功连接并登录到 SMTP 服务器。
已成功传输	`250 Ok MessageID`	*MessageID* 是 Amazon SES 用来识别消息的唯一字符串。
服务不可用	`421 Too many concurrent SMTP connections`	Amazon SES 无法处理请求，因为当前与 SMTP 服务器的连接过多。
本地处理错误	`451 Temporary service failure`	Amazon SES 无法处理请求。请求可能存在阻止它被处理的问题。
超时	`451 Timeout waiting for data from client`	请求之间间隔的时间太长，因此 SMTP 服务器已关闭连接。
超出每日发送配额	`454 Throttling failure: Daily message quota exceeded`	您已超出 Amazon SES 允许您在 24 小时内发送的电子邮件的最大数量。有关更多信息，请参阅 管理您的 Amazon SES 发送限制。
超出最大发送速率	`454 Throttling failure: Maximum sending rate exceeded`	您已超出 Amazon SES 允许您每秒发送的电子邮件的最大数量。有关更多信息，请参阅 管理您的 Amazon SES 发送限制。
验证 SMTP 凭证时的 Amazon SES 问题	`454 Temporary authentication failure`	可能导致此问题的问题包括（但不限于）：

描述	响应代码	更多信息
		• 电子邮件发送应用程序和 Amazon SES 之间的加密有问题。请注意，连接到 Amazon SES 时必须使用加密连接。有关更多信息，请参阅 连接到 Amazon SES SMTP 端点。 • Amazon SES 可能会遇到问题。有关更新，请查看 AWS Service Health Dashboard。
接收请求的问题	`454 Temporary service failure`	Amazon SES 未成功接收请求。因此，邮件未发送。
凭证错误	`530 Authentication required`	您用于发送电子邮件的应用程序在连接到 Amazon SES SMTP 接口时未尝试进行身份验证。
身份验证凭证无效	`535 Authentication Credentials Invalid`	您用于发送电子邮件的应用程序未向 Amazon SES 提供正确的 SMTP 凭证。请注意，您的 SMTP 凭据与您的 AWS 凭据不同。有关更多信息，请参阅 获取 Amazon SES SMTP 凭证。
账户未订阅 Amazon SES	`535 Account not subscribed to SES`	拥有 SMTP 凭证的用户尚未注册 Amazon SES。 AWS 账户
邮件太长	`552 Message is too long.`	您尝试发送的邮件大小大于最大邮件大小。
账户未订阅 Amazon SES	`535 Account not subscribed to SES`	拥有 SMTP 凭证的用户尚未注册 Amazon SES。 AWS 账户

描述	响应代码	更多信息
MAIL FROM 语法错误	`553 <email-address > Invalid email address`	SMTP 邮件的 MAIL FROM 部分存在语法错误。请检查您是否遵循正确的格式，不要忘记在“<>”中附上电子邮件地址。
RCPT TO 语法错误	`553 <email-address > address unknown`	SMTP 消息的 RCPT TO 部分存在语法错误。请检查您是否遵循正确的格式，不要忘记在“<>”中附上电子邮件地址。
用户没有权限来调用 Amazon SES SMTP 端点	`554 Access denied: User UserARN is not authorized to perform ses:SendRawEmail on resource IdentityARN`	拥有 SMTP 证书的用户的 AWS Identity and Access Management (IAM) 策略或 Amazon SES 发送授权策略不允许调用 Amazon SES SMTP 终端节点。

描述	响应代码	更多信息
未经验证的电子邮件地址	`554 Message rejected: Email address is not verified. The following identities failed the check in region` *`region`*`: `*`identity0`*` , `*`identity1`*` , `*`identity2`*	您正在尝试通过未验证是否可通过 Amazon SES 账户发送电子邮件的电子邮件地址或域发送电子邮件。此错误可能适用于“发件人”、“源”、“发送者”或“退回路径”地址。如果您的账户仍在沙盒中，则还必须验证每个收件人电子邮件地址（Amazon SES 邮箱模拟器提供的收件人除外）。如果 Amazon SES 无法显示未通过验证检查的所有身份，则错误消息将以三个句点（...）结束。 **Note** Amazon SES 有多个终端节点 AWS 区域，每个终端节点的电子邮件地址验证状态各不相同 AWS 区域。您必须为要使用的每个发件人完成验证过程。 AWS 区域

Note

对于本页上的故障排除未能解决的 SMTP 问题，请尝试联系我们下列出的 SES 支持选项。

Amazon SES 常见问题解答 (FAQs)

本节包含与使用 Amazon SES 相关的几个常见问题的解答。

本节包含 FAQs 以下主题：

- 专用 IP 地址（托管） FAQs
- Amazon SES 发送审核流程 FAQs
- DNS 黑洞名单 (DNSBL) FAQs
- Amazon SES 电子邮件发送指标 FAQs

专用 IP 地址（托管） FAQs

虽然为专用 IP 管理、扩展和预热专用 IP 地址（托管式）提供了许多自动化功能，但人们对这种自动化的程度和 SES 的职责存在一些误解。假设 “托管” 意味着 SES 可以完全处理 IP 信誉和列表问题的各个方面，这是不正确的。为了澄清这些误解，我们需要强调的是，虽然该服务可以自动执行扩展和预热等技术方面，但您仍有责任维护发送声誉并管理任何与声誉相关的问题，例如被列入信誉屏蔽列表 (RBL)。

它们 FAQs 解决了对该功能范围的常见误解，并阐明了您和SES之间的责任共担模式。这些常见问题解答强调，尽管 “托管” 方面指的是技术基础设施管理，但您仍必须积极监控和维护您的发送声誉，保持较低的退回率，并自己处理大多数 RBL 下架请求。

问题 1：我能否让 SES 从 RBL 中列出的专用 IP 地址（托管）中删除？

如果您的专用 IP 地址（托管）列在收件人邮件提供商的任何 RBL 中，则不是 SES 的责任，您必须直接向 RBL 管理员申请删除。通过跟踪退回通知和 SMTP 响应消息来识别屏蔽，监控您的专用 IPs（托管）非常重要。这种监控有助于保护您的电子邮件发送声誉，并允许您快速处理可能发生的任何 RBL 事件，从而确保稳定的电子邮件送达率。

问题 2：我能否让 SES 分配一个新的专用 IP 地址（托管）来替换 Spamhaus 未提供的 RBL 中列出的当前专用 IP 地址？

不是。 SES 不会轮换专用 IP 地址。由于您要对自己的专用 IP 地址（托管地址或标准地址）负责，因此您需要弄清楚它们被列入 RBL 的原因，然后自己将其除名。

问题 3：SES 能否监控分配给我的账户的专用 IP 地址（托管）的跳出率，并在跳出率变高时轮换地址？

不是。 当账户出现高跳出率时，SES 不会轮换专用 IP 地址。当您租用专用 IP 地址（托管）时，只有您的账户才拥有通过该专用 IP 地址发送电子邮件的专有权利，因此 SES 无法监控您的账户。您有责任管理您的发件人声誉和电子邮件组件，包括管理投诉和将退回率保持在2%以下。

问题 4：我刚刚租了一个专用 IP 地址（托管），但由于地址在 RBL 上，通过它发送的电子邮件被退回了。在将专用 IP 地址（托管）出租给账户之前，SES 是否会检查其信誉？

是。在将任何专用 IP 地址（托管）租给 SES 账户之前，SES 会将其重置（30 天）。声誉通常会重置为大多数主要提供商。SES 确保该地址不在任何 RBL 上，例如 Spamhaus；但是，SES 不会监控所有 RBLs 可用的，例如较小的区域性。 RBLs如果您因关注B2B或区域提供商而对使用这些域名的RBL有任何疑问，则需要自己查看专用 IP 地址（托管）的信誉状态。

问题 5：如果 SES 在除垃圾邮件之外的 RBL 中列出专用 IP 地址（托管）时不采取行动，我为什么要使用它们？

专用 IP 地址（托管）是通过根据流量添加和删除 IP 地址来自动缩放来管理的。此外，它还为您节省了每个 ISP 的预热管理时间；因此，托管池可以根据历史发送模式跟踪每个 IP 地址可以发送多少电子邮件。更多好处可以在中找到the section called “优点和特点”。

问题 6：如何追踪租给我账户的专用 IP 地址（托管）？

您可以将 SES 配置集与为亚马逊 Data Firehose 或 Amazon SNS 主题定义的事件发布目标一起使用。SES 交付事件包括标签`ses:outgoing-ip`。因此，如果电子邮件因专用 IP 地址（托管）的信誉而被退回，则可以在退回事件的标签中找到违规的专用 IP 地址（托管）。`ses:outgoing-ip`

Amazon SES 发送审核流程 FAQs

我们监控通过 Amazon SES 发送的电子邮件，以确保此服务不被用于传递恶意、未经请求或低质量的电子邮件。如果我们确定用户正在发送属于这些类别之一的内容，我们将对此用户账户采取措施。我们将此流程称为发送审核流程。

在许多情况下，当检测到账户存在问题时，我们将对账户进行审核。在另一些情况下，我们会暂停账户发送电子邮件的功能。我们采取这些措施保护每个账户的发件人声誉，避免其他 SES 用户遇到服务中断和送达率问题。

内容

账户审核常见问题

问题 1：我收到一条消息，指示正在对我的账户进行审核。这表示什么？

我们检测到与从您账户发送的电子邮件相关的问题，我们将给您时间修复此问题。您可以继续按常规方式发送电子邮件，但您还是应该解决导致账户审核的问题。如果您在审核期结束前未更正问题，我们会暂停您继续发送电子邮件的功能。

问题 2：如果要对我的账户进行审核，我一定会收到通知吗？

是。与您的 AWS 账户关联的电子邮件地址将收到一条通知。

问题 3：为什么我没有收到关于我的账户正在接受审核的通知？

当您的账户处于审核状态时，我们会自动向与您的 AWS 账户关联的电子邮件地址发送通知。此电子邮件地址是您在创建 AWS 账户时指定的电子邮件地址。在某些情况下，此电子邮件地址可能不同于您经由 SES 发送电子邮件时使用的地址。

我们建议您定期查看您的声誉指标，以此监控您的发件人信誉。您也可以在 Amazon 中设置自动警报 CloudWatch。当您的声誉指标超过特定阈值时，这些警报可以向您发送通知。您也可以 CloudWatch 将 Amazon 配置为通过其他方式与您联系，例如向您的手机发送短信。

问题 4：我的 SES 账户正在接受审核这一事实会影响我对其他 AWS 服务的使用吗？

在审核您的 SES 账户期间，您仍然可以使用其他 AWS 服务。但是，如果您请求提高对发送出站通信的其他 AWS 服务（例如 Amazon SNS）的服务配额，则在您的 SES 账户被解除审核状态之前，该请求可能会被拒绝。

问题 5：如果我的账户正在接受审核，我该怎么办？

您应该执行以下操作：

- 如果您的情况允许，停止发送邮件，直至修复问题。在账户审核期间，您仍然可以发送电子邮件。但是，如果继续发送邮件而不进行任何更改，可能会无意间让问题变得更糟。
- 查看我们发送给您的电子邮件以了解问题的摘要信息。
- 调查您的发送以确定您的发送具体是哪方面触发了问题。
- 在您做出您认为可以解决问题的更改后，登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供有关您为解决问题所采取的步骤的详细信息，并说明这些步骤将如何防止日后再次出现相同的问题。
- 请务必提供我们明确请求的任何信息。我们需要此信息来评估您的案例。

问题 6：我如何要求重审？

您可以要求我们重审将您的账户置于审核状态的决定。要申请审核，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。

在您的请求中，提供以下信息：

- 有关导致您的账户接受审核的事件的根本原因的信息。
- 您为纠正问题所做的更改的列表。仅包含已实施的步骤，而不包括您计划在未来实施的步骤。
- 有关这些更改将如何防止日后再次出现同一问题的信息。

根据促使我们使您的账户接受审核的事件的性质，我们可能需要其他信息。有关应在请求中包含的信息的列表，请参阅与您遇到的问题关联的常见问题解答主题。

问题 7：如果我的重审要求未被接受，该怎么办？

我们会回复您的请求，并陈述我们不接受此请求的原因。在某些情况下，如果能够证明您已解决问题，并且您的更改可防止日后再次出现同一问题，则可以提交新的请求。

问题 8：您能否帮助我诊断问题？

一般情况下，我们只会为您提供问题的粗略概述 (例如，您具有退回邮件的问题)。您将需要自行调查根本原因。

问题 9：我如何知道对我的账户的审核是否已结束？

声誉指标包括有关您账户的当前状态的信息。有关更多信息，请参阅 使用声誉指标跟踪退回邮件率和投诉率。

问题 10：每次出现问题时，都会对我的账户进行审核吗？

不是的。在某些情况下，我们会暂停账户发送电子邮件的功能，而不是先对账户进行审核。例如：

- 当问题非常严重时。
- 当账户曾因同一问题多次受到审核时。因此，重要的是解决潜在问题，而不是仅处理导致账户受到审核的特定事件。例如，如果某次营销活动导致我们对您的账户进行审核，您不能只停止此次营销活动就算了事。您应该确定该活动的哪些属性存在问题并确保采取措施，以防未来的活动产生相同的问题。

在上述任一情况下，当我们暂停您的账户发送电子邮件的功能时，都会自动向您发送通知。

问题 11：如果我在审核期结束前就解决了问题会怎么样？

登录 AWS Management Console 并前往 Support Center。回复我们代表您开立的问题。在您对问题的回复中，告知我们您已经解决了问题。

问题 12：我能否从我的 AWS 代表或 Premium Support 那里获得帮助？

如果您已经在与 AWS 客户代表合作，当您的账户接受审核时，我们会自动与他或她联系。您的账户代表可能能够提供其他信息，帮助您更好地理解该问题。如果您使用 Premium Support，您还应联系该团队以获取更多帮助。

与暂停发送功能相关的常见问题

问题 1：我收到一条消息，指示已暂停我的账户发送电子邮件的功能。这表示什么？

由于您发送的电子邮件存在严重问题，我们暂停了您的账户发送电子邮件的功能。大多数情况下，我们暂停账户发送功能的原因如下：

- 此前，您的账户受到审核。审核期结束时导致账户审核的问题仍未得到纠正，因此我们暂停了您的账户发送电子邮件的功能。
- 您的账户因为同一问题多次受到审核。
- 您的账户发送的电子邮件违反了 AWS 服务条款。如果违规问题非常严重，我们会暂停账户发送电子邮件的功能，而不是先对账户进行审核。

问题 2：暂停我的账户发送电子邮件的功能时，我一定会收到通知吗？

是。与您的 AWS 账户关联的电子邮件地址将收到一条通知。

问题 3：我的账户发送电子邮件的功能被暂停了。为什么我没有收到通知？

当我们暂停账户的发送电子邮件功能时，将自动将通知发送到与该账户关联的电子邮件地址。

> **Note**
>
> 创建 AWS 账户时，必须提供一个电子邮件地址。您可以随时更改此地址。有关更改与 AWS 账户关联的地址的更多信息，请参阅AWS 账单与成本管理 用户指南中的管理 AWS 账户。

您可以使用 Amazon CloudWatch 创建警报，在退回率和投诉率过高时通知您。创建警报是一种很好的方法，可接收可能会导致暂停您的账户发送电子邮件功能的因素的提前警报。但是，除了邮件退回率和投诉率之外，还有其他的因素会导致我们暂停您发送电子邮件的功能。有关在中创建警报的更多信息 CloudWatch，请参阅使用创建信誉监控警报 CloudWatch。

您还可以使用 Account dashboard（账户控制面板）来确定账户的当前状态。例如，如果当前暂停了您的账户发送电子邮件的功能，账户控制面板的 Account status（账户状态）部分将显示状态 Paused（已暂停）。如果您的账户可以正常发送电子邮件，它将显示状态 Healthy（正常）。

最后，您可以查看 https://phd.aws.amazon.com/处的 AWS Health Dashboard (PHD)，以确定您的账户当前是否已暂停发送电子邮件的功能。在我们暂停账户发送电子邮件的功能时，我们会自动将 SES sending paused (SES 暂停发送功能) 事件发送到 PHD 的 Event log (事件日志) 部分中。SES sending paused (SES 暂停发送功能) 事件的状态始终为 Closed (已关闭)，无论账户发送电子邮件的功能当前是否已暂停。事件日志还包括发送暂停事件发生时我们发送到与您的 AWS 账户关联的电子邮件地址的电子邮件的副本。

您可以使用创建警报 CloudWatch ，在您的 Personal Health Dashboard 上出现新事件时提醒您。有关更多信息，请参阅《AWS Health 用户指南》中的使用 AWS Health CloudWatch 事件监控事件。

问题 4：我的账户发送电子邮件的功能被暂停了。这会影响我使用其他 AWS 服务的能力吗？

当您的账户暂停发送电子邮件的功能时，您仍然可以使用其他 AWS 服务。但是，如果您请求提高对发送出站通信的其他 AWS 服务（例如 Amazon SNS）的服务配额，则在您的账户恢复发送电子邮件功能之前，我们会拒绝请求。

问题 5：如果我的账户发送电子邮件的功能被暂停了，我该怎么办？

您应该执行以下操作：

- 查看我们发送给您的电子邮件以了解问题的摘要信息。
- 调查您的发送以确定您的发送具体是哪方面触发了问题。
- 在您做出您认为可以解决问题的更改后，登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供有关您为解决问题所采取的步骤的详细信息，并说明这些步骤将如何防止日后再次出现相同的问题。
- 请务必提供我们明确请求的任何信息。我们需要此信息来评估您的案例。

问题 6：什么是审核？

您可以要求我们重审我们的审核决定。有关要求重审的更多信息，请参阅以下问题。

问题 7：我如何要求重审？

要申请审核，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。

在您的请求中，提供以下信息：

- 有关导致问题的原因的信息。
- 您为纠正问题所做的更改的列表。仅包含已实施的步骤，而不包括您计划在未来实施的步骤。
- 有关这些更改将如何防止日后再次出现同一问题的信息。

根据促使我们暂停您的账户发送电子邮件功能的事件的性质，我们可能需要其他信息。有关应在请求中包含的信息的列表，请参阅与您遇到的问题关联的常见问题解答主题。

问题 8：如果我的请求未被接受，该怎么办？

我们会回复您的请求，并陈述我们不接受此请求的原因。在某些情况下，如果能够证明您已解决问题，并且您的更改可防止日后再次出现同一问题，则可以提交新的请求。

问题 9：您能否帮助我诊断问题？

一般情况下，我们只会为您提供问题的粗略概述 (例如，您具有退回邮件的问题)。您有责任解决该问题。

问题 10：我如何知道我的账户是否已恢复发送电子邮件的功能？

声誉指标包括有关您账户的当前状态的信息。有关更多信息，请参阅 使用声誉指标跟踪退回邮件率和投诉率。

问题 11：我能否从我的 AWS 代表或 Premium Support 那里获得帮助？

如果您已经在与 AWS 客户代表合作，如果我们暂停了您的账户发送电子邮件的权限，我们将自动与他或她联系。您的账户代表可能能够提供其他信息，帮助您更好地理解该问题。如果您使用 Premium Support，您还应联系该团队以获取更多帮助。

与退回邮件相关的常见问题

问题 1：为什么您关心我的退回邮件？

电子邮件提供商和反垃圾邮件组织等实体通常使用高退回率来检测参与恶意电子邮件发送行为的发件人。高退回率会导致电子邮件发送到垃圾邮件文件夹，而不是收件箱。

问题 2：如果我收到通知，得知我的账户正在接受审核，或者我的账户的邮件退回率导致发送功能暂停，我该怎么办？

确定问题原因，然后纠正问题。在您做出您认为可以解决问题的更改后，登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供有关您为解决问题所采取的步骤的详细信息，并说明这些步骤将如何防止日后再次出现相同的问题。还包括以下信息：

- 您用于跟踪退回邮件的方法
- 在向新收件人发送电子邮件之前，您如何确保新收件人的电子邮件地址是有效的。例如，您遵循了问题 11：我可以采取哪些措施来最大限度地减少退回邮件？中的哪些建议

问题 3：哪些类型的退回邮件将计入我的退回邮件率？

您的退回邮件率仅包含发送到您尚未验证的域的查无此人的邮件。查无此人的邮件是永久性送达失败 (如“地址不存在”)。因为 IP 地址受阻而出现的临时和间歇性故障 (如“邮箱已满”) 或退回邮件不会计入您的退回邮件率。

问题 4：是否会公开可能导致账户审核或发送功能暂停的邮件退回率？

为了获得最佳结果，您应将退回邮件率保持在 2% 以下。退回邮件率高会影响电子邮件的传送。

如果邮件退回率为 5% 或更高，我们将对账户进行审核。如果邮件退回率为 10% 或更高，我们会暂停账户继续发送电子邮件的功能，直到您解决导致高邮件退回率的问题。

问题 5：我的退回邮件率是在哪个时段内计算的？

我们不会基于固定时间段计算退回邮件率，因为不同发件人的发送率不同。相反，我们考量的是典型量，该电子邮件量代表您的典型发送实践。为了对发件量大和发件量小的发件人公平起见，每个用户的典型量都不同并且会随着用户的发送模式的变化而变化。

问题 6：我能否使用 SES 控制台或 GetSendStatistics API 中的信息来计算自己的跳出率？

不能。邮件退回率是使用典型量计算得出的（请参阅问题 5：我的退回邮件率是在哪个时段内计算的？）。根据您的发送率，您的退回邮件率可追溯的时间会比 SES 控制台或 `GetSendStatistics` 可检索的时间更长。此外，在计算您的退回邮件率时，仅会考虑发送到未验证域的电子邮件。不过，如果使用这些方法定期监控邮件退回率，您应该仍会获得有用的指示，可用于捕获问题以免问题达到导致我们对账户进行审核或暂停账户发送电子邮件的功能的程度。

问题 7：我如何才能找出退回邮件的电子邮件地址？

检查 SES 发送给您的退回邮件通知。SES 将通知转发到的电子邮件地址取决于您发送原始邮件的方式，如通过电子邮件接收 Amazon SES 通知中所述。您还可以通过 Amazon Simple Notification Service（Amazon SNS）来设置退回邮件通知，如为 Amazon SES 设置事件通知中所述。请注意，仅从列表中删除退回邮件的地址而不进行任何调查可能解决不了根本问题。有关您可以采取哪些措施来减少退回邮件的信息，请参阅问题 11：我可以采取哪些措施来最大限度地减少退回邮件？。

问题 8：如果我尚未监控我的退回邮件，您能否为我提供已产生退回邮件的地址的列表？

否，我们无法提供已退回地址的完整列表。您有责任监控和处理您的账户的退回邮件。

问题 9：我应该如何处理退回邮件？

您需要从邮件列表中删除退回邮件的地址并立即停止向它们发送邮件。如果您的发件量小，那么通过电子邮件监控退回邮件并从邮件列表中手动删除退回邮件的地址可能就足够了。如果您的发件量大，可能需要为此过程设置自动化，方法是通过以编程方式处理收到退回邮件的邮箱，或通过 Amazon SNS 设置退回邮件通知。有关更多信息，请参阅 为 Amazon SES 设置事件通知。

问题 10：我的电子邮件是否会因为我达到发送配额而退回？

否。退回与发送配额无关。如果您尝试超出您的发送配额，则在您尝试发送电子邮件时，将会收到一条来自 SES API 或 SMTP 接口的错误。

问题 11：我可以采取哪些措施来最大限度地减少退回邮件？

首先，请确保您了解您的退回邮件 (请参阅问题 7：我如何才能找出退回邮件的电子邮件地址？)。然后遵循以下准则：

- 请勿购买、出租或共享电子邮件地址。只向明确要求从您那儿接收电子邮件的收件人发送电子邮件。
- 从列表中删除退回邮件的电子邮件地址。
- 在 Web 表单上，要求用户输入其电子邮件地址两次，并检查以确保这两个地址匹配，然后才能提交表单。
- 使用双向选择性加入来注册新用户。也就是说，当新用户注册时，向他们发送一封确认电子邮件，他们需要点击此电子邮件，然后才能接收任何其他邮件。这将防止用户注册成他人以及意外注册。
- 如果您必须将电子邮件发送到近来无邮件往来的地址 (因此您不确定地址是否仍然有效)，那么请仅发送一小部分邮件。有关更多信息，请参阅我们的博客文章切记不要向旧地址发送邮件，但如果必须这样做，该怎么办？。
- 确保您未组织注册以鼓励用户使用虚构地址。例如，不提供任何附加值或优势，直到收件人验证他们的地址。
- 如果您有“向好友发送电子邮件”功能，请使用 CAPTCHA 或类似机制防止自动使用此功能，并且不要允许用户插入任意内容。
- 如果您使用 SES 来发送系统通知，请确保将通知发送到可收到邮件的真实地址。此外，考虑关闭不需要的通知。
- 如果您正在测试新系统，请确保发送到可收到电子邮件的真实地址，或确保您使用 SES 邮箱模拟器。有关更多信息，请参阅 手动使用邮箱模拟器。

与投诉相关的常见问题

问题 1：什么是投诉？

当收件人报告他们不想接收某封电子邮件时就出现了投诉。他们可能单击了其电子邮件客户端中的“报告垃圾邮件”按钮，向其邮件提供商投诉，直接或通过某种其他方式通知 SES。本主题包含有关投诉的一般信息。如果您的通知包含有关投诉原因的特定信息，另请阅读相关主题：

- 有关 SES 投诉（通过反馈循环接收）的常见问题
- 有关 SES 投诉（直接来自收件人）的常见问题
- 有关 SES 投诉（通过电子邮件提供商）的常见问题

问题 2：为什么您关心我的投诉？

许多实体（如电子邮件提供商和反垃圾邮件组织）通常使用高投诉率作为指标，表示发件人正在将电子邮件发送给未专门注册接收电子邮件的收件人，或者发件人发送的内容与收件人注册请求的内容的类型不同。

问题 3：如果我收到通知，得知我的账户正在接受审核，或者由于与投诉相关的问题导致发送功能暂停，我该怎么办？

检查您的列表获取过程和您的电子邮件内容，尝试了解收件人为什么不喜欢收到您的电子邮件。确定问题原因，然后纠正问题。在您做出您认为可以解决问题的更改后，登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供有关您为解决问题所采取的步骤的详细信息，并说明这些步骤将如何防止日后再次出现相同的问题。

问题 4：我可以采取哪些措施来最大限度地减少投诉？

首先，请务必监控 SES 可能会通知您的投诉，这些投诉是 SES 通过反馈循环接收的（请参阅有关 SES 投诉（通过反馈循环接收）的常见问题）。然后遵循以下准则：

- 请勿购买、出租或共享电子邮件地址。仅使用专门请求您的邮件的地址。
- 使用双向选择性加入来注册新用户。也就是说，当用户注册时，向他们发送一封确认电子邮件，他们需要单击此电子邮件，然后才能接收任何其他邮件。这将防止用户注册成他人以及意外注册。
- 监控您发送的邮件的参与度并停止发送到未打开或点击您的邮件的收件人。
- 当新用户注册时，明确他们将要从您那里收到的电子邮件的类型，并确保您仅发送他们注册请求的邮件类型。例如，如果用户注册接收新闻更新，则请勿向他们发送广告。
- 确保您的邮件格式良好且外观专业。
- 确保您的邮件来源明确，不会与其他内容混淆。
- 为用户明显且简单的方式来取消订阅您的邮件。

有关 SES 投诉（通过反馈循环接收）的常见问题

本主题提供有关 SES 通过反馈循环从电子邮件提供商接收的投诉的信息。有关适用于所有类型的投诉的一般信息，请参阅 与投诉相关的常见问题。

问题 1：这种类型的投诉如何报告？

大多数电子邮件客户端程序会提供一个带有“标记为垃圾邮件”标签的按钮或类似按钮，用于将邮件移至垃圾邮件文件夹并将它转发给电子邮件提供商。此外，大多数电子邮件提供商会维护一个滥用地址（例

如，abuse@example.com），用户可以将不需要的电子邮件转发到此地址并请求电子邮件提供商采取措施阻止它们。如果 SES 与电子邮件提供商一起设置了反馈循环（FBL），电子邮件提供商会将投诉发送回 SES。

> **Note**
>
> 当您发送消息时，SES 会自动设置 Feedback-ID 标头，从而使邮箱提供商能够汇总投诉率、垃圾邮件率等投递统计数据，并将这些数据提供给您。SES 提供的 Feedback-ID 标头值由以下部分组成：
>
> - `FeedBackId:((SESInternalID):(AmazonSES))`，其中：
>   - SESInternalID 是 SES 用于收集投诉信息的标识符。
>   - AmazonSES 是一个将 SES 标识为发送平台的静态标签。
>
> 或者，除了 SES 提供的标准 Feedback-ID 标头值外，您还可以使用`ses:feedback-id-a`和`ses:feedback-id-b`消息标签指定自己的自定义反馈 IDs（最多两个），请参阅。the section called “电子邮件营销活动的细粒度反馈”

问题 2：这些投诉是否包含在 SES 控制台中显示并由 GetSendStatistics API 返回的投诉率统计数据中？

是。但是，请注意，投诉率统计数据不包括来自未向 SES 提供反馈的电子邮件提供商的投诉。来自反馈提供的域的投诉率还可能代表您的其他发送。

问题 3：如何向我通知这些投诉？

可通过电子邮件或 Amazon SNS 通知来通知您。请参阅为 Amazon SES 设置事件通知中的设置说明。

问题 4：如果我通过电子邮件或 Amazon SNS 收到投诉通知，该怎么办？

首先，您需要从邮件列表中删除产生投诉的地址并立即停止向它们发送邮件。请勿发送表示您已收到取消订阅请求的电子邮件。考虑为此过程设置自动化，方式是通过以编程方式处理收到投诉的邮箱，或通过 Amazon SNS 设置投诉通知。有关更多信息，请参阅 为 Amazon SES 设置事件通知。

然后，仔细查看您的发送以确定您的收件人不喜欢您所发送邮件的原因，并解决根本问题。对于每起投诉而言，可能潜在存在无数不喜欢您的邮件但未 (或无法) 投诉的人。如果您只是删除实际投诉的收件人，则没有解决根本问题。

问题 5：是否会公开可能导致账户审核或者可能导致账户发送电子邮件的功能暂停的 SES 投诉率？

为了获得最佳结果，您应将投诉率保持在 0.1% 以下。较高的投诉率会影响电子邮件的传送。

如果投诉率为 0.1% 或更高，我们将对账户进行审核。如果投诉率为 0.5% 或更高，我们会暂停账户继续发送电子邮件的功能，直到您解决导致高投诉率的问题。

问题 6：我的投诉率是在哪个时段内计算的？

我们不会基于固定时间段计算投诉率，因为不同发件人的发送率不同。相反，我们考量的是典型量，该电子邮件量代表您的典型发送实践。为了对发件量大和发件量小的发件人公平起见，每个用户的典型量都不同并且会随着用户的发送模式的变化而变化。此外，投诉率不是基于每封电子邮件计算得出的。相反，它是按照发送到域（向 SES 发送投诉反馈）的邮件的投诉百分比计算的。

问题 7：我能否使用 SES 控制台或 GetSendStatistics API 中的指标来计算自己的投诉率？

不能。有两个主要原因：

- 投诉率是使用典型量计算得出的 (请参阅问题 6：我的投诉率是在哪个时段内计算的？)。根据发送率，投诉率可追溯的时间会比 SES 控制台或 `GetSendStatistics` API 可检索的时间更长。因此，建议定期使用这些方法监控账户的投诉率。以这种方式监控投诉率可以在问题达到可能影响电子邮件送达率的程度之前为您提供识别问题所需的信息。
- 计算投诉率时，不是计算每封电子邮件。投诉率是按照发送到域（向 SES 发送投诉反馈）的邮件的投诉百分比计算的。

问题 8：我如何才能找出产生投诉的电子邮件地址？

检查 SES 通过电子邮件或 Amazon SNS 发送给您的投诉通知（请参阅为 Amazon SES 设置事件通知）。但是，不同的电子邮件提供商会提供不同的信息量，并且一些电子邮件提供商会先编辑收件人的电子邮件地址，然后再将投诉通知传递到 SES。为使您以后能够找到收件人的电子邮件地址，最好是存储标识符与 SES 在接受电子邮件后传递给您的 SES 邮件 ID 之间的映射。请注意，SES 不会保留您添加的任何自定义消息 IDs。

问题 9：如果我尚未监控我的投诉，您能否为我提供已产生投诉的地址的列表？

很遗憾，我们无法为您提供完整的列表。但是，您可以通过电子邮件或 Amazon SNS 监控未来的投诉。

问题 10：我能否获得示例电子邮件？

我们无法应请求向您发送示例电子邮件，但您可在投诉通知中查找此信息。有关更多信息，请参阅 问题 8：我如何才能找出产生投诉的电子邮件地址？。

有关 SES 投诉（直接来自收件人）的常见问题

本主题提供有关 SES 直接从收件人接收的投诉的信息。有关适用于所有类型的投诉的一般信息，请参阅 与投诉相关的常见问题。

问题 1：这种类型的投诉如何报告？

许多收件人会通过电子邮件或一些其他方式就您的邮件直接联系 SES。

问题 2：这些投诉是否包含在 SES 控制台中显示并由 GetSendStatistics API 返回的投诉率统计数据中？

否。您使用 SES 控制台或 `GetSendStatistics` API 检索的投诉率统计数据，仅包含 SES 通过反馈循环收到的投诉。有关这些类型的投诉的更多信息，请参阅有关 SES 投诉（通过反馈循环接收）的常见问题。

问题 3：为什么我尚未通过电子邮件反馈通知或 Amazon SNS 了解到这些投诉？

电子邮件反馈转发和 Amazon SNS 通知仅包含 SES 通过反馈循环收到的投诉。您将不会收到收件人直接向 SES 提出的投诉的通知。

问题 4：我如何才能找出产生投诉的电子邮件地址？

为了保护投诉的接收人的身份，我们无法列出投诉您的电子邮件的电子邮件地址。

我们建议您确定导致投诉的问题，而不是专注于从您的列表中删除单独的收件人。我们建议从审核您的征取客户流程开始，并且从您的列表中删除未明确要求从您那儿接收电子邮件的任何客户。您还应分析您的电子邮件的内容，尽力了解收件人为什么会投诉。

问题 5：我能否获得示例电子邮件？

为了保护投诉的接收人的身份，我们无法提供导致您的收件人投诉的电子邮件副本。

问题 6：如果我收到通知，得知我的账户正在接受审核，或者由于直接投诉问题导致发送功能暂停，我该怎么办？

立即更改您的发送流程，以便只向专门注册来接收您的邮件的收件人发送邮件。此外，确保您要发送的内容类型是您的收件人注册要接收的类型。在您做出您认为可以解决问题的更改后，登录 AWS 控制台

并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供有关您为解决问题所采取的步骤的详细信息，并说明这些步骤将如何防止日后再次出现相同的问题。

如果您未在三周内要求重审，并且我们继续收到收件人直接投诉，我们会暂停账户发送电子邮件的功能。

有关 SES 投诉（通过电子邮件提供商）的常见问题

本主题提供有关 SES 通过电子邮件提供商（也称为邮箱提供商）收到的投诉的信息。有关适用于所有类型的投诉的一般信息，请参阅 与投诉相关的常见问题。

问题 1：这种类型的投诉如何报告？

某电子邮件提供商向 SES 报告，大量客户将您的电子邮件标记为垃圾邮件。该报告是通过有关 SES 投诉（通过反馈循环接收）的常见问题中所述的反馈循环之外的方式提供给 SES 的。

问题 2：这些投诉是否包含在 SES 控制台中显示并由 GetSendStatistics API 返回的投诉率统计数据中？

否。您使用 SES 控制台或 `GetSendStatistics` API 检索的投诉率统计数据，仅包含 SES 通过反馈循环收到的投诉。

问题 3：为什么我尚未通过电子邮件反馈通知或 Amazon SNS 了解到这些投诉？

电子邮件反馈转发和 Amazon SNS 通知仅包含 SES 通过反馈循环收到的投诉。

问题 4：我如何才能找出产生投诉的电子邮件地址？

电子邮件提供商一般不会披露此信息。但是，您需要将重点放在找出并修复根本问题上，而不是放在从您的列表中删除个别收件人上。首先检查您的列表获取过程和您的电子邮件内容以尝试了解您的收件人为什么不喜欢您的电子邮件。

问题 5：我能否获得示例电子邮件？

不能。电子邮件提供商一般不提供示例电子邮件。

问题 6：如果我收到通知，得知我的账户正在接受审核，或者由于电子邮件提供商投诉问题导致发送功能暂停，我该怎么办？

确定问题原因，然后纠正问题。在您做出您认为可以解决问题的更改后，登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供有关您为解决问题所采取的步骤的

详细信息，并说明这些步骤将如何防止日后再次出现相同的问题。如果您三周内未要求重审，并且我们继续收到提供商的投诉，我们会暂停账户继续发送电子邮件的功能。

与垃圾邮件陷阱相关的常见问题

问题 1：什么是垃圾邮件陷阱？

垃圾邮件陷阱是 Internet 服务提供商 (ISP)、电子邮件提供商或反垃圾电子邮件组织维护的一个特殊电子邮件地址。由于这个地址永远不会用来注册接收电子邮件，维护这些垃圾邮件陷阱的组织就会知道，任何向这些地址发送邮件的人可能正在从事有问题的电子邮件活动。

问题 2：如何设置垃圾邮件陷阱？

垃圾邮件陷阱地址有多种设置方式。它们可以从曾经有效、但长期未使用 (和退回) 的地址转换而来。它们也可以是专为垃圾邮件陷阱而设置的地址。它们可以是不常使用的、很难猜的地址，并且有时它们是接近真实地址 (例如，在常见域名中引入拼写错误) 的地址。垃圾邮件陷阱通常通过利用多种方式放置在 Internet 上来“潜入”世界，但并非总是如此。

问题 3：SES 如何知道我是否发送到垃圾邮件陷阱？

某些运行垃圾邮件陷阱的组织将会在 SES 发件人命中其垃圾游戏陷阱时发送 SES 通知。

问题 4：SES 如何使用垃圾邮件陷阱报告？

我们审核报告。如果我们确定您的账户正在向垃圾邮件陷阱发送电子邮件，则对您的账户进行审核，并要求您解决根本问题。如果您在审核期结束前未解决该问题，我们会暂停账户继续发送电子邮件的功能。如果垃圾邮件陷阱问题非常严重，我们会立即暂停账户发送电子邮件的功能，而不是先对账户进行审核。

问题 5：如果我收到通知，得知我的账户正在接受审核，或者由于垃圾邮件陷阱问题导致发送功能暂停，我该怎么办？

首先，您应该解决导致账户审核或暂停发送电子邮件功能的问题。接下来，登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供有关您为解决问题所采取的步骤的详细信息，并说明这些步骤将如何防止日后再次出现相同的问题。如果我们认同您所做的更改能够妥善解决问题，我们将取消账户的审核期或恢复发送电子邮件的功能。

鉴于垃圾邮件陷阱命中的报告方式，我们可能需要三周或更长时间才能确定您所做的更改是否解决了问题。

问题 6：我命中垃圾邮件陷阱几次就会审核我的账户或暂停账户发送电子邮件的功能？

我们不会透露导致对您的账户采取措施的垃圾邮件陷阱的具体次数。但是，请务必注意，即使命中垃圾邮件陷阱的次数很少，也会对您作为发件人的声誉产生非常不利的影响。因此，您应该认真对待垃圾邮件陷阱报告。

问题 7：您是否会披露垃圾邮件陷阱地址？

不会。为了使垃圾邮件陷阱生效，非常重要的是应该对它们保密。垃圾邮件陷阱组织仅披露垃圾游戏陷阱命中的出现，而不会披露实际的垃圾邮件陷阱地址。

问题 8：我可以采取哪些措施来避免发送到垃圾邮件陷阱？

要降低发送到垃圾邮件陷阱的风险，请遵循以下准则：

- 请勿购买、出租或共享电子邮件地址。仅使用专门请求您的邮件的地址。
- 在 Web 表单上，要求用户输入其电子邮件地址两次，并检查以确保这两个地址匹配，然后才能提交表单。
- 使用双向选择性加入来注册新用户。也就是说，当用户注册时，向他们发送一封确认电子邮件，他们需要单击此电子邮件，然后才能接收任何其他邮件。
- 确保从您的列表中删除查无此人的邮件的地址，以便早在它们转化为垃圾邮件陷阱之前删除它们。
- 确保监控收件人的参与情况，并停止发送到最近尚未参与您的电子邮件或网站的收件人。“参与用户”对应的时间范围取决于您的使用案例，但一般来说，如果用户在几个月内没有打开或点击您的电子邮件，您应该考虑删除他们，除非您有证据表明他们确实想要您的邮件。
- 在您有意联系最近未与您交互的人的重新参与活动中请务必谨慎。这些操作往往风险高，并且通常不仅会导致垃圾邮件陷阱发送问题，而且会导致退回邮件和投诉问题。
- 将选择加入邮件发送到您的整个邮件列表并保留仅点击验证链接的收件人。除了从您的列表中删除非活动收件人之外，此过程也有助于删除垃圾邮件陷阱地址。但是，如果认为邮件列表可能包含大量错误地址或者账户已遇到邮件退回问题，建议不要使用此方法，因为它可能会导致账户的邮件退回率进一步增加。

与手动调查相关的常见问题

问题 1：如果我收到通知，得知我的账户正在接受审核，或者由于人工调查问题导致发送功能暂停，我该怎么办？

SES 调查员已确定您存在严重的发送问题。典型问题包括但不限于以下内容：

- 您的发送违反 AWS 可接受的使用策略 (AUP)。
- 您的电子邮件似乎是未经请求的。
- 您的内容与网络钓鱼相关（包括仿真网络钓鱼）。
- 您的内容在其他方面与 Amazon SES 不支持的使用案例关联。

如果我们认为问题可以得到纠正，我们会给您的账户设置一段审核期。当您的账户处于审核状态时，您应该更改电子邮件发送实践以纠正问题。

如果我们认为问题无法纠正，或者问题非常严重，我们会暂停账户发送电子邮件的功能，而不是先审核账户。

问题 2：哪些问题可能导致对我的电子邮件发送进行人工审核？

有几个问题可能会导致我们开始对账户进行人工审核。这些原因包括但不限于以下内容：

- 收件人联系 SES 投诉从您的账户发送的电子邮件。
- 我们检测到您的电子邮件发送模式出现异常变化。
- 我们的垃圾邮件过滤器发现您的电子邮件具有未经请求或低质量电子邮件的特征。

在对账户进行审核或暂停账户发送电子邮件的功能时，我们会向您发送通知。在大多数情况下，此通知包含问题相关信息，并提供有关您可以执行的后续步骤的信息。

问题 3：什么是“未经请求的”电子邮件？

未经请求的电子邮件是指收件人未明确要求接收的电子邮件。这包括收件人注册了某种类型的邮件 (例如，通知) 但收到另一种类型的邮件 (例如，广告) 的情况。

在对账户进行审核或暂停账户发送电子邮件的功能时，我们会向您发送通知。如果您收到通知，说我们正在采取其中一项措施是因为不请自来的电子邮件出现问题，请登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，包含以下信息：

- 您发送的所有邮件是否都是收件人专门请求的内容，它们是否符合 AWS 可接受的使用策略？
- 您是否通过客户明确与您或您的网站交互并从中请求电子邮件之外的任何方式获取电子邮件地址？您应该说明您的邮件列表是如何积累起来的。
- 您的订阅和取消订阅流程如何工作？ 您应包含选择加入和选择退出链接。

问题 4：如果我收到通知，得知我的账户正在接受审核，或者由于人工审核问题导致发送功能暂停，我该怎么办？

确定问题原因，然后纠正问题。在您做出您认为可以解决问题的更改后，登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题。在您的邮件中，提供有关您为解决问题所采取的步骤的详细信息，并说明这些步骤将如何防止日后再次出现相同的问题。如果我们认同您所做的更改能够妥善解决问题，我们将取消账户的审核期。

问题 5：您将哪些类型的问题视为“可纠正的”？

一般而言，我们相信情况是可纠正的，前提是您有良好的发送实践的历史，并且您可以采取措施消除有问题的发送，同时继续大量发送。例如，如果您发送三种不同类型的电子邮件且只有一种类型存在问题，则可以仅停止有问题的发送并继续其余的发送。

问题 6：如果我找不到问题的原因，怎么办？

您可以登录 AWS 控制台并前往 Support Center。回复我们代表您开立的问题，并请求提供导致问题的邮件示例。

DNS 黑洞名单 (DNSBL) FAQs

基于域名系统的黑洞列表 (DNSBLs)（有时也称为实时黑洞列表 (RBLs)、拒绝名单、黑名单或黑名单）旨在向电子邮件提供商通报涉嫌发送有害电子邮件的 IP 地址。

不同 DNSBLs 会对电子邮件送达率产生不同的影响。本主题介绍如何 DNSBLs 影响您使用 Amazon SES 发送的电子邮件的传送，以及我们从中删除 Amazon SES IP 地址的政策 DNSBLs。

> **Note**
>
> 本主题是关于 DNSBLs 电子邮件提供商用来屏蔽传入消息的。对于其电子邮件地址之前已造成退回邮件的收件人，有关 Amazon SES 如何阻止向其发送传出电子邮件的信息，请参阅Amazon SES 全局黑名单。

问题 1：如何 DNSBLs 影响电子邮件的传送？

不同 DNSBLs 会对成功传递消息产生不同的影响。包括Gmail、Hotmail、AOL和Yahoo在内的主要电子邮件提供商似乎认出了极少数备受推崇的电子邮件 DNSBLs，例如Spamhaus提供的那些。根据我们的经验，其他邮件系统 DNSBLs 往往影响较小，尽管有些邮件系统强调某些邮件系统 DNSBLs 而不是其他邮件系统。

最后，许多电子邮件提供商有自己的内部拒绝名单。电子邮件提供商会密切保护这些名单，几乎不会公开分享。如果某个 IP 地址位于这些名单中，就会对您向使用该提供商的收件人发送电子邮件的功能造成巨大影响。

问题 2：IP 地址是如何进入的 DNSBLs？

有多种情况会导致 IP 地址进入 DNSBL。 DNSBLs 当他们向垃圾邮件陷阱发送电子邮件时，可以添加 IP 地址。垃圾邮件陷阱是不属于人类用户的电子邮件地址。垃圾邮件陷阱仅用于收集垃圾邮件和确定垃圾邮件发送者。有些 DNSBLs 还允许个人用户提交 IP 地址。有些 DNSBLs 甚至允许用户提交完整的 IP 地址范围。其他 DNSBLs 则通过电子邮件管理员的贡献进行维护，可能包括管理员认为滥用自己系统的 IP 地址。

问题 3：Amazon SES 是如何防止其 IP 地址出现在上面的 DNSBLs？

我们的系统会寻找滥用迹象。如果检测到可能导致 IP 地址被添加到 DNSBL 的发送模式或其他特征，我们会向发件人发送通知。如果情况严重，或者发送通知后发件人不解决问题，我们将暂停此发件人发送电子邮件的功能，直到他们解决问题。以这种方式执行我们的发送政策有助于减少我们的 IP 地址最终进入的机会 DNSBLs。

问题 4：Amazon SES 能否使其 IP 地址从 DNSBL 中移除？

对于共享的 SES IPs，我们会积极监控 DNSBLs 哪些因素可能会影响整个 Amazon SES 服务的交付，或者可能影响向使用主要电子邮件提供商（例如 Gmail、Yahoo、AOL 和 Hotmail）的收件人发送电子邮件的能力。Spamhaus DNSBLs 提供的产品属于这一类。当我们的某个 IP 地址出现在符合上述任一条件的名单中时，我们会立即采取措施，尽快将该地址从 DNSBL 中移除。

我们不监测 DNSBLs 那些不太可能影响整个 Amazon SES 服务的交付情况，或者对向主要电子邮件提供商的投递没有可衡量影响的情况。SORBS 和 UCEPROTECT DNSBLs 提供的属于这一类。由于运营这些列表的供应商的特定列入和除名做法，我们无法从这些列表中删除我们的 IP 地址。

对于收件人邮件提供商的任何 RBL 中列出的专用 IP 地址（托管或标准），SES 不负责将其除名，您必须直接向 RBL 管理员申请删除。 IPs

问题 5：一位电子邮件提供商拒绝了我的电子邮件，因为发送方 IP 地址列入了 Spamhaus 以外的 DNSBL。我该怎么办？

首先，确认确实由于 DNSBL 而阻止了邮件。如果因发送 IP 地址已进入 DNSBL 而拒绝了您的电子邮件，您会收到提及 DNSBL 提供商名称的退回邮件通知，如下例所示：

```
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
```

如果您收到退回邮件通知，但其中未包含类似于上述示例中所显示消息的信息，电子邮件提供商拒绝您邮件的原因可能与被添加到 DNSBL 无关。

如果您能够确认电子邮件提供商以发送 IP 地址进入 DNSBL 为由阻止了您的电子邮件，您可以执行以下操作：

- 与拒绝您的邮件的域的邮件管理员联系，请求他/她从其垃圾邮件过滤政策中排除您的 IP 地址。某些邮件管理员设有支持流程，并且可能会发布描述此流程的邮件管理页面。如果你要联系的域名没有发布其邮局局长支持政策，你可以通过向 postmast er@ 发送电子邮件来联系邮局局长*example.com*，该域名在*example.com*哪里。RFC 5321 要求域设置邮件管理员邮箱。

  联系邮件管理员时，请提供您收到的退回邮件代码、您尝试发送的电子邮件的标题、DNSBL 对您的电子邮件送达率的影响的衡量标准，以及您认为您的电子邮件被不当阻止的理由。您向邮件管理员提供证明您发送的是合法电子邮件的信息越多，邮件管理员为您的邮件添加例外的可能性就越大。
- 如果电子邮件提供商没有回复或不愿意更改其策略，请考虑使用专用 IP 地址。专用 地址是仅供您使用的 IP 地址。通过实施良好的发送实践，您可以确保高互动率，同时降低邮件退回率、投诉率和垃圾邮件陷阱命中率。良好的发送规范可以帮助确保您的地址不会被发现 DNSBLs。

问题 6：发送到 Gmail、Yahoo、Hotmail 或其他大型提供商的电子邮件被发送到垃圾邮件文件夹中。出现这种情况是否因为我的发送 IP 地址在 DNSBL 中？

可能不是。如果某个 IP 地址被具有重大影响的 DNSBL 列出，例如来 DNSBLs 自 Spamhaus 的电子邮件地址，则主要的电子邮件提供商将完全拒绝来自该IP地址的电子邮件，而不是将其发送到垃圾邮件文件夹。

大型电子邮件提供商在接收电子邮件（而不是拒收电子邮件）时，通常会根据用户互动 来确定是将邮件放入收件箱还是垃圾邮件文件夹。用户互动 是指用户与您之前向其发送的邮件进行交互的方式。

要增加您的邮件进入客户收件箱的机会，您应实施以下所有实践：

- 永远不要 租借或购买电子邮件地址列表。租借或购买列表的行为违反AWS 可接受使用政策（AUP），任何情况下都不允许在 Amazon SES 上发生这类行为。

- 只向明确要求接收您的电子邮件的客户发送电子邮件。在全球众多国家/地区和司法管辖区，向没有明确同意接收您发送的电子邮件的收件人发送电子邮件均是非法的。
- 如果客户没有打开或者点击您在过去 30 到 90 天内发送的邮件中的链接，请停止向这些客户发送电子邮件。此步骤可以帮助您保持较高的参与率，从而提高将来发送的邮件到达收件人收件箱的几率。
- 在发送的每封邮件中使用一致的设计元素和书写风格，以便客户轻松识别来自您的邮件。
- 使用电子邮件身份验证机制，例如 SPF 和 DKIM。
- 当客户使用 Web 表格订阅您的内容时，向他们发送电子邮件来确认对方希望从您这里接收电子邮件。在对方确认希望从您这里接收电子邮件之前，请勿向他们发送任何其他电子邮件。此过程称为确认选择加入 或双向选择加入。
- 让您的客户可以轻松地取消订阅，并且尊重对方的取消订阅请求，立即落实。
- 如果您发送包含链接的电子邮件，请根据 Spamhaus 域阻止列表 (DBL) 检查这些链接。要测试您的链接，请使用 Spamhaus 网站上的域查找工具。

通过实施上述实践，您可以提高发件人声誉，从而增加发送的电子邮件送达收件人收件箱的可能性。此外，实施这些实践还有助于将账户的邮件退回率和投诉率保持在较低水平，并降低向垃圾邮件陷阱发送电子邮件的风险。

Amazon SES 电子邮件发送指标 FAQs

Amazon SES 会收集与您发送的电子邮件有关的几个指标。这些指标可让您分析您的电子邮件程序的有效性并监控重要的统计数据 (如退回邮件率和投诉率)。

本 FAQs 节包含以下与电子邮件发送指标相关的主题：

- 一般性问题
- 打开情况跟踪
- 单击情况跟踪

> **Note**
>
> 事件跟踪取决于收件人的电子邮件服务提供商 (ESP) 以及它们是如何配置超出 Amazon SES 控制的隐私设置。在以下情况下，跟踪事件数可能会偏斜（返回不准确的计数）：
>
> - 电子邮件收件人正在使用保护其隐私的电子邮件服务提供商 (ESP)。
> - 电子邮件收件人明确不授予 ESP 共享其数据的权限。

- 电子邮件收件人的 ESP 缓存图像或链接，SES 只能计算初始打开次数，但无法计算后续打开次数。

一般性问题

问题 1：在传送电子邮件之后，Amazon SES 将在多长的时间内继续收集打开情况和单击情况指标？

在发送每一封电子邮件之后，Amazon SES 会在 60 天内收集打开情况和单击情况指标。

问题 2：如果用户打开了某封电子邮件多次，或者单击了某封电子邮件中的某个链接多次，这些事件是否都会被单独跟踪？

如果收件人打开了某封电子邮件多次，那么 Amazon SES 会将每次打开计为一个唯一的打开事件。同样，如果收件人单击了同一链接多次，那么 Amazon SES 会将每次单击计为一个唯一的单击事件。但是，这些计数可能会因上述备注框中概述的情况而偏斜。

问题 3：是否会聚合打开情况和单击情况指标，或者是否能将它们向下度量到收件人级别？

将在收件人级别跟踪打开情况和点击情况。利用打开情况和单击情况跟踪，您可以确定哪些收件人打开了电子邮件或单击了电子邮件中的链接。

问题 4：我是否能使用 Amazon SES API 来检索打开情况和单击情况指标？

Amazon SES API 不提供用于检索打开情况和单击情况指标的方法。但是，您可以使用 CloudWatch API 检索 Amazon SES 的打开和点击指标。例如，您可以通过发出以下命令来使用 CloudWatch API 检索点击指标： AWS CLI

```
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
```

上面显示的命令将检索 2017 年每天的单击事件的总数。要检索打开情况指标，请将 `metric-name` 参数的值更改为 `Open`。您还可以修改 `start-time` 和 `end-time` 参数以更改分析期，或更改 `period` 参数以实现更精细的分析。

打开情况跟踪

问题 1：打开情况跟踪如何工作？

在通过 Amazon SES 发送的每封电子邮件中插入 1 像素 x 1 像素的透明 GIF 图像，并包含一个对此图像文件的唯一引用；当此图像被下载时，SES 可以准确地分辨出哪封邮件被谁打开过。

默认情况下，此像素将插入电子邮件底部；不过，一些电子邮件提供商的应用程序会在电子邮件超出特定大小时截断电子邮件预览，并且可能会提供用于查看邮件的其余部分的链接。在此场景中，SES 像素跟踪图像不会加载，并且会摆脱您试图跟踪的打开率。要解决此问题，您可以选择将像素置于电子邮件的开头或其他任何位置，方法是将 `{{ses:openTracker}}` 占位符插入电子邮件正文中。在 SES 收到带占位符的邮件后，它将替换为打开跟踪像素图像。

> **Important**
>
> 只需添加一个 `{{ses:openTracker}}` 占位符，因为多个占位符会导致返回 `400 BadRequestException` 错误代码。

添加此跟踪像素不会改变您的电子邮件的外观。

问题 2：打开情况跟踪是否是默认启用的？

默认情况下，打开情况跟踪对所有 Amazon SES 用户可用。要使用打开情况跟踪，您必须执行以下操作：

1. 创建一个配置集。
2. 在该配置集中，创建一个事件目标。
3. 配置该事件目标以将打开事件通知发布到某个目标。
4. 在要跟踪打开情况的每封电子邮件中，指定您在步骤 1 中创建的配置集。

有关如何通过配置集的事件目标启用打开跟踪的详细信息，请参阅the section called “创建事件目标”。您可以像在格式化、原始和模板化电子邮件中一样，在 SMTP 电子邮件中使用像素占位符。

了解有关如何使用事件发布监控电子邮件发送的更多信息。

问题 3：我能否省略某些电子邮件中的打开情况跟踪像素？

有两种方法可以从您的电子邮件中省略打开情况跟踪像素。第一种方法是在不指定配置集的情况下发送电子邮件。或者，您也可以指定未配置为发布有关打开事件的数据的配置集。

问题 4：您是否跟踪纯文本电子邮件的打开情况？

打开情况跟踪仅适用于 HTML 电子邮件。由于打开情况跟踪依赖于包含一个图像，因此无法为使用纯文本 (非 HTML) 电子邮件客户端打开电子邮件的用户收集打开情况指标。

单击情况跟踪

问题 1：单击情况跟踪如何工作？

为了跟踪单击情况，Amazon SES 会修改电子邮件正文中的每个链接。当收件人打开某个链接时，链接请求将被发送到 Amazon SES 服务器，然后立即被转发到目标地址。与打开情况跟踪一样，每个重定向链接都是唯一的。这样使 Amazon SES 能够确定哪些收件人单击了链接，他们何时单击了链接，以及用来访问链接的电子邮件。

> **Important**
>
> 如果您将一封邮件发送给多个收件人，每个收件人都将保存相同的单击情况跟踪链接。要跟踪各个收件人的单击活动，请在每次发送操作中将电子邮件发送给一个收件人。

问题 2：我能否禁用单击情况跟踪？

您可以通过将属性 `ses:no-track` 添加到电子邮件的 HTML 正文中的锚点标签，禁用单独链接的单击跟踪。例如，如果您链接到 AWS 主页，则普通的锚链接如下所示：

```
<a href="https://aws.amazon.com">Amazon Web Services</a>
```

要禁用该链接的单击跟踪，请修改为类似以下内容：

```
<a ses:no-track href="aws.amazon.com">Amazon Web Services</a>
```

因为 `ses:no-track` 不是标准的 HTML 属性，所以 Amazon SES 会从到达您的收件人收件箱的电子邮件版本中自动删除它。

您还可以对使用特定配置集发送的所有邮件禁用单击跟踪。要禁用单击跟踪，请修改配置集事件目标，从而不捕获单击事件。

有关如何通过配置集的事件目标启用和禁用单击跟踪的详细信息，请参阅the section called “创建事件目标”。

了解有关如何使用事件发布监控电子邮件发送的更多信息。

问题 3：在每封电子邮件中可以跟踪多少个链接？

单击跟踪系统最多可以跟踪 250 个链接。

问题 4：是否为纯文本电子邮件中的链接收集单击指标？

只能跟踪 HTML 电子邮件中的单击操作。

问题 5：我能否用唯一标识符来标记链接？

您可以使用 `ses:tags` 属性将任意数量的标签 (键-值对形式) 添加到电子邮件中的链接。当您使用此属性时，请使用将用于传递内联 CSS 属性的相同格式指定键和值：输入键，后跟冒号 (:)，再后跟值。如果您需要传递多个键-值对，请用分号 (;) 分隔每个对。

例如，假设您要将标签 `product:book, genre:fiction, subgenre:scifi, type:newrelease` 添加到某个链接。生成的链接将类似于下面这样：

```
<a ses:tags="product:book;genre:fiction;subgenre:scifi;type:newrelease;"
    href="http://www.amazon.com/…/">New Releases in Science Fiction</a>
```

这些标签将传递到您的事件发布目标，这样您便可以对用户单击过的特定链接执行更多的分析。

> **Note**
>
> 链接标签可以包含数字 0-9、字母 A-Z（大写和小写字母）、连字符 (-) 和下划线 (_)。

问题 6：被跟踪的链接使用 HTTP 协议还是 HTTPS 协议？

跟踪链接与您电子邮件中的原始链接使用相同的协议。

例如，如果您的电子邮件包含一个指向 `https://www.amazon.com` 的链接，则该链接将替换为使用 HTTPS 协议的跟踪链接。如果您的电子邮件包含一个指向 `http://www.example.com` 的链接，

则该链接将替换为使用 HTTP 协议的跟踪链接。如果您的电子邮件包含前面提到的两种链接，那么 HTTPS 链接将替换为使用 HTTPS 协议的跟踪链接，HTTP 链接将替换为使用 HTTP 协议的跟踪链接。

问题 7：不跟踪我的电子邮件中的链接。为什么不跟踪？

Amazon SES 希望您的电子邮件中的链接包含经过正确编码的 URLs链接。具体 URLs 而言，您的链接必须符合 RFC 3986。如果电子邮件中的链接编码不正确，收件人仍会在电子邮件中看到链接，但 Amazon SES 不会跟踪链接的单击事件。

在包含查询字符串的情况下，通常会 URLs 出现与编码不当有关的问题。例如，如果电子邮件中的链接 URL 在查询字符串中包含非编码空格字符 (如以下示例中John" 和 "Doe“在以下示例中：http://www.example。 com/path/to/page？ name=John Doe），亚马逊 SES 不会追踪该链接。但是，如果 URL 改为使用编码的空格字符（例如以下示例中的 “%20”）：http://www.example。 com/path/to/page？ name=john%20DOE），亚马逊 SES 按预期对其进行了跟踪。

快速查找索引

创建以下索引的目的是帮助您快速找到 Amazon SES 中的内容，其中提供了两种搜索方式：操作方法或概念。操作方法描述“如何”做某件事，而概念则解释了更大的图景。

让我们知道您的想法

请使用右上角的 Feedback（反馈）按钮让我们知道...

- 此索引有用吗？
- 您是否希望将任何操作方法或概念添加到此索引中？
- 您认为有什么内容应该以不同方式分类吗？

SES 操作方法和概念链接

How-tos

SES 操作方法链接按字母顺序列出，带您进入相应章节，向您展示“如何”执行您选择的操作。

- 了解如何...
  - 在设置自定义 MAIL FROM 域的过程中添加 SPF 记录
  - 分配 IP 池
  - 阻止电子邮件接收垃圾邮件
  - 配置自定义打开/单击域
  - 配置 SNS 通知
  - 连接到 SMTP 端点
  - 创建配置集
  - 创建域身份
  - 创建电子邮件地址身份
  - 创建事件目标
  - 创建 IP 地址筛选器
  - 创建托管 IP 池以启用专用 IPs（托管）
  - 创建接收规则

- 使用创建信誉警报 CloudWatch
- 使用自定义策略创建发送授权策略
- 使用策略生成器创建发送授权策略
- 为专用 IP 地址（标准）创建标准专用 IP 池
- 删除身份
- 删除个人数据
- 编辑身份
- 启用电子邮件反馈转发
- 导出声誉指标
- 脱离沙盒
- 开始使用 SES
- 开始使用 Virtual Deliverability Manager
- 授予电子邮件接收权限
- 增加吞吐量
- 提升发送配额
- 与现有电子邮件服务器集成
- 记录 API 调用
- 管理配置集
- 管理 Easy DKIM 和 BYODKIM
- 监控发送和声誉指标
- 监控发送统计数据
- 监控使用情况统计数据
- 监控发送配额
- 获取身份的 DKIM 记录
- 获取 SMTP 凭证
- 使用配置集级别抑制来覆盖账户级黑名单
- 覆盖在电子邮件地址身份上继承的 DKIM 签名
- 暂停电子邮件发送
- 发布 MX 记录

- 举报滥用 AWS 资源的情况

- 请求专用 IP 地址
- 请求技术支持
- 使用 Virtual Deliverability Manager Advisor 解决送达和声誉问题
- 从中检索事件数据 CloudWatch
- 从 Kinesis Data Firehose 检索事件数据
- 从 SNS 检索事件数据
- 使用 AWS SDK 发送电子邮件
- 以编程方式发送电子邮件
- 使用 SES API 发送电子邮件
- 使用 SMTP 发送电子邮件
- 使用 CLI 或 SES API 发送带有附件的原始电子邮件
- 使用邮箱模拟器发送测试电子邮件
- 设置 BYODKIM（自带 DKIM）
- 设置 DMARC 策略
- 设置 Easy DKIM
- 设置电子邮件接收
- 设置事件发布
- 设置 MAIL FROM 域
- 设置发送授权（身份所有者任务）
- 设置发送授权（委托发件人任务）
- 在发送电子邮件时指定配置集
- 测试到 SMTP 接口的连接
- 跟踪退信率和投诉率
- 了解继承的 DKIM 签名属性
- 使用声誉指标
- 使用软件包发送电子邮件
- 使用订阅管理
- 使用模板发送电子邮件
- 使用账户级黑名单
- 验证域身份

- 验证电子邮件地址身份
- 查看身份
- 使用 Virtual Deliverability Manager 控制面板大致/详细查看账户送达指标
- 查看专用 SNDS 指标 IPs
- 预热专用 IP 地址

Concepts

SES 概念链接将按字母顺序列出，将带您进入相应的章节来解释所选概念。

- 查找以下相关信息...
  - 滥用 AWS 资源，举报
  - 账户控制面板
  - 账户级黑名单
  - 电子邮件接收的操作选项
  - 添加标头操作
  - 不支持的附件类型
  - 退信响应操作
  - BYODKIM - (自带 DKIM)
  - BYOIP (自带 IP)
  - 代码示例
  - 合规性验证
  - 配置集级别抑制
  - 配置集
  - 内容编码
  - 跨账户通知旧式支持
  - 自定义 MAIL FROM 域
  - 数据保护
  - 专用 IP 地址
  - 专用 IP 地址 (托管式)
  - 专用 IP 地址 (标准)
  - 使用 DKIM 对电子邮件进行身份验证

- DMARC（基于域的消息身份验证、报告和合规性）
- 通过 DKIM 遵守 DMARC
- 通过 SPF 遵守 DMARC
- Easy DKIM
- 电子邮件反馈转发目标
- 电子邮件接收身份验证
- 电子邮件接收概念
- 电子邮件接收控制台演练
- 电子邮件接收恶意软件扫描
- 电子邮件接收权限
- 电子邮件接收使用案例
- 电子邮件接收限制
- 电子邮件发送身份验证方法
- 端点
- 事件通知
- 通过电子邮件发送的事件通知
- 通过 SNS 发送的事件通知
- 事件发布
- FAQs（常见问题）
- 全局黑名单
- 支持的标头字段
- 管理身份
- 身份和访问管理
- 基础设施安全性
- 与 Amazon 集成 acti WorkMail on
- 使用 IP 地址筛选条件进行基于 IP 的控制
- 调用 Lambda 函数操作
- 列表管理
- 列表和订阅
- 日志记录和监控

- 恶意软件检测
- 手动 DKIM 签名
- 使用事件发布监控电子邮件发送
- 监控发件人声誉
- 监控发送活动
- 配额
- 接收规则
- 使用接收规则进行基于收件人的控制
- 区域
- 声誉指标
- 声誉指标消息
- 恢复功能
- 交付到 S3 存储桶操作
- 脱离沙盒
- 安全性
- 支持的安全协议
- 发送授权
- 发送授权策略剖析
- 发送授权策略示例
- 发送授权流程
- 专用 SNDS 指标 IPs
- SNS 通知内容
- SNS 通知示例
- 发布到 SNS 主题操作
- SPF（发件人策略框架）
- 停止规则集操作
- 订阅管理
- 支持，请求技术
- 用于自定义电子邮件验证的模板
- 故障排除

- 已验证的身份
- Virtual Deliverability Manager
- VPC 端点

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。