



デベロッパーガイド

# Amazon Simple Storage Service



API バージョン 2006-03-01

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

# Table of Contents

|                                                      |      |
|------------------------------------------------------|------|
| Amazon S3 とは .....                                   | 1    |
| API コールのレコメンデーション .....                              | 2    |
| 直接 REST API コールを行う .....                             | 2    |
| アクセス許可 .....                                         | 3    |
| Amazon S3 を使用した開発 .....                              | 4    |
| リクエストの実行 .....                                       | 4    |
| アクセスキーについて .....                                     | 5    |
| リクエストのエンドポイント .....                                  | 7    |
| IPv6 を使用したリクエストの実行 .....                             | 7    |
| AWS SDK を使用したリクエストの実行 .....                          | 18   |
| REST API を使用したリクエストの実行 .....                         | 57   |
| AWS CLI の使用 .....                                    | 70   |
| AWS CLI の詳細 .....                                    | 71   |
| AWS SDK を使用した開発 .....                                | 71   |
| SDK プログラミングインターフェイス .....                            | 72   |
| リクエスト認証での署名バージョンの指定 .....                            | 72   |
| サポート の Amazon S3 リクエスト ID を取得する .....                | 82   |
| AWS CLI を使用したリクエスト ID の取得 .....                      | 83   |
| Windows PowerShell を使用してリクエスト ID を取得する .....         | 83   |
| AWS CloudTrail データイベントを使用してリクエスト ID を取得する .....      | 83   |
| S3 サーバーアクセスログ記録を使用してリクエスト ID を取得する .....             | 83   |
| HTTP を使用したリクエスト ID の取得 .....                         | 84   |
| ウェブブラウザを使用したオブジェクトリクエスト ID の取得 .....                 | 84   |
| AWS SDK を使用したリクエスト ID の取得 .....                      | 84   |
| S3 Tables でサポートされている S3 オブジェクトレベルの API オペレーション ..... | 89   |
| コードの例 .....                                          | 95   |
| Amazon S3 .....                                      | 101  |
| 基本 .....                                             | 106  |
| シナリオ .....                                           | 808  |
| サーバーレスサンプル .....                                     | 1264 |
| Amazon S3 Control .....                              | 1276 |
| 基本 .....                                             | 1277 |
| S3 ディレクトリバケット .....                                  | 1358 |
| 基本 .....                                             | 1359 |

|                                          |      |
|------------------------------------------|------|
| シナリオ .....                               | 1485 |
| リクエストの認証 (AWS 署名バージョン 4) .....           | 1489 |
| 認証方法 .....                               | 1490 |
| リクエストへの署名の概要 .....                       | 1491 |
| Authorization ヘッダーの使用 .....              | 1492 |
| 概要 .....                                 | 1493 |
| 署名計算: 単一のチャンクでペイロードを転送する .....           | 1497 |
| 署名計算: ペイロードを複数のチャンクで転送する .....           | 1516 |
| 署名の計算: 末尾ヘッダーを含める .....                  | 1529 |
| クエリパラメータの使用 .....                        | 1535 |
| 署名の計算 .....                              | 1539 |
| 例 .....                                  | 1542 |
| 例 2 .....                                | 1544 |
| 例: 署名計算 .....                            | 1544 |
| Java を使用した署名計算の例 .....                   | 1545 |
| C# を使用した署名計算の例 .....                     | 1547 |
| HTTP POST リクエストの認証 .....                 | 1548 |
| 署名の計算 .....                              | 1550 |
| Amazon S3 署名バージョン 4 の認証固有のポリシーキー .....   | 1551 |
| 署名バージョン 4 に関連する条件キーを使用するバケットポリシーの例 ..... | 1554 |
| POST を使用したブラウザベースのアップロード .....           | 1558 |
| POST Object .....                        | 1559 |
| 説明 .....                                 | 1559 |
| バージョニング .....                            | 1560 |
| リクエスト .....                              | 1560 |
| 例 .....                                  | 1580 |
| 関連リソース .....                             | 1581 |
| POST Object restore .....                | 1582 |
| 説明 .....                                 | 1582 |
| Select リクエストを使用したアーカイブのクエリ .....         | 1582 |
| アーカイブの復元 .....                           | 1584 |
| リクエスト .....                              | 1585 |
| レスポンス .....                              | 1600 |
| 例 .....                                  | 1602 |
| 詳細 .....                                 | 1604 |
| HTTP POST を使用したブラウザベースのアップロード .....      | 1604 |



|                                                                 |      |
|-----------------------------------------------------------------|------|
| 署名の計算 .....                                                     | 1606 |
| HTML フォームの作成 .....                                              | 1607 |
| HTML フォーム宣言 .....                                               | 1608 |
| HTML フォームフィールド .....                                            | 1609 |
| POST ポリシー .....                                                 | 1615 |
| 有効期限 .....                                                      | 1616 |
| 条件一致 .....                                                      | 1617 |
| 条件 .....                                                        | 1618 |
| 文字のエスケープ .....                                                  | 1623 |
| POST アップロードの例 .....                                             | 1624 |
| HTTP POST を使用した Amazon S3 へのファイルのアップロード .....                   | 1624 |
| AWS Amplify を使用したブラウザベースのアップロード .....                           | 1627 |
| AWS Amplify JavaScript ライブラリを使用した Amazon S3 へのファイルのアップロード ..... | 1627 |
| 詳細 .....                                                        | 1628 |
| 一般的なリクエストヘッダー .....                                             | 1629 |
| 共通のレスポンスヘッダー .....                                              | 1633 |
| エラーレスポンス .....                                                  | 1638 |
| REST エラーレスポンス .....                                             | 1639 |
| エラーコードのリスト .....                                                | 1640 |
| SELECT Object Content エラーコードのリスト .....                          | 1672 |
| レプリケーション関連のエラーコードのリスト .....                                     | 1686 |
| タグ付け関連のエラーコードのリスト .....                                         | 1689 |
| Amazon S3 注釈関連のエラーコードのリスト .....                                 | 1690 |
| Amazon S3 on Outposts エラーコードのリスト .....                          | 1694 |
| Amazon S3 ストレージレンズエラーコードのリスト .....                              | 1696 |
| Amazon S3 Object Lambda エラーコードのリスト .....                        | 1706 |
| Amazon S3 の非同期エラーコードのリスト .....                                  | 1710 |
| Amazon S3 Access Grants エラーコードのリスト .....                        | 1712 |
| Amazon FSx 関連のエラーコードのリスト .....                                  | 1715 |
| Amazon S3 Tables のエラーコードのリスト .....                              | 1716 |
| Amazon S3 のエラーに関するベストプラクティス .....                               | 1717 |
| InternalError の場合は再試行する .....                                   | 1717 |
| SlowDown エラーを繰り返すアプリケーションを調整する .....                            | 1717 |
| エラーを分離する .....                                                  | 1718 |
| AWS 用語集 .....                                                   | 1719 |
| リソース .....                                                      | 1720 |

|                                                |      |
|------------------------------------------------|------|
| 付録 .....                                       | 1722 |
| 付録: SelectObjectContent レスポンス .....            | 1723 |
| 説明 .....                                       | 1723 |
| レスポンス .....                                    | 1723 |
| 関連リソース .....                                   | 1733 |
| 付録: OPTIONS オブジェクト .....                       | 1735 |
| 説明 .....                                       | 1735 |
| リクエスト .....                                    | 1735 |
| レスポンス .....                                    | 1736 |
| 例 .....                                        | 1738 |
| 関連リソース .....                                   | 1738 |
| 付録: SOAP API .....                             | 1739 |
| サービスに対するオペレーション (SOAP API) .....               | 1739 |
| バケットに対するオペレーション (SOAP API) .....               | 1741 |
| オブジェクトに対するオペレーション (SOAP API) .....             | 1754 |
| SOAP リクエストの認証方法 .....                          | 1779 |
| SOAP のアクセスポリシーの設定 .....                        | 1781 |
| 一般的な要素 .....                                   | 1783 |
| SOAP エラーレスポンス .....                            | 1783 |
| 付録: リクエストの認証 (AWS 署名バージョン 2) .....             | 1785 |
| REST API を使用したリクエストの認証(AWS 署名バージョン 2) .....    | 1786 |
| REST リクエストの署名と認証(AWS 署名バージョン 2) .....          | 1789 |
| POST (AWS 署名バージョン 2) を使用したブラウザベースのアップロード ..... | 1804 |
| 付録: ライフサイクル設定 API (廃止) .....                   | 1828 |
| PUT バケットライフサイクル (廃止) .....                     | 1830 |
| GET バケットライフサイクル (廃止) .....                     | 1847 |
| ドキュメント履歴 .....                                 | 1859 |

# Amazon S3 とは

Amazon Simple Storage Service (Amazon S3) は、業界をリードするスケーラビリティ、データ可用性、セキュリティ、およびパフォーマンスを提供するオブジェクトストレージサービスです。あらゆる規模や業界のお客様が、Amazon S3 を使用して、データレイク、ウェブサイト、モバイルアプリケーション、バックアップおよび復元、アーカイブ、エンタープライズアプリケーション、IoT デバイス、ビッグデータ分析など、広範なユースケースのデータを容量にかかわらず、保存して保護することができます。Amazon S3 には、特定のビジネス、組織、コンプライアンスの要件を満たすために、データへのアクセスを最適化、整理、設定できる管理機能があります。

このガイドでは、Amazon S3 REST API を使用してクラウドにデータを保存および取得するアプリケーションを構築する方法について、開発者向けの情報を提供します。

REST API を使用する場合、HTTP をサポートする任意のツールキットを使用できます。匿名で読み取り可能なオブジェクトであれば、ブラウザを使用して取得することもできます。

REST API は標準の HTTP ヘッダーとステータスコードを使用するため、標準のブラウザとツールキットが予期したとおりに機能します。一部のエリアでは、HTTP に機能が追加されています (たとえば、アクセスコントロールをサポートするヘッダーを追加しました)。このように新機能を追加する場合、できるだけ標準 HTTP 書式の使用法に合致するように最善を尽くしました。

## Note

HTTP 経由の SOAP のサポートは廃止されましたが、HTTPS 経由では引き続き利用可能です。ただし、SOAP では、Amazon S3 の新機能はサポートされなくなります。以下のリンクにある REST API または AWS SDK を使用することをお勧めします。

<https://aws.amazon.com/developer/tools/>

このデベロッパーガイドには以下が含まれます。

- [一般的なリクエストヘッダー](#) および [共通のレスポンスヘッダー](#)
- [エラーレスポンス](#)
- [POST \(AWS 署名バージョン 4\) を使用したブラウザベースのアップロード](#)

 Important

特定の API トピックに進む前に、認証とアクセスコントロールについて以下をお読みください。

Amazon S3 へのリクエストは、認証または匿名で行うことができます。認証を使用してのアクセスには、AWS がリクエストを認証するために使用できる認証情報が必要です。

## API コールのレコメンデーション

コードから直接 REST API コールを行うのは手間がかかる場合があります。リクエストを認証するための有効な署名を計算するために必要なコードを記述する必要があります。代わりに、次の代替方法をお勧めします。

- [AWS SDK](#) を使用してリクエストを送信します。

また、[サンプルコードとライブラリ](#)も参照してください。

SDK を使用する場合、SDK クライアントは指定したアクセスキーを使用してリクエストを認証するため、リクエスト認証の署名を計算するコードを記述する必要はありません。正当な理由がない限り、常に AWS SDK を使用してください。

- AWS CLI を使用して Amazon S3 API コールを行います。AWS CLI のセットアップと Amazon S3 コマンドの例については、以下のトピックを参照してください。

「Amazon Simple Storage Service ユーザーガイド」の「[AWS CLI のセットアップ](#)」。

「AWS Command Line Interface ユーザーガイド」の「[AWS コマンドラインインターフェイスでの Amazon S3 の使用](#)」。

## 直接 REST API コールを行う

 Note

PUT リクエストヘッダーのサイズは 8 KB に制限されています。PUT リクエストヘッダー内のシステム定義メタデータのサイズは 2 KB に制限されています。システム定義メタデータ

のサイズは、キーと値の各ペアを US-ASCII にエンコードしたバイト数の合計に基づいて測定されます。

上記の代替方法のいずれかを使用する代わりに独自の REST API コールを行う場合は、いくつかの点に留意する必要があります。

- コードから直接 REST API コールを行うには、有効な認証情報を使用して署名を作成し、リクエストにその署名を含めます。さまざまな認証方法と署名の計算については、「[リクエストの認証 \(AWS 署名バージョン 4\)](#)」を参照してください。
- REST API は標準の HTTP ヘッダーとステータスコードを使用するため、標準のブラウザとツールキットが予期したとおりに機能します。一部のエリアでは、HTTP に機能が追加されています (たとえば、アクセスコントロールをサポートするヘッダーを追加しました)。このように新機能を追加する場合、できるだけ標準 HTTP 書式の用法に合致するように最善を尽くしました。リクエストの実行の詳細については、「[リクエストの実行](#)」を参照してください。

## アクセス許可

リクエストを認証するために有効な認証情報を持つことができますが、アカウント所有者またはバケット所有者からの S3 のアクセス許可を持っていない場合は Amazon S3 リソースを作成またはアクセスすることはできません。これらのアクセス許可は通常、バケットポリシーなどの AWS Identity and Access Management (IAM) [ポリシー](#)を通じて付与されます。例えば、S3 バケットを作成したり、バケット内のオブジェクトを取得したりするには、アクセス許可が必要です。S3 のアクセス許可の一覧については、「[Amazon S3 のアクション、リソース、および条件キー](#)」を参照してください。

S3 リソースタイプ別の S3 API オペレーションへのアクセス許可の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[Amazon S3 API オペレーションに必要なアクセス許可](#)」を参照してください。

AWS アカウントのルートユーザー認証情報を使用する場合は、すべてのアクセス許可があります。ただし、ルートユーザーの認証情報は使用しないことをお勧めします。代わりに、アカウントに AWS Identity and Access Management (IAM) ロールを作成し、ユーザーのアクセス許可を管理することをお勧めします。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[アクセス管理](#)」を参照してください。

# Amazon S3 を使用した開発

このセクションでは、Amazon S3 の使用についてデベロッパーに関連するトピックを説明します。詳細については、以下のトピックを参照してください。

## トピック

- [リクエストの実行](#)
- [AWS CLI を使用した Amazon S3 での開発](#)
- [AWS SDK を使用した Amazon S3 での開発](#)
- [AWS サポート の Amazon S3 リクエスト ID の取得](#)
- [S3 Tables でサポートされている Amazon S3 オブジェクトレベルの API オペレーション](#)

## リクエストの実行

Amazon S3 は REST サービスです。REST API または基礎となる Amazon S3 REST API をラップする AWS SDK ([サンプルコードとライブラリ](#)を参照) ラッパーライブラリを使用して、Amazon S3 にリクエストを送信できます。これにより、プログラミング作業が簡易になります。

Amazon S3 とのすべてのやり取りは認証されるか匿名で行われます。認証は、アマゾン ウェブ サービス (AWS) 製品にアクセスを試みるリクエストのアイデンティティを検証するプロセスです。認証リクエストには、リクエストの送信元を認証する署名値を含める必要があります。署名値の一部は、リクエストの AWS アクセスキー (アクセスキー ID とシークレットアクセスキー) から生成されます。アクセスキーの取得の詳細については、AWS 全般のリファレンスの「[セキュリティ認証情報の取得方法](#)」を参照してください。

AWS SDK を使用している場合、指定したキーから、ライブラリによって署名が計算されます。ただし、アプリケーションで直接 REST API を呼び出す場合、署名を計算するコードを作成し、それをリクエストに追加する必要があります。

## トピック

- [アクセスキーについて](#)
- [リクエストのエンドポイント](#)
- [IPv6 経由で Amazon S3 へのリクエストを行う](#)
- [AWS SDK を使用したリクエストの実行](#)
- [REST API を使用したリクエストの実行](#)

## アクセスキーについて

以下のセクションでは、認証リクエストの実行に使用できるアクセスキーの種類について説明します。

### AWS アカウント アクセスキー

アカウントアクセスキーを使用することで、アカウントが所有する AWS リソースにフルアクセスすることができます。アクセスキーの例を次に示します。

- アクセスキー ID (半角英数字で構成された 20 文字)。例: AKIAIOSFODNN7EXAMPLE
- シークレットアクセスキー (40 文字の文字列)。例: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

アクセスキー ID によって AWS アカウント が一意に識別されます。このアクセスキーを使用して認証リクエストを Amazon S3 に送信できます。

### IAM ユーザーアクセスキー

お客様の会社用に 1 つの AWS アカウント を作成することができます。ただし、組織の AWS リソースへのアクセス権を必要とする社員が組織内に複数いる場合があります。AWS アカウント アクセスキーを共有するとセキュリティは低くなり、また、各社員に個別の AWS アカウント を作成することは実用的ではない場合があります。さらに、バケットやオブジェクトなどのリソースは別のアカウントが所有しているので、簡単に共有できません。リソースを共有するには、追加の作業でアクセス許可を付与する必要があります。

この場合、AWS Identity and Access Management (IAM) を使用して、AWS アカウントでユーザーと各自のアクセスキーを作成し、これらのユーザーに該当するリソースへのアクセス許可を付与する IAM ユーザーポリシーをアタッチします。このようなユーザーをより効率的に管理するために、IAM では、ユーザーのグループを作成し、グループ内のユーザー全員に適用されるグループレベルのアクセス許可を付与できます。

で作成および管理するユーザーは、IAM ユーザーと呼ばれますAWS 親アカウントは、ユーザーが にアクセスできるかどうかを制御しますAWS IAM ユーザーが作成するリソースは、親 AWS アカウント が管理し、支払いを行います。これらの IAM ユーザーは、各自のセキュリティ証明書を使用して Amazon S3 に認証リクエストを送信できます。お客様の AWS アカウント の下でユーザーを作成、管理する方法の詳細については、「[AWS Identity and Access Management 製品詳細ページ](#)」を参照してください。



## 一時的な認証情報

IAM では、IAM ユーザーおよび独自のアクセスキーを作成できるだけでなく、任意の IAM ユーザーに一時的なセキュリティ認証情報 (一時的なアクセスキーとセキュリティトークン) を付与して、AWS のサービスおよびリソースにアクセスできるようにすることもできます。また、ではなく、使用しているシステムでユーザーを管理することもできます。AWS のようなユーザーはフェデレーテッドユーザーと呼ばれます。また、このユーザーには、AWS リソースにアクセスするために作成したアプリケーションも含まれます。

IAM には、一時的なセキュリティ認証情報をリクエストするための AWS Security Token Service API が用意されています。一時的なセキュリティ認証情報をリクエストするには、AWS STS API または AWS SDK を使用できます。この API は一時的なセキュリティ認証情報 (アクセスキー ID およびシークレットアクセスキー) とセキュリティトークンを返します。これらの認証情報は、リクエスト時に指定した有効期間の間のみ有効です。アクセスキー ID およびシークレットキーの使用法は、AWS アカウント または IAM ユーザーのアクセスキーを使用してリクエストを送信するときの使用法と同様です。また、Amazon S3 に送信する各リクエストにトークンを含める必要があります。

IAM ユーザーは、これらの一時的なセキュリティ認証情報をリクエストした後、自分で使用することも、フェデレーテッドユーザーまたはアプリケーションに渡すこともできます。フェデレーションユーザー用に一時セキュリティ証明書を送信する場合、ユーザー名と、その一時セキュリティ証明書に関連付ける許可を定義した IAM ポリシーを指定する必要があります。フェデレーションユーザーに、一時証明書をリクエストした親 IAM ユーザーよりも多くの権限が付与されることはありません。

Amazon S3 にリクエストするときに、これらの一時的なセキュリティ認証情報を使用できます。API ライブラリによって、これらの認証情報を使用して必要な署名値が計算されて、リクエストが認証されます。失効した証明書を使用してリクエストを送信した場合、Amazon S3 はリクエストを拒否します。

REST API リクエストで一時セキュリティ証明書を使用してリクエストに署名する方法については、[REST リクエストの署名と認証\(AWS 署名バージョン 2\)](#) を参照してください。AWS SDK を使用してリクエストを送信する方法については、[AWS SDK を使用したリクエストの実行](#) を参照してください。

一時的なセキュリティ認証情報の IAM によるサポートの詳細については、IAM ユーザーガイドの「[IAM の一時的なセキュリティ認証情報](#)」を参照してください。

Amazon S3 リソースにアクセスする際のセキュリティを高めるには、バケットポリシーを設定して多要素認証 (MFA) を要求することができます。詳細については、「[バケットポリシーの例: MFA が](#)



[必要](#)」を参照してください。Amazon S3 リソースにアクセスするために MFA を要求した後に、これらのリソースにアクセスできる唯一の方法は、MFA キーで作成された一時的な認証情報を提供することです。詳細については、[AWS 多要素認証](#)の詳細ページと IAM ユーザーガイドの [MFA 保護 API アクセスの設定](#)を参照してください。

## リクエストのエンドポイント

サービスの定義済みエンドポイントに対して REST リクエストを送信します。すべての AWS のサービスとそれに対応するエンドポイントのリストについては、AWS 全般のリファレンスの「[リジョンとエンドポイント](#)」を参照してください。

## IPv6 経由で Amazon S3 へのリクエストを行う

Amazon Simple Storage Service (Amazon S3) では、IPv4 プロトコルに加えて、Internet Protocol version 6 (IPv6) を使用して S3 バケットにアクセスする機能をサポートしています。Amazon S3 デュアルスタックエンドポイントは、IPv6 および IPv4 を使用した S3 バケットへのリクエストをサポートしています。IPv6 を使用して Amazon S3 にアクセスする場合、追加料金はかかりません。料金に関する詳細については、「[Amazon S3 の料金](#)」を参照してください。

### Note

以下の情報は、パブリック Amazon S3 IPv6 エンドポイントに関するものです。AWS 仮想プライベートクラウド (VPC) の詳細については、「[AWS PrivateLink for Amazon S3](#)」を参照してください。

### トピック

- [IPv6 を使用したリクエストの実行の開始方法](#)
- [IAM ポリシーでの IPv6 アドレスの使用](#)
- [IP アドレス互換性のテスト](#)
- [Amazon S3 デュアルスタックのエンドポイントの使用](#)

## IPv6 を使用したリクエストの実行の開始方法

IPv6 を使用して S3 バケットにリクエストを実行するには、デュアルスタックのエンドポイントを使用する必要があります。次のセクションでは、デュアルスタックのエンドポイントを使用した IPv6 でのリクエストの実行方法について説明します。

以下は、IPv6 でバケットにアクセスを試行する前に知っておく必要がある点です。

- バケットにアクセスするクライアントやネットワークは、IPv6 の使用を有効にする必要があります。
- 仮想ホスト形式およびパス形式のリクエストは、IPv6 アクセスをサポートしています。詳細については、「[Amazon S3 デュアルスタックのエンドポイントを使用する](#)」を参照してください。
- AWS Identity and Access Management (IAM) ユーザーまたはバケットポリシーで、ソース IP アドレスによるフィルタリングを使用する場合、IPv6 アドレス範囲を含めるようポリシーを更新する必要があります。詳細については、「[IAM ポリシーでの IPv6 アドレスの使用](#)」を参照してください。
- IPv6 を使用する場合、サーバーのアクセスログファイルは IPv6 形式の IP アドレスを出力します。IPv6 形式の Remote IP アドレスを解析できるように、Amazon S3 ログファイルの解析に使用する既存のツール、スクリプト、ソフトウェアを更新する必要があります。詳細については、「[サーバーアクセスログによるリクエストのログ記録](#)」を参照してください。

 Note

ログファイルで IPv6 アドレスに関連する問題を見つけた場合、[AWS サポート](#) にお問い合わせください。

デュアルスタックのエンドポイントを使用して IPv6 経由でリクエストを実行する

IPv6 での Amazon S3 API コールを使用したリクエストは、デュアルスタックのエンドポイントを使用して行います。Amazon S3 API オペレーションは、IPv6 経由でも IPv4 経由でも、Amazon S3 へのアクセスは同じように機能します。パフォーマンスも同じです。

REST API を使用する場合、デュアルスタックのエンドポイントに直接アクセスします。詳細については、「[デュアルスタックのエンドポイント](#)」を参照してください。

AWS Command Line Interface (AWS CLI) や AWS SDK を使用する場合、パラメータまたはフラグを使ってデュアルスタックのエンドポイントに変更できます。設定ファイルの Amazon S3 エンドポイントに上書きしてデュアルスタックのエンドポイントを直接指定することもできます。

デュアルスタックのエンドポイントを使用して、次のいずれかから IPv6 でバケットにアクセスできます。

- AWS CLI については、「[AWS CLI からのデュアルスタックのエンドポイントの使用](#)」を参照してください。
- AWS SDK については、[AWS SDK からデュアルスタックのエンドポイントを使用する](#) を参照してください。
- REST API については、[REST API を使用したデュアルスタックのエンドポイントへのリクエストの実行](#)を参照してください。

## IPv6 で使用できない機能

IPv6 を介して S3 バケットにアクセスする場合、S3 バケットからの静的ウェブサイトホスティングの機能は現在サポートされていません。

## IAM ポリシーでの IPv6 アドレスの使用

IPv6 を使用してバケットにアクセスを試行する前に、IP アドレスによるフィルタリングに使用される IAM ユーザーまたは S3 バケットポリシーが IPv6 アドレス範囲を含むように更新されているか確認する必要があります。IPv6 アドレスを処理するために更新されていない IP アドレスのフィルタリングポリシーでは、IPv6 の使用を開始する際、間違ってクライアントがバケットへのアクセスを取得または失う場合があります。IAM でのアクセス許可管理の詳細については、「[Amazon S3 用 Identity and Access Management](#)」を参照してください。

IP アドレスをフィルタリングする IAM ポリシーは、[IP アドレス条件演算子](#)を使用します。次のバケットポリシーは、IP アドレス条件演算子を使用して 54.240.143.\* の範囲の許可された IPv4 アドレスを識別します。この範囲外のすべての IP アドレスはバケットへのアクセスを拒否されます (examplebucket)。すべての IPv6 アドレスは許可範囲外であるため、このポリシーは IPv6 アドレスの examplebucket へのアクセスをブロックします。

## JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
```

```
    "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
  }
}
]
```

次の例のように、バケットポリシーの Condition エlement を変更して、IPv4 (54.240.143.0/24) および IPv6 (2001:DB8:1234:5678::/64) アドレス範囲の両方を許可できます。例に示すように、IAM ユーザーとバケットポリシーの両方を更新するために同じタイプの Condition ブロックを使用できます。

```
"Condition": {
  "IpAddress": {
    "aws:SourceIp": [
      "54.240.143.0/24",
      "2001:DB8:1234:5678::/64"
    ]
  }
}
```

IPv6 を使用する前に、IP アドレスのフィルタリングを使用するすべての関連 IAM ユーザーとバケットポリシーを更新する必要があります。バケットポリシーで IP アドレスのフィルタリングを使用することはお勧めしません。

IAM ユーザーポリシーは、<https://console.aws.amazon.com/iam/> の IAM コンソールを使用して確認できます。IAM の詳細については、[IAM ユーザーガイド](#)を参照してください。S3 バケットポリシーの編集については、「[バケットポリシーの追加](#)」を参照してください。

## IP アドレス互換性のテスト

Linux/Unix または Mac OS X を使用している場合、次の例に示すように curl コマンドを使用して IPv6 でデュアルスタックのエンドポイントにアクセスできるかどうかテストできます。

### Example

```
curl -v http://s3.dualstack.us-west-2.amazonaws.com/
```

次の例のような情報を取得できます。IPv6 で接続している場合、接続されている IP アドレスは IPv6 アドレスです。

```
* About to connect() to s3-us-west-2.amazonaws.com port 80 (#0)
* Trying IPv6 address... connected
* Connected to s3.dualstack.us-west-2.amazonaws.com (IPv6 address) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1 OpenSSL/1.0.1t
  zlib/1.2.3
> Host: s3.dualstack.us-west-2.amazonaws.com
```

Windows 7 または 10 を使用している場合、次の例に示すように ping コマンドを使用して IPv6 または IPv4 でデュアルスタックのエンドポイントにアクセスできるかどうかテストできます。

```
ping ipv6.s3.dualstack.us-west-2.amazonaws.com
```

## Amazon S3 デュアルスタックのエンドポイントの使用

Amazon S3 デュアルスタックエンドポイントは、IPv6 および IPv4 を使用した S3 バケットへのリクエストをサポートしています。このセクションは、スタックのエンドポイントを使用する方法を説明します。

### トピック

- [Amazon S3 デュアルスタックのエンドポイントを使用する](#)
- [AWS CLI からのデュアルスタックのエンドポイントの使用](#)
- [AWS SDK からデュアルスタックのエンドポイントを使用する](#)
- [REST API からのデュアルスタックのエンドポイントの使用](#)

### Amazon S3 デュアルスタックのエンドポイントを使用する

デュアルスタックのエンドポイントにリクエストを行うと、バケット URL は IPv6 または IPv4 アドレスに解決されます。IPv6 でのバケットへのアクセスに関する詳細は、「[IPv6 経由で Amazon S3 へのリクエストを行う](#)」を参照してください。

REST API を使用する場合、エンドポイント名 (URI) を使用して直接 Amazon S3 エンドポイントにアクセスします。仮想ホスト形式またはパス形式のエンドポイント名を使用することで、デュアルスタックのエンドポイント経由で S3 バケットにアクセスできます。Amazon S3 はリージョンのデュアルスタックエンドポイント名のみをサポートしており、名前の一部としてリージョンを指定する必要があります。

デュアルスタックの仮想ホスト形式とパス形式のエンドポイント名には、次の命名規則を使用します。

- 仮想ホスティング形式のデュアルスタックのエンドポイント:

`bucketname.s3.dualstack.aws-region.amazonaws.com`

- パス形式のデュアルスタックのエンドポイント:

`s3.dualstack.aws-region.amazonaws.com/bucketname`

エンドポイント名のスタイルの詳細については、「[Amazon S3 バケットに対するアクセスと一覧表示](#)」を参照してください。Amazon S3 エンドポイントのリストは、AWS 全般のリファレンスの「[リージョンとエンドポイント](#)」を参照してください。

#### Important

デュアルスタックのエンドポイントでは、Transfer Acceleration を使用できます。詳細については、「[Amazon S3 Transfer Acceleration の開始方法](#)」を参照してください。

#### Note

Amazon S3 へのアクセスに使用する 2 つのタイプの仮想プライベートクラウド (VPC) エンドポイント (インターフェイス VPC エンドポイントとゲートウェイ VPC エンドポイント) は、デュアルスタックをサポートするようになりました。Amazon S3 の VPC エンドポイントの詳細については、「[AWS PrivateLink for Amazon S3](#)」を参照してください。

AWS Command Line Interface (AWS CLI) や AWS SDK を使用する場合、パラメータまたはフラグを使ってデュアルスタックのエンドポイントに変更できます。設定ファイルの Amazon S3 エンドポイントに上書きしてデュアルスタックのエンドポイントを直接指定することもできます。以下のセクションでは、デュアルスタックのエンドポイントを AWS CLI と AWS SDK から使用方法を説明します。

## AWS CLI からのデュアルスタックのエンドポイントの使用

このセクションでは、デュアルスタックのエンドポイントへリクエストするのに使用される AWS CLI コマンドの例を示します。AWS CLI をセットアップする手順については、「[AWS CLI を使用した Amazon S3 での開発](#)」を参照してください。

AWS Config ファイルのプロファイル内で設定値 `use_dualstack_endpoint` を `true` に設定すると、`s3` と `s3api` AWS CLI コマンドによるすべての Amazon S3 リクエストが指定されたリージョンのデュアルスタックのエンドポイントに転送されます。`--region` オプションを使用して設定ファイルまたはコマンドでリージョンを指定します。

AWS CLI でデュアルスタックのエンドポイントを使用する場合、`path` と `virtual` の両方のアドレス形式がサポートされます。バケット名がホスト名または URL の一部にある場合、設定ファイルにあるアドレス形式が統制します。デフォルトでは、CLI は可能な限り仮想形式の使用を試みますが、必要に応じてパス形式に戻します。詳細については、[AWS CLI Amazon S3 の設定](#)を参照してください。

以下の例に示すように、デフォルトプロファイルで `use_dualstack_endpoint` を `true` に、また `addressing_style` を `virtual` に設定するようコマンドを使用して構成を変更することもできます。

```
$ aws configure set default.s3.use_dualstack_endpoint true
$ aws configure set default.s3.addressing_style virtual
```

すべてのコマンドではなく、指定した AWS CLI コマンドについてのみデュアルスタックのエンドポイントを使用するには、次のいずれかの方法を使用できます。

- コマンドでデュアルスタックのエンドポイントを使用する。その場合は、すべての `--endpoint-url` または `https://s3.dualstack.aws-region.amazonaws.com` コマンドで、`http://s3.dualstack.aws-region.amazonaws.com` パラメータを `s3` または `s3api` に設定します。

```
$ aws s3api list-objects --bucket bucketname --endpoint-url https://s3.dualstack.aws-region.amazonaws.com
```

- AWS Config ファイル内で別々のプロファイルを設定する。たとえば、`use_dualstack_endpoint` を `true` に設定するプロファイルと `use_dualstack_endpoint` を設定しないプロファイルを作成します。コマンドを実行する際には、デュアルスタックのエンドポイントを使用するかどうかによって、適切なプロファイルを指定します。



 Note

AWS CLI を使用していると、現在は、デュアルスタックのエンドポイントでは Transfer Acceleration を使用できません。ただし、AWS CLI は間もなくサポートされるようになります。詳細については、「[S3 Transfer Acceleration の有効化と使用](#)」を参照してください。

## AWS SDK からデュアルスタックのエンドポイントを使用する

このセクションでは、AWS SDK を使用してデュアルスタックのエンドポイントにアクセスする方法の例を示します。

### AWS SDK for Java のデュアルスタックのエンドポイントの例

次の例では、AWS SDK for Java を使用して Amazon S3 クライアントの作成時にデュアルスタック エンドポイントを有効化する方法を示します。

作業用の Java サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「[使用開始](#)」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

public class DualStackEndpoints {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            // Create an Amazon S3 client with dual-stack endpoints enabled.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .withDualstackEnabled(true)
                .build();
```



```
s3Client.listObjects(bucketName);
} catch (AmazonServiceException e) {
    // The call was transmitted successfully, but Amazon S3 couldn't process
    // it, so it returned an error response.
    e.printStackTrace();
} catch (SdkClientException e) {
    // Amazon S3 couldn't be contacted for a response, or the client
    // couldn't parse the response from Amazon S3.
    e.printStackTrace();
}
}
```

Windows で AWS SDK for Java を使用している場合は、必要に応じて、次の Java 仮想マシン (JVM) のプロパティを設定します。

```
java.net.preferIPv6Addresses=true
```

### AWS .NET SDK のデュアルスタックのエンドポイントの例

AWS SDK for .NET を使用するときは、次の例に示すように、デュアルスタックのエンドポイントの使用を有効にするため `AmazonS3Config` クラスを使用します。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DualStackEndpointTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            var config = new AmazonS3Config
            {
```

```
        UseDualstackEndpoint = true,
        RegionEndpoint = bucketRegion
    };
    client = new AmazonS3Client(config);
    Console.WriteLine("Listing objects stored in a bucket");
    ListingObjectsAsync().Wait();
}

private static async Task ListingObjectsAsync()
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 10
        };
        ListObjectsV2Response response;
        do
        {
            response = await client.ListObjectsV2Async(request);

            // Process the response.
            foreach (S3Object entry in response.S3Objects)
            {
                Console.WriteLine("key = {0} size = {1}",
                    entry.Key, entry.Size);
            }
            Console.WriteLine("Next Continuation Token: {0}",
response.NextContinuationToken);
            request.ContinuationToken = response.NextContinuationToken;
        } while (response.IsTruncated == true);
    }
    catch (AmazonS3Exception amazonS3Exception)
    {
        Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
amazonS3Exception.ToString());
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception: " + e.ToString());
    }
}
}
```

```
}
```

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「[AWS SDK for .NET の開始方法](#)」を参照してください。

#### REST API からのデュアルスタックのエンドポイントの使用

REST API を使用したデュアルスタックのエンドポイントへのリクエストについては、「[REST API を使用したデュアルスタックのエンドポイントへのリクエストの実行](#)」を参照してください。

# AWS SDK を使用したリクエストの実行

## トピック

- [AWS アカウント または IAM ユーザーの認証情報を使用したリクエストの実行](#)
- [IAM ユーザーの一時的な認証情報を使用したリクエストの実行](#)
- [フェデレーションユーザーの一時的な認証情報を使用したリクエストの実行](#)

認証リクエストを Amazon S3 に送信するには、AWS SDK を使用するか、アプリケーションで直接 REST API を呼び出します。AWS SDK API は、指定した証明書を使用して認証用の署名を計算します。アプリケーションで直接 REST API を使用する場合、リクエストを認証するために、署名の計算に必要なコードを記述する必要があります。利用可能な AWS SDK のリストについては、[サンプルコードとライブラリ](#)を参照してください。

## AWS アカウント または IAM ユーザーの認証情報を使用したリクエストの実行

AWS アカウント または IAM ユーザーのセキュリティ認証情報を使用して、認証リクエストを Amazon S3 に送信できます。このセクションでは、AWS SDK for Java、AWS SDK for .NET、および AWS SDK for PHP を使用して認証リクエストを送信する方法の例を示します。利用可能な AWS SDK のリストについては、[サンプルコードとライブラリ](#)を参照してください。

各 AWS SDK は、SDK 固有の認証情報プロバイダチェーンを使用して認証情報の検索を行い、その認証情報を使用して認証情報の所有者に代わってアクションを実行します。これらの認証情報プロバイダチェーンに共通していることは、ユーザーのローカルの AWS 認証情報ファイルを探すことです。

詳細については、以下のトピックをご参照ください。

## トピック

- [ローカルの AWS 認証情報ファイルを作成するには](#)
- [AWS SDK を使用した、認証されたリクエストの送信](#)
- [関連リソース](#)

## ローカルの AWS 認証情報ファイルを作成するには

AWS SDK 用の認証情報を設定する最も簡単な方法は、AWS 認証情報ファイルを使用することです。AWS Command Line Interface (AWS CLI) を使用している場合、既にローカルの AWS 認証情報

ファイルを設定済みかもしれません。そうでない場合は、以下の手順で認証情報ファイルを設定します。

1. AWS マネジメントコンソール にサインインして、IAM コンソール <https://console.aws.amazon.com/iam/> を開きます。
2. コードがアクセス許可を使用する必要があるサービスやアクションに限定した許可を持つ、新しいユーザーを作成します。新しいユーザーの作成方法の詳細については、「[IAM ユーザーの作成 \(コンソール\)](#)」を参照し、ステップ 8 までの手順に従ってください。
3. AWS 認証情報のローカルコピーを保存するには、[Download .csv] (.csv のダウンロード) を選択します。
4. コンピュータでホームディレクトリに移動し .aws ディレクトリを作成します。Linux または OS X のような Unix ベースのシステムでは、次の場所になります。

```
~/ .aws
```

Windows では、次の場所になります。

```
%HOMEPATH%\ .aws
```

5. .aws ディレクトリで `credentials` という名前の新しいファイルを作成します。
6. IAM コンソールからダウンロードした認証情報の .csv ファイルを開き、その内容を以下の形式で `credentials` ファイル内にコピーします。

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

7. `credentials` ファイルを保存し、ステップ 3 でダウンロードした .csv ファイルを削除します。

これで共有認証情報ファイルがローカルコンピュータで設定され、AWS SDK で使用する準備ができました。

AWS SDK を使用した、認証されたリクエストの送信

AWS SDK を使用して認証されたリクエストを送信します。認証済みリクエストの送信の詳細については、「[AWS セキュリティ認証情報](#)」または「[IAM Identity Center 認証](#)」を参照してください。

## Java

AWS SDK for Java を使用したリクエストの認証については、「AWS SDK およびツールリファレンスガイド」の「[共有ファイル config と credentials を使用して AWS SDK とツールをグローバルに設定する](#)」および「[AWS SDK とツールを使用した認証とアクセス](#)」を参照してください。

## .NET

AWS アカウント または IAM ユーザーの認証情報を使用して、認証済みのリクエストを送信するには

- AmazonS3Client クラスのインスタンスを作成します。
- AmazonS3Client メソッドのいずれかを実行して、Amazon S3 にリクエストを送信します。クライアントにより、指定した認証情報から必要な署名が生成されて、Amazon S3 に送信されるリクエストに追加されます。

詳細については、「[AWS アカウントまたは IAM ユーザー認証情報を使用したリクエストの実行](#)」を参照してください。

### Note

- AmazonS3Client クライアントは、セキュリティ認証情報を指定せずに作成できます。このクライアントを使用して送信されるリクエストは、署名なしの匿名リクエストです。公開されていないリソースに対して匿名リクエストを送信すると、Amazon S3 はエラーを返します。
- AWS アカウントを作成し、必要なユーザーを作成できます。また、これらのユーザーの認証情報も管理できます。以下の例のタスクを実行するときは、これらの認証情報が必要になります。詳細については、「SDK for .NET デベロッパーガイド」の「[AWS 認証情報の設定](#)」を参照してください。

その後、プロファイルと認証情報をアクティブに取得して、AWS のサービスのクライアントの作成時にそれら認証情報を明示的に使用するように、アプリケーションを設定することもできます。詳細については、SDK for .NET デベロッパーガイドの[アプリケーションの認証情報およびプロファイルへのアクセス](#)を参照してください。

次の C# の例では、前のタスクを実行する方法を示します。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「[AWS SDK for .NET の開始方法](#)」を参照してください。

## Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class MakeS3RequestTest
    {
        private const string bucketName = "**** bucket name ****";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
            RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            using (client = new AmazonS3Client(bucketRegion))
            {
                Console.WriteLine("Listing objects stored in a bucket");
                ListingObjectsAsync().Wait();
            }
        }

        static async Task ListingObjectsAsync()
        {
            try
            {
                ListObjectsRequest request = new ListObjectsRequest
                {
                    BucketName = bucketName,
                    MaxKeys = 2
                };
                do
                {

```

```
        ListObjectsResponse response = await
client.ListObjectsAsync(request);
        // Process the response.
        foreach (S3Object entry in response.S3Objects)
        {
            Console.WriteLine("key = {0} size = {1}",
                entry.Key, entry.Size);
        }

        // If the response is truncated, set the marker to get the next
        // set of keys.
        if (response.IsTruncated)
        {
            request.Marker = response.NextMarker;
        }
        else
        {
            request = null;
        }
    } while (request != null);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
}
catch (Exception e)
{
    Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
}
}
}
```

## PHP

このセクションでは、バージョン 3 の AWS SDK for PHP のクラスで AWS アカウント または IAM ユーザー認証情報を使用して、認証済みのリクエストを送信する方法について説明します。AWS SDK for Ruby API の詳細については、[AWS SDK for Ruby - バージョン 2](#) を参照してください。



次の PHP の例では、クライアントでセキュリティ認証情報を使用して、アカウントのバケットを一覧表示するリクエストを実行する方法を示します。

### Example

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

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
]);

// Retrieve the list of buckets.
$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // Print the list of objects to the page.
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

#### Note

S3Client クライアントは、セキュリティ認証情報を指定せずに作成できます。このクライアントを使用して送信されるリクエストは、署名なしの匿名リクエストです。公開さ

れていないリソースに対して匿名リクエストを送信すると、Amazon S3 はエラーを返します。詳細については、[AWS SDK for PHP ドキュメント](#)の[匿名クライアントの作成](#)を参照してください。

## Ruby

バージョン 3 の AWS SDK for Ruby を使用して Amazon S3 を呼び出す前に、SDK がバケットやオブジェクトへのアクセスの確認に使用する AWS アクセス認証情報を設定する必要があります。ローカルシステムの AWS 認証情報プロファイルで設定した認証情報を共有している場合、バージョン 3 の SDK for Ruby では、コード内で宣言することなく、これらの認証情報を使用できます。共有認証情報の設定の詳細については、「[AWS アカウントまたは IAM ユーザー認証情報を使用したリクエストの実行](#)」を参照してください。

次の Ruby コードスニペットは、ローカルコンピュータ上の共有 AWS 認証情報ファイルの認証情報を使用して、特定のバケット内のすべてのオブジェクトキー名を取得するリクエストを認証します。以下が実行されます。

1. `Aws::S3::Client` クラスのインスタンスを作成します。
2. `list_objects_v2` の `Aws::S3::Client` メソッドを使用してバケット内のオブジェクトを列挙することで、Amazon S3 へのリクエストを実行します。クライアントは、コンピュータ上の AWS 認証情報ファイルの認証情報から必要な署名値を生成し、それを Amazon S3 に送信するリクエストに含めます。
3. オブジェクトキー名の配列をターミナルに出力します。

## Example

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'amzn-s3-demo-bucket')
```

```
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts 'The object keys in this bucket are (first 50 objects):'
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts 'No objects found in this bucket.'
  end

  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  false
end

# Example usage:
def run_me
  region = 'us-west-2'
  bucket_name = 'BUCKET_NAME'
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

ローカルの AWS 認証情報ファイルがない場合でも、`Aws::S3::Client` リソースを作成し、Amazon S3 のバケットとオブジェクトに対してコードを実行できます。バージョン 3 の SDK for Ruby を使用して送信されるリクエストは匿名であり、デフォルトでは署名はありません。公開されていないリソースに対して匿名リクエストを送信すると、Amazon S3 はエラーを返します。

次のより堅牢な例のように、SDK for Ruby アプリケーション用の前述のコードスニペットを使用して展開することができます。

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'amzn-s3-demo-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts 'The object keys in this bucket are (first 50 objects):'
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts 'No objects found in this bucket.'
  end

  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  false
end

# Example usage:
def run_me
  region = 'us-west-2'
  bucket_name = 'BUCKET_NAME'
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

## Go

### Example

次の例では、SDK for Go が共有認証情報ファイルから自動的に読み込む AWS 認証情報を使用しています。

```
package main

import (
    "context"
    "errors"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    s3Client := s3.NewFromConfig(sdkConfig)
    count := 10
    fmt.Printf("Let's list up to %v buckets for your account.\n", count)
    result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        var ae smithy.APIError
        if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
            fmt.Println("You don't have permission to list buckets for this account.")
        }
    }
}
```

```
    } else {
        fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
    }
    return
}
if len(result.Buckets) == 0 {
    fmt.Println("You don't have any buckets!")
} else {
    if count > len(result.Buckets) {
        count = len(result.Buckets)
    }
    for _, bucket := range result.Buckets[:count] {
        fmt.Printf("\t%v\n", *bucket.Name)
    }
}
}
```

## 関連リソース

- [AWS SDK を使用した Amazon S3 での開発](#)
- [Amazon S3 向け AWS SDK for PHP Aws\S3\S3Client クラスの場合](#)
- [AWS SDK for PHP ドキュメント](#)

## IAM ユーザーの一時的な認証情報を使用したリクエストの実行

AWS アカウント または IAM ユーザーは、一時的なセキュリティ認証情報をリクエストし、そのセキュリティ認証情報を使用して、認証リクエストを Amazon S3 に送信できます。このセクションでは、AWS SDK for Java、.NET、および PHP を使用して一時セキュリティ認証情報を取得する方法と、Amazon S3 に対するリクエストの認証にその認証情報を使用する方法について説明します。

### Java

IAM ユーザーまたは AWS アカウント は、AWS SDK for Java を使用して一時的なセキュリティ認証情報をリクエストし ([リクエストの実行](#) を参照)、その認証情報を使用して Amazon S3 にアクセスできます。これらの認証情報は、指定したセッションの有効期間が過ぎると失効します。

デフォルトでは、セッションの有効期間は 1 時間です。IAM ユーザーの認証情報を使用する場合、一時的なセキュリティ認証情報をリクエストする期間を 15 分からロールの最大セッション期間までで指定できます。一時的なセキュリティ認証情報の詳細については、[IAM ユーザーガイド](#)の「IAM の一時的なセキュリティ認証情報」を参照してください。リクエストの実行の詳細については、「[リクエストの実行](#)」を参照してください。

一時的なセキュリティ認証情報を取得して Amazon S3 にアクセスするには

1. AWSSecurityTokenService クラスのインスタンスを作成します。
2. Security Token Service (STS) クライアントの `assumeRole()` メソッドを呼び出して、該当ロールの一時的なセキュリティ認証情報を取得します。
3. 一時的なセキュリティ認証情報を `BasicSessionCredentials` オブジェクトにパッケージ化します。このオブジェクトを使用して、一時的なセキュリティ認証情報を Amazon S3 クライアントに提供します。
4. 一時的なセキュリティ認証情報を使用して、`AmazonS3Client` クラスのインスタンスを作成します。このクライアントを使用してリクエストを Amazon S3 に送信します。失効した認証情報を使用してリクエストを送信すると、Amazon S3 はエラーを返します。

#### Note

AWS アカウント のセキュリティ認証情報を使用して一時的なセキュリティ認証情報を取得した場合、この認証情報は 1 時間だけ有効です。セッションの有効期間を指定できるのは、セッションのリクエストに IAM ユーザー証明書を使用する場合のみです。

次の例では、指定したバケット内のオブジェクトキーを一覧表示します。この例では、セッションの一時的なセキュリティ認証情報を取得し、これを使用して認証済みのリクエストを Amazon S3 に送信します。

IAM ユーザーの認証情報を使用してサンプルをテストする場合は、AWS アカウントで IAM ユーザーを作成する必要があります。IAM ユーザーを作成する方法の詳細については、「IAM ユーザーガイド」の「[最初の IAM ユーザーおよび管理者グループの作成](#)」を参照してください。

作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「[使用開始](#)」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.AssumeRoleRequest;
import com.amazonaws.services.securitytoken.model.AssumeRoleResult;
import com.amazonaws.services.securitytoken.model.Credentials;

public class MakingRequestsWithIAMTempCredentials {
    public static void main(String[] args) {
        String clientRegion = "**** Client region ****";
        String roleARN = "**** ARN for role to be assumed ****";
        String roleSessionName = "**** Role session name ****";
        String bucketName = "**** Bucket name ****";

        try {
            // Creating the STS client is part of your trusted code. It has
            // the security credentials you use to obtain temporary security
            credentials.
            AWSSecurityTokenService stsClient =
            AWSSecurityTokenServiceClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();
```



```
role of    // Obtain credentials for the IAM role. Note that you cannot assume the
           // an AWS root account;
           // Amazon S3 will deny access. You must use credentials for an IAM user
or an      // IAM role.
           AssumeRoleRequest roleRequest = new AssumeRoleRequest()
               .withRoleArn(roleARN)
               .withRoleSessionName(roleSessionName);
           AssumeRoleResult roleResponse = stsClient.assumeRole(roleRequest);
           Credentials sessionCredentials = roleResponse.getCredentials();

           // Create a BasicSessionCredentials object that contains the credentials
you         // just retrieved.
           BasicSessionCredentials awsCredentials = new BasicSessionCredentials(
               sessionCredentials.getAccessKeyId(),
               sessionCredentials.getSecretAccessKey(),
               sessionCredentials.getSessionToken());

           // Provide temporary security credentials so that the Amazon S3 client
           // can send authenticated requests to Amazon S3. You create the client
           // using the sessionCredentials object.
           AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
               .withCredentials(new
AWSStaticCredentialsProvider(awsCredentials))
               .withRegion(clientRegion)
               .build();

           // Verify that assuming the role worked and the permissions are set
correctly   // by getting a set of object keys from the bucket.
           ObjectListing objects = s3Client.listObjects(bucketName);
           System.out.println("No. of Objects: " +
objects.getObjectSummaries().size());
           } catch (AmazonServiceException e) {
               // The call was transmitted successfully, but Amazon S3 couldn't process
               // it, so it returned an error response.
               e.printStackTrace();
           } catch (SdkClientException e) {
               // Amazon S3 couldn't be contacted for a response, or the client
               // couldn't parse the response from Amazon S3.
               e.printStackTrace();
           }
}
```

```
}  
}
```

## .NET

IAM ユーザーまたは AWS アカウント は、AWS SDK for .NET を使用して一時的なセキュリティ認証情報をリクエストし、その認証情報を使用して Amazon S3 にアクセスできます。これらの認証情報は、セッションの有効期間が過ぎると失効します。

デフォルトでは、セッションの有効期間は 1 時間です。IAM ユーザーの認証情報を使用する場合、一時的なセキュリティ認証情報をリクエストする期間を 15 分からロールの最大セッション期間までで指定できます。一時的なセキュリティ認証情報の詳細については、[IAM ユーザーガイド](#)の「IAM の一時的なセキュリティ認証情報」を参照してください。リクエストの実行の詳細については、「[リクエストの実行](#)」を参照してください。

一時的なセキュリティ認証情報を取得して Amazon S3 にアクセスするには

1. AWS Security Token Service クライアント `AmazonSecurityTokenServiceClient` のインスタンスを作成します。
2. 前の手順で作成した STS クライアントの `GetSessionToken` メソッドを呼び出して、セッションを開始します。 `GetSessionTokenRequest` オブジェクトを使用して、このメソッドにセッション情報を指定します。

このメソッドは一時的なセキュリティ認証情報を返します。

3. `SessionAWSCredentials` オブジェクトのインスタンスに一時セキュリティ証明書をパッケージ化します。このオブジェクトを使用して、一時的なセキュリティ認証情報を Amazon S3 クライアントに提供します。
4. 一時セキュリティ証明書を渡して、`AmazonS3Client` クラスのインスタンスを作成します。このクライアントを使用してリクエストを Amazon S3 に送信します。失効した認証情報を使用してリクエストを送信すると、Amazon S3 はエラーを返します。

### Note

AWS アカウント のセキュリティ認証情報を使用して一時的なセキュリティ認証情報を取得した場合、この認証情報は 1 時間だけ有効です。セッションの有効期間を指定でき

るのは、IAM ユーザーの認証情報を使用してセッションをリクエストする場合に限ります。

次の C# の例では、指定したバケットのオブジェクトキーを一覧表示します。この例では、わかりやすいように、デフォルトの 1 時間のセッションの一時セキュリティ認証情報を取得し、それを使用して認証リクエストを Amazon S3 に送信しています。

IAM ユーザーの認証情報を使用してサンプルをテストする場合は、AWS アカウントで IAM ユーザーを作成する必要があります。IAM ユーザーを作成する方法の詳細については、「IAM ユーザーガイド」の「[最初の IAM ユーザーおよび管理者グループの作成](#)」を参照してください。リクエストの実行の詳細については、「[リクエストの実行](#)」を参照してください。

コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「[AWS SDK for .NET の開始方法](#)」を参照してください。

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempCredExplicitSessionStartTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
            RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {

```

```

        try
        {
            // Credentials use the default AWS SDK for .NET credential search
chain.
            // On local development machines, this is your default profile.
            Console.WriteLine("Listing objects stored in a bucket");
            SessionAWSCredentials tempCredentials = await
GetTemporaryCredentialsAsync();

            // Create a client by providing temporary security credentials.
            using (s3Client = new AmazonS3Client(tempCredentials, bucketRegion))
            {
                var listObjectRequest = new ListObjectsRequest
                {
                    BucketName = bucketName
                };
                // Send request to Amazon S3.
                ListObjectsResponse response = await
s3Client.ListObjectsAsync(listObjectRequest);
                List<S3Object> objects = response.S3Objects;
                Console.WriteLine("Object count = {0}", objects.Count);
            }
        }
        catch (AmazonS3Exception s3Exception)
        {
            Console.WriteLine(s3Exception.Message, s3Exception.InnerException);
        }
        catch (AmazonSecurityTokenServiceException stsException)
        {
            Console.WriteLine(stsException.Message,
stsException.InnerException);
        }
    }

    private static async Task<SessionAWSCredentials>
GetTemporaryCredentialsAsync()
    {
        using (var stsClient = new AmazonSecurityTokenServiceClient())
        {
            var getSessionTokenRequest = new GetSessionTokenRequest
            {
                DurationSeconds = 7200 // seconds
            };

```

```
        GetSessionTokenResponse sessionTokenResponse =  
            await  
stsClient.GetSessionTokenAsync(getSessionTokenRequest);  
  
        Credentials credentials = sessionTokenResponse.Credentials;  
  
        var sessionCredentials =  
            new SessionAWSCredentials(credentials.AccessKeyId,  
                                       credentials.SecretAccessKey,  
                                       credentials.SessionToken);  
  
        return sessionCredentials;  
    }  
}  
}
```

## PHP

AWS SDK for Ruby API の詳細については、[AWS SDK for Ruby - バージョン 2](#) を参照してください。

IAM ユーザーまたは AWS アカウント は、バージョン 3 の AWS SDK for PHP を使用して一時的なセキュリティ認証情報をリクエストできます。次に、この一時的な認証情報を使用して Amazon S3 にアクセスできます。認証情報は、セッションの有効期間が失効すると同時に失効します。

デフォルトでは、セッションの有効期間は 1 時間です。IAM ユーザーの認証情報を使用する場合、一時的なセキュリティ認証情報をリクエストする期間を 15 分からロールの最大セッション期間までで指定できます。一時的なセキュリティ認証情報の詳細については、[IAM ユーザーガイド](#)の「IAM の一時的なセキュリティ認証情報」を参照してください。リクエストの実行の詳細については、「[リクエストの実行](#)」を参照してください。

### Note

AWS アカウント のセキュリティ認証情報を使用して一時的なセキュリティ認証情報を取得する場合、取得した一時的なセキュリティ認証情報は 1 時間だけ有効です。セッションの有効期間を指定できるのは、セッションのリクエストに IAM ユーザー証明書を使用する場合のみです。

## Example

次の PHP の例では、一時的なセキュリティ認証情報を使用して、指定したバケットのオブジェクトキーを一覧表示します。この例では、一時的なセキュリティ認証情報をデフォルトの 1 時間のセッションとして取得し、それを使用して認証済みのリクエストを Amazon S3 に送信します。AWS SDK for Ruby API の詳細については、[AWS SDK for Ruby - バージョン 2](#) を参照してください。

IAM ユーザーの認証情報を使用して例をテストする場合は、AWS アカウントで IAM ユーザーを作成する必要があります。IAM ユーザーを作成する方法については、「IAM ユーザーガイド」の「[最初の IAM ユーザーおよび管理者グループの作成](#)」を参照してください。IAM ユーザーの認証情報を使用してセッションをリクエストするときにセッションの有効期間を設定する例については、「[IAM ユーザーの一時的な認証情報を使用したリクエストの実行](#)」を参照してください。

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

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

$sessionToken = $sts->getSessionToken();

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

$result = $s3->listBuckets();

try {
```

```
// Retrieve a paginator for listing objects.
$objects = $s3->getPaginator('ListObjects', [
    'Bucket' => $bucket
]);

echo "Keys retrieved!" . PHP_EOL;

// List objects
foreach ($objects as $object) {
    echo $object['Key'] . PHP_EOL;
}
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

## Ruby

IAM ユーザーまたは AWS アカウント は、AWS SDK for Ruby を使用して一時的なセキュリティ認証情報をリクエストし、その認証情報を使用して Amazon S3 にアクセスできます。これらの認証情報は、セッションの有効期間が過ぎると失効します。

デフォルトでは、セッションの有効期間は 1 時間です。IAM ユーザーの認証情報を使用する場合、一時的なセキュリティ認証情報をリクエストする期間を 15 分からロールの最大セッション期間までで指定できます。一時的なセキュリティ認証情報の詳細については、[IAM ユーザーガイド](#)の「IAM の一時的なセキュリティ認証情報」を参照してください。リクエストの実行の詳細については、「[リクエストの実行](#)」を参照してください。

### Note

AWS アカウント のセキュリティ認証情報を使用して一時的なセキュリティ認証情報を取得する場合、取得した一時的なセキュリティ認証情報は 1 時間だけ有効です。セッションの有効期間を指定できるのは、セッションのリクエストに IAM ユーザー認証情報を使用する場合のみです。

次の Ruby の例では、有効期間が 1 時間の一時的なユーザーを作成し、指定したバケットの項目を一覧表示します。この例を使用するには、新しい AWS Security Token Service (AWS STS) クライアントを作成し、Amazon S3 バケットを一覧表示するための許可が AWS 認証情報が必要です。

```
# Prerequisites:
# - A user in AWS Identity and Access Management (IAM). This user must
#   be able to assume the following IAM role. You must run this code example
#   within the context of this user.
# - An existing role in IAM that allows all of the Amazon S3 actions for all of the
#   resources in this code example. This role must also trust the preceding IAM
#   user.
# - An existing S3 bucket.

require 'aws-sdk-core'
require 'aws-sdk-s3'
require 'aws-sdk-iam'

# Checks whether a user exists in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Boolean] true if the user exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless user_exists?(iam_client, 'my-user')
def user_exists?(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return true if response.user.user_name
rescue Aws::IAM::Errors::NoSuchEntity
  # User doesn't exist.
rescue StandardError => e
  puts "Error while determining whether the user ' \
    '#{user_name}' exists: #{e.message}"
end

# Creates a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS::IAM::Types::User] The new user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = create_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def create_user(iam_client, user_name)
  response = iam_client.create_user(user_name: user_name)
```



```

    response.user
  rescue StandardError => e
    puts "Error while creating the user '#{user_name}': #{e.message}"
  end

  # Gets a user in IAM.
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  # @param user_name [String] The user's name.
  # @return [AWS::IAM::Types::User] The existing user.
  # @example
  #   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
  #   user = get_user(iam_client, 'my-user')
  #   exit 1 unless user.user_name
  def get_user(iam_client, user_name)
    response = iam_client.get_user(user_name: user_name)
    response.user
  rescue StandardError => e
    puts "Error while getting the user '#{user_name}': #{e.message}"
  end

  # Checks whether a role exists in IAM.
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  # @param role_name [String] The role's name.
  # @return [Boolean] true if the role exists; otherwise, false.
  # @example
  #   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
  #   exit 1 unless role_exists?(iam_client, 'my-role')
  def role_exists?(iam_client, role_name)
    response = iam_client.get_role(role_name: role_name)
    return true if response.role.role_name
  rescue StandardError => e
    puts "Error while determining whether the role ' \
      '#{role_name}' exists: #{e.message}"
  end

  # Gets credentials for a role in IAM.
  #
  # @param sts_client [Aws::STS::Client] An initialized AWS STS client.
  # @param role_arn [String] The role's Amazon Resource Name (ARN).
  # @param role_session_name [String] A name for this role's session.
  # @param duration_seconds [Integer] The number of seconds this session is valid.
  # @return [AWS::AssumeRoleCredentials] The credentials.

```

```

# @example
#   sts_client = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_credentials(
#     sts_client,
#     'arn:aws:iam::123456789012:role/AmazonS3ReadOnly',
#     'ReadAmazonS3Bucket',
#     3600
#   )
#   exit 1 if credentials.nil?
def get_credentials(sts_client, role_arn, role_session_name, duration_seconds)
  Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: role_session_name,
    duration_seconds: duration_seconds
  )
rescue StandardError => e
  puts "Error while getting credentials: #{e.message}"
end

# Checks whether a bucket exists in Amazon S3.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The name of the bucket.
# @return [Boolean] true if the bucket exists; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless bucket_exists?(s3_client, 'amzn-s3-demo-bucket')
def bucket_exists?(s3_client, bucket_name)
  response = s3_client.list_buckets
  response.buckets.each do |bucket|
    return true if bucket.name == bucket_name
  end
rescue StandardError => e
  puts "Error while checking whether the bucket '#{bucket_name}' " \
    "exists: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example

```

```
# s3_client = Aws::S3::Client.new(region: 'us-west-2')
# exit 1 unless list_objects_in_bucket?(s3_client, 'amzn-s3-demo-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts 'Name => ETag'
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end
```

## フェデレーションユーザーの一時的な認証情報を使用したリクエストの実行

一時的なセキュリティ認証情報をリクエストし、AWS リソースにアクセスする必要があるフェデレーションユーザーまたはアプリケーションにその認証情報を提供できます。このセクションでは、AWS SDK を使用して、フェデレーションユーザーまたはアプリケーションのために一時セキュリティ証明書を取得し、その証明書を使用して認証リクエストを Amazon S3 に送信する方法の例について説明します。利用可能な AWS SDK のリストについては、[サンプルコードとライブラリ](#)を参照してください。

### Note

AWS アカウント と IAM ユーザーのいずれも、フェデレーティッドユーザーの一時的なセキュリティ認証情報をリクエストできます。ただし、より高度なセキュリティのためには、必要な権限を持つ IAM ユーザーのみが、この一時証明書をリクエストできるようにすることをお勧めします。こうすることで、フェデレーションユーザーに付与される権限が、リクエストを送信した IAM ユーザーの権限以下となります。アプリケーションによっては、フェデレーティッドユーザーおよびアプリケーションに一時的なセキュリティ認証情報を付与することを唯一の目的として、特定のアクセス許可を持つ IAM ユーザーを作成する方が適切な場合もあります。

## Java

フェデレーティッドユーザーおよびアプリケーションから AWS リソースへのアクセスのリクエストが認証されて送信されるように、フェデレーティッドユーザーおよびアプリケーションに一時的なセキュリティ認証情報を提供することができます。このような一時的な認証情報をリクエストする場合、ユーザー名と、付与するリソースに対するアクセス許可を示す IAM ポリシーを指定する必要があります。デフォルトでは、セッションの有効期間は 1 時間です。フェデレーションユーザーおよびアプリケーション用に一時的なセキュリティ認証情報をリクエストするときに、明示的に別の有効期間値を設定できます。

### Note

フェデレーティッドユーザーおよびアプリケーション用の一時的なセキュリティ認証情報をリクエストする際のセキュリティ強化策として、必要なアクセス許可のみを持つ専用の IAM ユーザーを使用することをお勧めします。作成した一時ユーザーに、一時的なセキュリティ認証情報をリクエストした IAM ユーザーより多くの権限が付与されることは

ありません。詳細については、「[AWS Identity and Access Management に関するよくある質問](#)」を参照してください。

セキュリティ認証情報を提供し、認証されたリクエストを送信してリソースにアクセスするには、以下の操作を行います。

- `AWSecurityTokenServiceClient` クラスのインスタンスを作成します。
- Security Token Service (STS) クライアントの `getFederationToken()` メソッドを呼び出してセッションを開始します。一時的な認証情報にアタッチするユーザー名と IAM ポリシーなどのセッション情報を指定します。必要に応じて、セッションの有効期間を指定できます。このメソッドは一時的なセキュリティ認証情報を返します。
- `BasicSessionCredentials` オブジェクトのインスタンスに一時セキュリティ証明書をパッケージ化します。このオブジェクトを使用して、一時的なセキュリティ認証情報を Amazon S3 クライアントに提供します。
- 一時的なセキュリティ認証情報を使用して、`AmazonS3Client` クラスのインスタンスを作成します。このクライアントを使用してリクエストを Amazon S3 に送信します。失効した認証情報を使用してリクエストを送信すると、Amazon S3 はエラーを返します。

## Example

例では、指定された S3 バケットのキーをリストしています。この例では、フェデレーティッドユーザー用に 2 時間のセッションの一時的なセキュリティ認証情報を取得し、それを使用して認証リクエストを Amazon S3 に送信します。例を実行するには、ユーザーに一時的なセキュリティ認証情報のリクエストと AWS リソースの一覧表示を許可するアタッチされたポリシーで、IAM ユーザーを作成する必要があります。次のポリシーでは、これが達成されます。

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

IAM ユーザーを作成する方法の詳細については、「IAM ユーザーガイド」の「[最初の IAM ユーザーおよび管理者グループの作成](#)」を参照してください。

IAM ユーザーを作成し、前述のポリシーをアタッチしたら、以下の例を実行できます。作業サンプルの作成およびテストの手順については、「AWS SDK for Java のデベロッパーガイド」の「[使用開始](#)」を参照してください。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;

import java.io.IOException;

public class MakingRequestsWithFederatedTempCredentials {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Specify bucket name ***";
        String federatedUser = "*** Federated user name ***";
        String resourceARN = "arn:aws:s3:::" + bucketName;

        try {
            AWSSecurityTokenService stsClient = AWSSecurityTokenServiceClientBuilder
                .standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
```

```
        .build();

        GetFederationTokenRequest getFederationTokenRequest = new
GetFederationTokenRequest();
        getFederationTokenRequest.setDurationSeconds(7200);
        getFederationTokenRequest.setName(federatedUser);

        // Define the policy and add it to the request.
        Policy policy = new Policy();
        policy.withStatements(new Statement(Effect.Allow)
            .withActions(S3Actions.ListObjects)
            .withResources(new Resource(resourceARN)));
        getFederationTokenRequest.setPolicy(policy.toJson());

        // Get the temporary security credentials.
        GetFederationTokenResult federationTokenResult =
stsClient.getFederationToken(getFederationTokenRequest);
        Credentials sessionCredentials = federationTokenResult.getCredentials();

        // Package the session credentials as a BasicSessionCredentials
        // object for an Amazon S3 client object to use.
        BasicSessionCredentials basicSessionCredentials = new
BasicSessionCredentials(
            sessionCredentials.getAccessKeyId(),
            sessionCredentials.getSecretAccessKey(),
            sessionCredentials.getSessionToken());
        AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
            .withCredentials(new
AWSStaticCredentialsProvider(basicSessionCredentials))
            .withRegion(clientRegion)
            .build();

        // To verify that the client works, send a listObjects request using
        // the temporary security credentials.
        ObjectListing objects = s3Client.listObjects(bucketName);
        System.out.println("No. of Objects = " +
objects.getObjectSummaries().size());
    } catch (AmazonServiceException e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    } catch (SdkClientException e) {
        // Amazon S3 couldn't be contacted for a response, or the client
        // couldn't parse the response from Amazon S3.
    }
}
```

```
        e.printStackTrace();
    }
}
```

## .NET

フェデレーティッドユーザーおよびアプリケーションから AWS リソースへのアクセスのリクエストが認証されて送信されるように、フェデレーティッドユーザーおよびアプリケーションに一時的なセキュリティ認証情報を提供することができます。このような一時的な認証情報をリクエストする場合、ユーザー名と、付与するリソースに対するアクセス許可を示す IAM ポリシーを指定する必要があります。デフォルトでは、セッションの有効期間は 1 時間です。フェデレーションユーザーおよびアプリケーション用に一時的なセキュリティ認証情報をリクエストするときに、明示的に別の有効期間値を設定できます。認証されたリクエストを送信する方法については、「[リクエストの実行](#)」を参照してください。

### Note

フェデレーティッドユーザーおよびアプリケーション用の一時的なセキュリティ認証情報をリクエストする際のセキュリティ強化策として、必要なアクセス許可のみを持つ専用の IAM ユーザーを使用することをお勧めします。作成した一時ユーザーに、一時的なセキュリティ認証情報をリクエストした IAM ユーザーより多くの権限が付与されることはありません。詳細については、「[AWS Identity and Access Management に関するよくある質問](#)」を参照してください。

次の作業を行います。

- AWS Security Token Service クライアント `AmazonSecurityTokenServiceClient` クラスのインスタンスを作成します。
- STS クライアントの `GetFederationToken` メソッドを呼び出してセッションを開始します。一時的な認証情報にアタッチするユーザー名と IAM ポリシーなどのセッション情報を指定する必要があります。必要に応じて、セッションの有効期間を指定できます。このメソッドは一時的なセキュリティ認証情報を返します。
- `SessionAWSCredentials` オブジェクトのインスタンスに一時セキュリティ証明書をパッケージ化します。このオブジェクトを使用して、一時的なセキュリティ認証情報を Amazon S3 クライアントに提供します。



- 一時的なセキュリティ認証情報を渡して、AmazonS3Client クラスのインスタンスを作成します。このクライアントを使用してリクエストを Amazon S3 に送信します。失効した認証情報を使用してリクエストを送信すると、Amazon S3 はエラーを返します。

## Example

次の C# の例では、指定したバケットのキーをリストします。この例では、フェデレーテッドユーザー (User1) 用に 2 時間のセッションの一時的なセキュリティ認証情報を取得し、それを使用して認証リクエストを Amazon S3 に送信します。

- この演習では、最小限のアクセス許可を持つ IAM ユーザーを作成します。この IAM ユーザーの認証情報を使用して、他のユーザーの一時的な認証情報をリクエストします。この例では、特定のバケットに含まれるオブジェクトのみ表示されています。以下のポリシーをアタッチして IAM ユーザーを作成します。

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

このポリシーでは、IAM ユーザーが一時的なセキュリティ認証情報と、AWS リソースを一覧表示できるだけの許可をリクエストすることを許可しています。IAM ユーザーを作成する方法の詳細については、「IAM ユーザーガイド」の「[IAM ユーザーおよび管理者グループの作成](#)」を参照してください。

- IAM ユーザーのセキュリティ認証情報を使用して、以下の例をテストします。この例では、一時的なセキュリティ認証情報を使用して、認証リクエストを Amazon S3 に送信します。この例では、フェデレーテッドユーザー (User1) 用に、アクセスを特定のバケット (YourBucketName) 内のオブジェクトの列挙に制限する一時的なセキュリティ認証情報をリクエストするときに、以下のポリシーを指定します。ポリシーを更新し、独自の既存のバケット名を指定する必要があります。

```
{
  "Statement": [
```

```
{
  "Sid": "1",
  "Action": ["s3:ListBucket"],
  "Effect": "Allow",
  "Resource": "arn:aws:s3:::YourBucketName"
}
]
```

- Example

以下の例を編集して、前述のフェデレーティッドユーザーアクセスポリシーで指定したバケット名を指定します。コード例を設定および実行する方法の詳細については、「AWS SDK for .NET デベロッパーガイド」の「[AWS SDK for .NET の開始方法](#)」を参照してください。

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempFederatedCredentialsTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
            RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
```

```
{
    Console.WriteLine("Listing objects stored in a bucket");
    // Credentials use the default AWS SDK for .NET credential search
chain.
    // On local development machines, this is your default profile.
    SessionAWSCredentials tempCredentials =
        await GetTemporaryFederatedCredentialsAsync();

    // Create a client by providing temporary security credentials.
    using (client = new AmazonS3Client(bucketRegion))
    {
        ListObjectsRequest listObjectRequest = new
ListObjectsRequest();
        listObjectRequest.BucketName = bucketName;

        ListObjectsResponse response = await
client.ListObjectsAsync(listObjectRequest);
        List<S3Object> objects = response.S3Objects;
        Console.WriteLine("Object count = {0}", objects.Count);

        Console.WriteLine("Press any key to continue...");
        Console.ReadKey();
    }
}
catch (AmazonS3Exception e)
{
    Console.WriteLine("Error encountered ***. Message:'{0}' when
writing an object", e.Message);
}
catch (Exception e)
{
    Console.WriteLine("Unknown encountered on server. Message:'{0}'
when writing an object", e.Message);
}
}

private static async Task<SessionAWSCredentials>
GetTemporaryFederatedCredentialsAsync()
{
    AmazonSecurityTokenServiceConfig config = new
AmazonSecurityTokenServiceConfig();
    AmazonSecurityTokenServiceClient stsClient =
        new AmazonSecurityTokenServiceClient(
            config);
```

```

        GetFederationTokenRequest federationTokenRequest =
            new GetFederationTokenRequest();
        federationTokenRequest.DurationSeconds = 7200;
        federationTokenRequest.Name = "User1";
        federationTokenRequest.Policy = @"{
            ""Statement"":
            [
                {
                    ""Sid"":""Stmt1311212314284"",
                    ""Action"":[""s3:ListBucket""],
                    ""Effect"":""Allow"",
                    ""Resource"":""arn:aws:s3:::" + bucketName + @""
                }
            ]
        }
";

        GetFederationTokenResponse federationTokenResponse =
            await
stsClient.GetFederationTokenAsync(federationTokenRequest);
        Credentials credentials = federationTokenResponse.Credentials;

        SessionAWSCredentials sessionCredentials =
            new SessionAWSCredentials(credentials.AccessKeyId,
                                      credentials.SecretAccessKey,
                                      credentials.SessionToken);

        return sessionCredentials;
    }
}

```

## PHP

このトピックでは、バージョン 3 の AWS SDK for PHP のクラスを使用して、フェデレーティッドユーザーおよびアプリケーションの一時的なセキュリティ認証情報をリクエストし、その認証情報を使用して Amazon S3 に保存されているリソースにアクセスする手順を示します。AWS SDK for Ruby API の詳細については、[AWS SDK for Ruby - バージョン 2](#) を参照してください。

フェデレーティッドユーザーおよびアプリケーションから AWS リソースへのアクセスをリクエストできるように、一時的なセキュリティ認証情報を提供することができます。このような一時的な認証情報をリクエストする場合、ユーザー名と、付与するリソースに対するアクセス許可を

示す IAM ポリシーを指定する必要があります。これらの認証情報は、セッションの有効期間が失効すると同時に失効します。デフォルトでは、セッションの有効期間は 1 時間です。フェデレーテッドユーザーおよびアプリケーション用に一時的なセキュリティ認証情報をリクエストするときに、明示的に有効期間の別の値を設定できます。一時的なセキュリティ認証情報の詳細については、[IAM ユーザーガイド](#)の「IAM の一時的なセキュリティ認証情報」を参照してください。フェデレーテッドユーザーおよびアプリケーションに一時的なセキュリティ認証情報を指定する方法については、「[リクエストの実行](#)」を参照してください。

フェデレーテッドユーザーおよびアプリケーション用の一時的なセキュリティ認証情報をリクエストする際のセキュリティ強化策として、必要なアクセス許可のみを持つ専用の IAM ユーザーを使用することをお勧めします。作成した一時ユーザーに、一時的なセキュリティ認証情報をリクエストした IAM ユーザーより多くの権限が付与されることはありません。ID フェデレーションについて詳しくは、「[AWS Identity and Access Management に関するよくある質問](#)」を参照してください。

AWS SDK for Ruby API の詳細については、[AWS SDK for Ruby - バージョン 2](#) を参照してください。

### Example

次の PHP の例では、指定したバケットのキーをリストします。この例では、フェデレーテッドユーザー (User1) 用に 1 時間のセッションの一時的なセキュリティ認証情報を取得します。次に、その一時的なセキュリティ認証情報を使用して、認証リクエストを Amazon S3 に送信します。

他のユーザー用に一時的な認証情報をリクエストする場合、セキュリティ強化策として、一時的なセキュリティ認証情報をリクエストするアクセス許可を持つ IAM ユーザーのセキュリティ認証情報を使用します。また、IAM ユーザーがフェデレーテッドユーザーに最小限のアプリケーション固有のアクセス許可しか付与できないように、この IAM ユーザーのアクセス許可を制限することができます。この例では、特定のバケットに含まれるオブジェクトのみ表示されています。以下のポリシーをアタッチして IAM ユーザーを作成します。

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

```
]
}
```

このポリシーでは、IAM ユーザーが一時的なセキュリティ認証情報と、AWS リソースを一覧表示できるだけの許可をリクエストすることを許可しています。IAM ユーザーを作成する方法の詳細については、「IAM ユーザーガイド」の「[最初の IAM ユーザーおよび管理者グループの作成](#)」を参照してください。

これで、IAM ユーザーのセキュリティ認証情報を使用して、以下の例をテストできるようになりました。この例では、一時的なセキュリティ認証情報を使用して、認証リクエストを Amazon S3 に送信します。この例では、フェデレーティッドユーザー (User1) 用に、アクセスを特定のバケット内のオブジェクトの列挙に制限する一時的なセキュリティ認証情報をリクエストするときに、以下のポリシーを指定します。ポリシーは、使用しているバケット名で更新してください。

```
{
  "Statement": [
    {
      "Sid": "1",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::YourBucketName"
    }
  ]
}
```

以下の例では、ポリシーリソースを指定するときに、YourBucketName を、使用しているバケット名で置き換えます。

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

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

// In real applications, the following code is part of your trusted code. It has
// the security credentials that you use to obtain temporary security credentials.
$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
```

```

]);

// Fetch the federated credentials.
$sessionToken = $sts->getFederationToken([
    'Name' => 'User1',
    'DurationSeconds' => '3600',
    'Policy' => json_encode([
        'Statement' => [
            'Sid' => 'randomstatementid' . time(),
            'Action' => ['s3:ListBucket'],
            'Effect' => 'Allow',
            'Resource' => 'arn:aws:s3:::' . $bucket
        ]
    ])
]);

// The following will be part of your less trusted code. You provide temporary
// security credentials so the code can send authenticated requests to Amazon S3.

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

try {
    $result = $s3->listObjects([
        'Bucket' => $bucket
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}

```

## Ruby

フェデレーティッドユーザーおよびアプリケーションから AWS リソースへのアクセスのリクエストが認証されて送信されるように、フェデレーティッドユーザーおよびアプリケーションに一時的なセキュリティ認証情報を提供することができます。IAM サービスに対して一時的な認証

情報をリクエストする場合は、ユーザー名と、リソースに対するアクセス許可を付与するための IAM ポリシーを指定する必要があります。デフォルトでは、セッションの有効期間は 1 時間です。ただし、IAM ユーザー認証情報を使用して一時的な認証情報をリクエストする場合、フェデレーションユーザーおよびアプリケーション用の一時的なセキュリティ認証情報をリクエストするときに、別の有効期間値を明示的に設定できます。フェデレーティッドユーザーおよびアプリケーションの一時的なセキュリティ認証情報については、「[リクエストの実行](#)」を参照してください。

 Note

フェデレーティッドユーザーおよびアプリケーションの一時的なセキュリティ認証情報をリクエストする場合、セキュリティ強化策として、専用の IAM ユーザーを使用して必要なアクセス許可のみを付与できます。作成した一時ユーザーに、一時的なセキュリティ認証情報をリクエストした IAM ユーザーより多くの権限が付与されることはありません。詳細については、「[AWS Identity and Access Management に関するよくある質問](#)」を参照してください。

## Example

次の Ruby コード例では、フェデレーティッドユーザーに付与するアクセス許可を制限した上で、指定したバケットのキーを一覧表示することを許可します。

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'
require 'aws-sdk-iam'
require 'json'

# Checks to see whether a user exists in IAM; otherwise,
# creates the user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Aws::IAM::Types::User] The existing or new user.
# @example
#   iam = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam, 'my-user')
#   exit 1 unless user.user_name
#   puts "User's name: #{user.user_name}"
```



```

def get_user(iam, user_name)
  puts "Checking for a user with the name '#{user_name}'..."
  response = iam.get_user(user_name: user_name)
  puts "A user with the name '#{user_name}' already exists."
  response.user
# If the user doesn't exist, create them.
rescue Aws::IAM::Errors::NoSuchEntity
  puts "A user with the name '#{user_name}' doesn't exist. Creating this user..."
  response = iam.create_user(user_name: user_name)
  iam.wait_until(:user_exists, user_name: user_name)
  puts "Created user with the name '#{user_name}'."
  response.user
rescue StandardError => e
  puts "Error while accessing or creating the user named '#{user_name}':
#{e.message}"
end

# Gets temporary AWS credentials for an IAM user with the specified permissions.
#
# @param sts [Aws::STS::Client] An initialized AWS STS client.
# @param duration_seconds [Integer] The number of seconds for valid credentials.
# @param user_name [String] The user's name.
# @param policy [Hash] The access policy.
# @return [Aws::STS::Types::Credentials] AWS credentials for API authentication.
# @example
#   sts = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_temporary_credentials(sts, duration_seconds, user_name,
#     {
#       'Version' => '2012-10-17',
#       'Statement' => [
#         'Sid' => 'Stmt1',
#         'Effect' => 'Allow',
#         'Action' => 's3:ListBucket',
#         'Resource' => 'arn:aws:s3:::amzn-s3-demo-bucket'
#       ]
#     }
#   )
#   exit 1 unless credentials.access_key_id
#   puts "Access key ID: #{credentials.access_key_id}"
def get_temporary_credentials(sts, duration_seconds, user_name, policy)
  response = sts.get_federation_token(
    duration_seconds: duration_seconds,
    name: user_name,
    policy: policy.to_json
  )

```

```

    )
    response.credentials
  rescue StandardError => e
    puts "Error while getting federation token: #{e.message}"
  end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'amzn-s3-demo-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts 'Name => ETag'
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end

# Example usage:
def run_me
  region = "us-west-2"
  user_name = "my-user"
  bucket_name = "amzn-s3-demo-bucket"

  iam = Aws::IAM::Client.new(region: region)
  user = get_user(iam, user_name)

```

```
exit 1 unless user.user_name

puts "User's name: #{user.user_name}"
sts = Aws::STS::Client.new(region: region)
credentials = get_temporary_credentials(sts, 3600, user_name,
                                         {
                                           'Version' => '2012-10-17',
                                           'Statement' => [
                                             'Sid' => 'Stmt1',
                                             'Effect' => 'Allow',
                                             'Action' => 's3:ListBucket',
                                             'Resource' =>
"arn:aws:s3:::#{bucket_name}"
                                           ]
                                         })

exit 1 unless credentials.access_key_id

puts "Access key ID: #{credentials.access_key_id}"
s3_client = Aws::S3::Client.new(region: region, credentials: credentials)

exit 1 unless list_objects_in_bucket?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

## REST API を使用したリクエストの実行

このセクションでは、REST API を使用して Amazon S3 のエンドポイントにリクエストを生成する方法について説明します。Amazon S3 エンドポイントのリストは、AWS 全般のリファレンスの「[リージョンとエンドポイント](#)」を参照してください。

### REST API リクエスト用の S3 ホスト名の生成

Amazon S3 エンドポイントは、以下の構造に従います。

```
s3.Region.amazonaws.com
```

Amazon S3 アクセスポイントエンドポイントおよびデュアルスタックエンドポイントも標準構造に従います。

- Amazon S3 アクセスポイント -s3-accesspoint.*Region*.amazonaws.com
- デュアルスタック - s3.dualstack.*Region*.amazonaws.com

Amazon S3 リージョンとエンドポイントの完全なリストについては、Amazon Web Services 全般のリファレンスの「[Amazon S3 エンドポイントとクォータ](#)」を参照してください。

## 仮想ホスティング形式およびパス形式のリクエスト

REST API を使用してリクエストを生成すると、Amazon S3 のエンドポイントに仮想ホスティング形式またはパス形式の URI を使用できます。詳細については、「[パス形式のリクエスト](#)」を参照してください。

### Example 仮想ホスティング形式のリクエスト

米国西部 (オレゴン) リージョンにある puppy.jpg ファイルを examplebucket という名前のバケットから削除する仮想ホスティング形式のリクエストの例を次に示します。仮想ホスト形式のリクエストの詳細については、「[パス形式のリクエスト](#)」を参照してください。

```
DELETE /puppy.jpg HTTP/1.1
Host: examplebucket.s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

### Example パス形式のリクエスト

同じリクエストのパス形式バージョンの例を次に示します。

```
DELETE /examplebucket/puppy.jpg HTTP/1.1
Host: s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

現在、Amazon S3 では、すべての AWS リージョン で仮想ホスト形式の URL とパス形式の URL の両方をサポートしています。ただし、パス形式の URL は将来廃止される予定です。詳細については、次の重要な注記を参照してください。

パス形式のリクエストについての詳細は、「[パス形式のリクエスト](#)」を参照してください。

**⚠ Important**

更新 (2020 年 9 月 23 日) – お客様が仮想ホスティング形式の URL への移行に必要な時間を確保できるように、パス形式 URL の非推奨化を延期することが決定しました。詳細については、AWS ニュースブログの [Amazon S3 Path Deprecation Plan – The Rest of the Story](#) を参照してください。

## REST API を使用したデュアルスタックのエンドポイントへのリクエストの実行

REST API を使用すると、仮想ホスティング形式やパス形式のエンドポイント名 (URI) を使用してデュアルスタックのエンドポイントに直接アクセスできます。すべての Amazon S3 デュアルスタックのエンドポイント名は名前にリージョンが含まれます。標準の IPv4 のみのエンドポイントとは異なり、仮想ホスティング形式とパス形式のエンドポイントではリージョン固有のエンドポイント名を使用します。

### Example 仮想ホスティング形式のデュアルスタックのエンドポイントのリクエスト

米国西部 (オレゴン) リージョンにある puppy.jpg という名前のバケットから examplebucket オブジェクトを取得する次の例に示すように、REST リクエストで仮想ホスティング形式のエンドポイントを使用できます。

```
GET /puppy.jpg HTTP/1.1
Host: examplebucket.s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

### Example パス形式のデュアルスタックのエンドポイントのリクエスト

または次の例に示すように、リクエストでパス形式のエンドポイントを使用することができます。

```
GET /examplebucket/puppy.jpg HTTP/1.1
Host: s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

デュアルスタックのエンドポイントの詳細については、「[Amazon S3 デュアルスタックのエンドポイントの使用](#)」を参照してください。

REST API を使用したリクエストの詳細については、次のトピックを参照してください。

## トピック

- [リクエストのリダイレクトと REST API](#)
- [リクエストルーティング](#)

## リクエストのリダイレクトと REST API

### トピック

- [リダイレクトおよび HTTP ユーザーエージェント](#)
- [リダイレクトおよび 100-continue](#)
- [リダイレクトの例](#)

このセクションでは、Amazon S3 REST API を使用して HTTP リダイレクトを処理する方法について説明します。Amazon S3 リダイレクトの一般的な情報については、Amazon Simple Storage Service API リファレンスの「[リクエストの実行](#)」を参照してください。

### リダイレクトおよび HTTP ユーザーエージェント

Amazon S3 REST API を使用するプログラムは、アプリケーションレイヤーレベルまたは HTTP レイヤーレベルのいずれかでリダイレクトを処理する必要があります。HTTP クライアントライブラリとユーザーエージェントの多くは、リダイレクトを自動的に正しく処理するように設定することができます。ただし、リダイレクト実装が不正確あるいは不完全であるものも多くあります。

ライブラリでリダイレクト要件を満たすようにする前に、次のテストを行ってください。

- すべての HTTP リクエストヘッダーが (Authorization、Date などの HTTP 標準を含む)、リダイレクトされたリクエスト (リダイレクトの受信後の 2 番目のリクエスト) に適切に含まれていることを確認します。
- PUT、DELETE など、非 GET リダイレクトが正しく動作することを確認します。
- 大きな PUT リクエストがリダイレクトに正しく従っていることを確認します。
- 100-continue レスポンスが届くまでに時間がかかる場合は、PUT リクエストがリダイレクトに正しく従っていることを確認します。

HTTP リクエストが GET または HEAD でない場合、RFC 2616 に厳密に準拠する HTTP ユーザーエージェントは、リダイレクトにしたがう前に明示的な確認が必要になることがあります。

一般的に、Amazon S3 によって自動生成されたリダイレクトに従うと安全です。システムが amazonaws.com ドメイン内のホストのみに対してリダイレクトを発行し、リダイレクトされたリクエストの効果は元のリクエストと同じになるためです。

## リダイレクトおよび 100-continue

リダイレクトの処理を簡素化し、効率性を高め、リダイレクトされたリクエストボディを 2 回送信するためのコストを排除するには、PUT オペレーションに対して 100-continue を使用するようにアプリケーションを設定します。100-continue を使用しているアプリケーションは、確認を受け取るまでリクエストボディを送信しません。ヘッダーに基づいてメッセージが拒否された場合、メッセージの本文は送信されません。100-continue の詳細については、「[RFC 2616 Section 8.2.3](#)」を参照してください。

### Note

RFC 2616 にしたがって、不明な HTTP サーバーで Expect: Continue を使用する場合は、リクエストボディを送信する前の待機時間は無期限にしないでください。HTTP サーバーの中には、100-continue を認識しないものがあるからです。しかし、Amazon S3 は、リクエストに Expect: Continue が含まれているかどうか、また一時的な 100-continue ステータスを返すか、最終ステータスコードを返すかを認識します。さらに、一時的な 100-continue の承認を受け取った後、リダイレクトエラーが発生することはありません。これは、リクエスト本文を書き込んでいる間は、リダイレクトレスポンスを受け取らないようにする際に役立ちます。

## リダイレクトの例

このセクションでは、HTTP リダイレクトと 100-continue を使用したクライアントとサーバーのやり取りの例を紹介します。

quotes.s3.amazonaws.com バケットに対する PUT の例を次に示します。

```
PUT /nelson.txt HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 は以下を返します。

```
HTTP/1.1 307 Temporary Redirect
Location: http://quotes.s3-4c25d83b.amazonaws.com/nelson.txt?rk=8d47490b
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Mon, 15 Oct 2007 22:18:46 GMT

Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the
    specified temporary endpoint. Continue to use the
    original request endpoint for future requests.
  </Message>
  <Endpoint>quotes.s3-4c25d83b.amazonaws.com</Endpoint>
  <Bucket>quotes</Bucket>
</Error>
```

クライアントは、リダイレクトレスポンスにしたがって、新しいリクエストを quotes.s3-4c25d83b.amazonaws.com 一時エンドポイントに発行します。

```
PUT /nelson.txt?rk=8d47490b HTTP/1.1
Host: quotes.s3-4c25d83b.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 は 100-continue 返します。これは、クライアントがリクエストボディを送信し続ける必要があることを示します。

```
HTTP/1.1 100 Continue
```

クライアントはリクエストボディを送信します。

```
ha ha\n
```

Amazon S3 は最終レスポンスを返します。



```
HTTP/1.1 200 OK
Date: Mon, 15 Oct 2007 22:18:48 GMT

ETag: "a2c8d6b872054293afd41061e93bc289"
Content-Length: 0
Server: AmazonS3
```

## リクエストルーティング

[CreateBucketConfiguration](#) を含む [CreateBucket](#) API を使用して作成されたバケットに対してリクエストを実行するプログラムは、リダイレクトをサポートしている必要があります。また、DNS TTL に従わない一部のクライアントで問題が発生する場合があります。

このセクションでは、Amazon S3 で使用するサービスまたはアプリケーションを設計するときに考慮すべきルーティングおよび DNS の問題について説明します。

### リクエストのリダイレクトと REST API

Amazon S3 はドメインネームシステム (DNS) を使用して、リクエストを処理できる施設にルーティングします。このシステムは有効に機能しますが、一時的なルーティングエラーが発生する場合があります。リクエストが誤った Amazon S3 ロケーションに到達した場合、Amazon S3 は、リクエストを新しいエンドポイントに再送するようリクエストに指示する一時的なリダイレクトで応答します。リクエストの形式が正しくない場合、Amazon S3 は恒久的なリダイレクトを使用して、リクエストを正しく実行する方法について指示します。

#### Important

この機能を使用するには、Amazon S3 リダイレクトレスポンスを処理できるアプリケーションが必要です。唯一の例外は、<CreateBucketConfiguration> なしで作成されたバケットのみを使用するアプリケーションです。場所の制約の詳細については、「[Amazon S3 バケットに対するアクセスと一覧表示](#)」を参照してください。

2019 年 3 月 20 日より後に開始されたすべてのリージョンで、リクエストが間違った Amazon S3 ロケーションに到着した場合、Amazon S3 は HTTP 400 Bad Request エラーを返します。

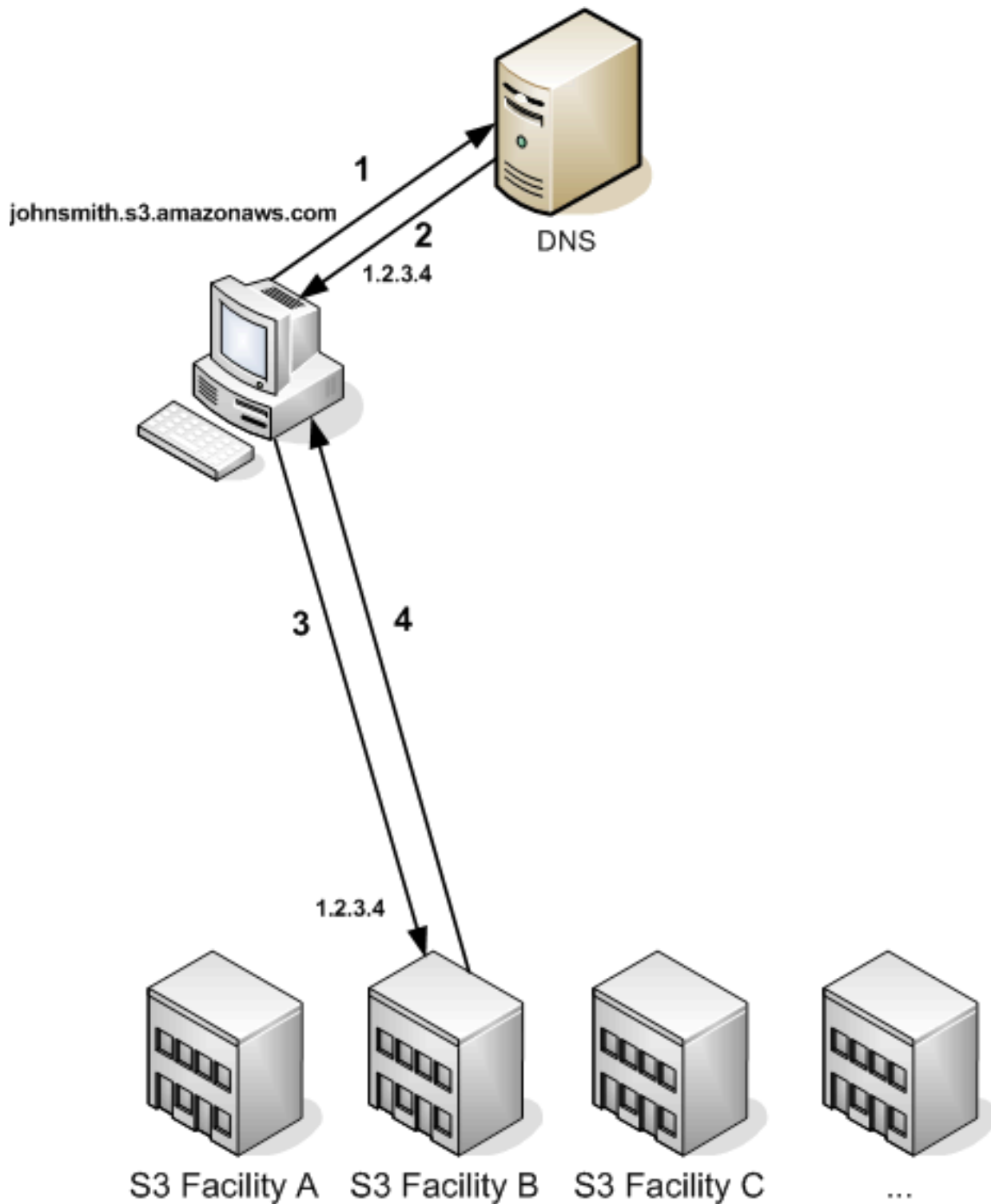
AWS リージョンの有効化または無効化の詳細については、「AWS 全般のリファレンス」の「[AWS リージョンとエンドポイント](#)」を参照してください。

## トピック

- [DNS ルーティング](#)
- [一時的なリクエストのリダイレクト](#)
- [永続的なリクエストのリダイレクト](#)
- [リクエストのリダイレクトの例](#)

## DNS ルーティング

DNS ルーティングは、リクエストを適切な Amazon S3 施設にルーティングします。以下の図とステップは、DNS ルーティングの例を示しています。



### DNS ルーティングリクエストのステップ

1. クライアントが、Amazon S3 に保存されているオブジェクトを取得するために DNS リクエストを行います。

2. クライアントが、リクエストを処理できる施設の 1 つまたは複数の IP アドレスを受け取ります。この例で、IP アドレスは施設 B のものです。
3. クライアントが、Amazon S3 施設 B へのリクエストを行います。
4. 施設 B が、オブジェクトのコピーをクライアントに返します。

### 一時的なリクエストのリダイレクト

一時的なリダイレクトとは、リクエストを異なるエンドポイントに再送する必要があることをリクエストに知らせる一種のエラーレスポンスです。Amazon S3 が持つ分散型の特質により、リクエストが誤った施設に一時的にルーティングされる場合があります。このエラーは、大半の場合にバケットが作成または削除された直後に発生します。

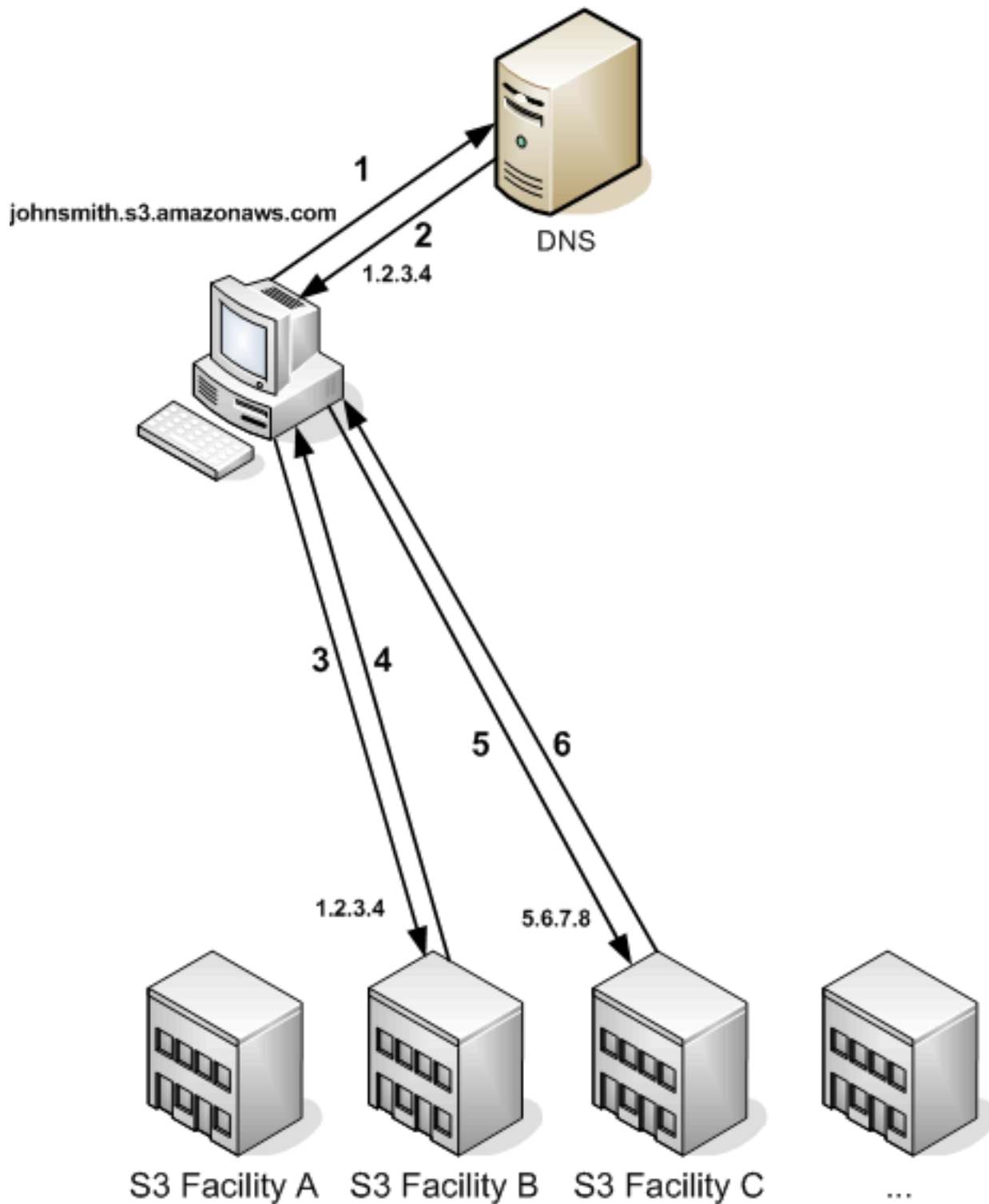
たとえば、新しいバケットを作成し、ただちにバケットへのリクエストを実行する場合、バケットのロケーションの制約によっては一時的にリダイレクトされる可能性もあります。AWS リージョンの米国東部 (バージニア北部) にバケットを作成した場合、これはデフォルトの Amazon S3 エンドポイントでもあるため、そのバケットへのリクエストはリダイレクトされることはありません。

一方、その他のリージョンにバケットを作成した場合、バケットの DNS エントリは伝播されますが、バケットに対するリクエストはデフォルトのエンドポイントに転送されます。デフォルトのエンドポイントは、リクエストを正しいエンドポイントにリダイレクトし、HTTP 302 レスポンスを返します。一時的なリダイレクトには正しい施設への URI が含まれており、これを使用して直ちにリクエストを再送できます。

#### Important

以前のリダイレクトレスポンスで返されたエンドポイントを再利用しないでください。このようなエンドポイントは (長期的にも) 機能するようには見えますが、予期しない結果を招くことがあります。最終的には通知なしに失敗します。

以下の図とステップは、一時的なリダイレクトの例を示しています。



### 一時的なリクエストのリダイレクトのステップ

1. クライアントが、Amazon S3 に保存されているオブジェクトを取得するために DNS リクエストを行います。

2. クライアントが、リクエストを処理できる施設の 1 つまたは複数の IP アドレスを受け取ります。
3. クライアントが、Amazon S3 施設 B へのリクエストを行います。
4. 施設 B がリダイレクトを返して、オブジェクトがロケーション C から利用可能であることを示します。
5. クライアントがリクエストを施設 C に再送します。
6. 施設 C がオブジェクトのコピーを返します。

## 永続的なリクエストのリダイレクト

恒久的なリダイレクトは、リクエストがリソースのアドレス指定を不適切に行ったことを示します。たとえば、パス形式のリクエストを使用して、<CreateBucketConfiguration> によって作成されたバケットにアクセスした場合、恒久的なリダイレクトが発生します。詳細については、「[Amazon S3 バケットに対するアクセスと一覧表示](#)」を参照してください。

開発中にこれらのエラーを見つけやすいように、このタイプのリダイレクトには、リクエストを正しいロケーションに自動的に送信するためのロケーション HTTP ヘッダーが含まれていません。正しい Amazon S3 エンドポイントを使用するために、生成される XML エラードキュメントを参照してください。

## リクエストのリダイレクトの例

一時的なリクエストのリダイレクトレスポンスの例を以下に示します。

## REST API の一時的なリダイレクトレスポンス

```
HTTP/1.1 307 Temporary Redirect
Location: http://awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com/photos/puppy.jpg?rk=e2c69a31
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Fri, 12 Oct 2007 01:12:56 GMT
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the specified temporary endpoint.
```

```
Continue to use the original request endpoint for future requests.</Message>
<Endpoint>awsexamplebucket1.s3-gztlb4pa9sq.amazonaws.com</Endpoint>
</Error>
```

## SOAP API の一時的なリダイレクトレスポンス

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.TemporaryRedirect</Faultcode>
    <Faultstring>Please re-send this request to the specified temporary endpoint.
    Continue to use the original request endpoint for future requests.</Faultstring>
    <Detail>
      <Bucket>images</Bucket>
      <Endpoint>s3-gztlb4pa9sq.amazonaws.com</Endpoint>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

## DNS に関する考慮事項

Amazon S3 の設計要件の 1 つは、非常に高い可用性です。この要件を満たす方法の 1 つとして、DNS 内の Amazon S3 エンドポイントに関連付けられた IP アドレスを必要に応じて更新します。これらの変更は、存続時間の短いクライアントには自動的に反映されますが、存続時間の長いクライアントには反映されない場合があります。存続時間の長いクライアントの場合、これらの変更を活用するには、Amazon S3 エンドポイントを定期的に再解決するための特殊なアクションを実行する必要があります。仮想マシン (VM) の詳細については、次のトピックを参照してください。

- Java の場合、Sun の JVM はデフォルトで DNS ルックアップを永続的にキャッシュします。この動作を変更する方法については、[InetAddress のドキュメント](#)の「InetAddress キャッシング」セクションを参照してください。
- PHP の場合、ほとんどの一般的な導入構成で実行される永続的な PHP VM は、VM が再起動されるまで DNS ルックアップをキャッシュに保持します。[getHostByName についての PHP ドキュメント](#)を参照してください。

# AWS CLI を使用した Amazon S3 での開発

Amazon S3 AWS CLI コマンドは、AWS CLI コマンドリファレンスのさまざまなセットに整理されており、それぞれに独自のコマンドがあります。探しているコマンドがあるセットで見つからない場合は、他のセットのいずれかを確認してください。それぞれのセットは次のとおりです。

- `s3` – オブジェクトとバケットを操作するための高レベルのコマンドについて説明します。これには、コピー、リスト、移動アクションが含まれます。このセットのコマンドの完全なリストについては、AWS CLI コマンドリファレンスの「[s3](#)」を参照してください。
- `s3api` – バケット、オブジェクト、セッション、マルチパートアップロードなどの S3 リソースを管理する低レベルのコマンドについて説明します。これらのコマンドは、[Amazon S3 API](#) オペレーションに対応しています。このセットの CLI コマンドの完全なリストについては、AWS CLI コマンドリファレンスの「[s3api](#)」を参照してください。
- `s3control` – Access Grants、ストレージレンズグループ、Amazon S3 on Outposts バケットなど、他のすべての S3 リソースを管理する低レベルコマンドについて説明します。これらのコマンドは、[Amazon S3 Control](#) API オペレーションに対応しています。このセットの CLI コマンドの完全なリストについては、AWS CLI コマンドリファレンスの「[s3control](#)」を参照してください。

AWS CLI の設定に問題がある場合は、AWS re:Post ナレッジセンターの「[Amazon S3 で「認証情報が見つかりません」エラーを解決する方法](#)」を参照してください。

## Note

Amazon S3 などの AWS のサービスは、サービスにアクセスする際に認証情報を提供する必要があります。これにより、サービスのリソースへアクセスするための権限があるかどうか判断できるようになります。コンソールを使用するにはパスワードが必要です。AWS アカウントによる AWS CLI または API へのアクセス用にアクセスキーを作成できます。ただし、AWS アカウントの認証情報を使用して AWS にアクセスすることは推奨されません。代わりに AWS Identity and Access Management (IAM) を使用することをお勧めします。IAM ユーザーを作成し、管理者アクセス許可を持つ IAM グループにユーザーを追加したら、作成した IAM ユーザーに管理者アクセス許可を付与します。これで、特別な URL と IAM ユーザーの認証情報を使って AWS にアクセスできます。手順については、「IAM ユーザーガイド」の「[最初の IAM ユーザーと管理者グループの作成](#)」を参照してください。



## AWS CLI の詳細

AWS の詳細については、以下のリソースを参照してください。

- [AWS Command Line Interface](#)
- [AWS Command Line Interface バージョン 2 用ユーザーガイド](#)

## AWS SDK を使用した Amazon S3 での開発

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

### Note

AWS Amplify を使用して、ウェブおよびモバイルアプリのエンドツーエンドのフルスタック開発を行うことができます。Amplify Storage は、Amazon S3 上に構築されたフロントエンドのウェブおよびモバイルアプリにファイルストレージと管理機能をシームレスに統合します。詳細については、「Amplify ユーザーガイド」の「[Storage](#)」を参照してください。

| SDK ドキュメント                             | コードの例                                        |
|----------------------------------------|----------------------------------------------|
| <a href="#">AWS SDK for C++</a>        | <a href="#">AWS SDK for C++ コードの例</a>        |
| <a href="#">AWS CLI</a>                | <a href="#">AWS CLI コードの例</a>                |
| <a href="#">AWS SDK for Go</a>         | <a href="#">AWS SDK for Go コードの例</a>         |
| <a href="#">AWS SDK for Java</a>       | <a href="#">AWS SDK for Java コードの例</a>       |
| <a href="#">AWS SDK for JavaScript</a> | <a href="#">AWS SDK for JavaScript コードの例</a> |
| <a href="#">AWS SDK for Kotlin</a>     | <a href="#">AWS SDK for Kotlin コードの例</a>     |
| <a href="#">AWS SDK for .NET</a>       | <a href="#">AWS SDK for .NET コードの例</a>       |
| <a href="#">AWS SDK for PHP</a>        | <a href="#">AWS SDK for PHP コードの例</a>        |

| SDK ドキュメント                                 | コードの例                                            |
|--------------------------------------------|--------------------------------------------------|
| <a href="#">AWS Tools for PowerShell</a>   | <a href="#">Tools for PowerShell のコード例</a>       |
| <a href="#">AWS SDK for Python (Boto3)</a> | <a href="#">AWS SDK for Python (Boto3) コードの例</a> |
| <a href="#">AWS SDK for Ruby</a>           | <a href="#">AWS SDK for Ruby コードの例</a>           |
| <a href="#">AWS SDK for Rust</a>           | <a href="#">AWS SDK for Rust コードの例</a>           |
| <a href="#">AWS SDK for SAP ABAP</a>       | <a href="#">AWS SDK for SAP ABAP コードの例</a>       |
| <a href="#">AWS SDK for Swift</a>          | <a href="#">AWS SDK for Swift コードの例</a>          |

具体的な例については、「[AWS SDK を使用した Amazon S3 のコード例](#)」を参照してください。

## SDK プログラミングインターフェイス

各 AWS SDK には、Amazon S3 を操作するプログラミングインターフェイスが 1 つ以上用意されています。各 SDK は、Amazon S3 用の低レベルインターフェイスを提供します。これには、API オペレーションに似たメソッドが含まれます。一部の SDK は、一般的なユースケースを簡素化するための抽象化である Amazon S3 用の高レベルインターフェイスを提供します。

例えば、低レベル API オペレーションを使用してマルチパートアップロードを実行する場合は、オペレーションを使用してアップロードを開始し、別のオペレーションを使用してパートをアップロードし、最後のオペレーションを使用してアップロードを完了します。高レベルのマルチパートアップロード API オペレーションを使用すると、1 回の API コールでアップロードに必要なすべてのオペレーションを実行できます。例については、「Amazon S3 ユーザーガイド」の「[マルチパートアップロードを使用したオブジェクトのアップロード](#)」を参照してください。

低レベルの API オペレーションを使用すると、アップロードをより詳細に制御できます。アップロードを一時停止して再開する必要がある場合、アップロード中にパートサイズを変更する場合、またはアップロードデータのサイズが事前にわからない場合は、低レベル API オペレーションの使用をお勧めします。

## リクエスト認証での署名バージョンの指定

Amazon S3 は、ほとんどの AWS リージョン で AWS Signature Version 4 のみをサポートしています。ただし、初期の AWS リージョン のいくつかでは、Amazon S3 は Signature Version 4 と

Signature Version 2 をサポートしています。ただし、署名バージョン 2 はオフになっています (廃止)。Signature Version 2 のサポート終了についての詳細は、「」を参照してください [Amazon S3 用の AWS 署名バージョン 2 がオフになっている \(廃止\)](#)

すべての Amazon S3 リージョンとリージョンでサポートされている署名バージョンのリストについては AWS 全般のリファレンスの [リージョンとエンドポイント](#) を参照してください。

すべての AWS リージョン では、AWS SDK はデフォルトで Signature Version 4 を使用してリクエストを認証します。2016 年 5 月以前にリリースされた AWS SDK を使用する場合、次の表に示すように、Signature Version 4 のリクエストが必要になることがあります。

| SDK            | リクエスト認証に署名バージョン 4 を使用するようにリクエストする                                                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AWS CLI        | <p>デフォルトプロファイルの場合、次のコマンドを実行します。</p> <pre>\$ aws configure set default.s3.signature_version s3v4</pre> <p>カスタムプロファイルの場合、次のコマンドを実行します。</p> <pre>\$ aws configure set profile.your_profile_name.s3.signature_version s3v4</pre> |
| Java SDK       | <p>コードに以下を追加します:</p> <pre>System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");</pre> <p>または、コマンドラインで次のように指定します。</p> <pre>-Dcom.amazonaws.services.s3.enableV4</pre>                           |
| JavaScript SDK | <p>クライアントを構築するときに、signatureVersion パラメータを v4 に設定します。</p> <pre>var s3 = new AWS.S3({signatureVersion: 'v4'});</pre>                                                                                                           |

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SDK             | リクエスト認証に署名バージョン 4 を使用するようにリクエストする                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| PHP SDK         | <p>PHP SDK v2 用の Amazon S3 サービスクライアントを構築する場合は、<code>signature</code> パラメータを <code>v4</code> に設定します。</p> <pre>&lt;?php \$client = S3Client::factory([     'region' =&gt; 'YOUR-REGION',     'version' =&gt; 'latest',     'signature' =&gt; 'v4' ]);</pre> <p>PHP SDK v3 を使用している場合、Amazon S3 サービスクライアントの構築中に <code>signature_version</code> パラメータを <code>v4</code> に設定します。</p> <pre>&lt;?php \$s3 = new Aws\S3\S3Client([     'version' =&gt; '2006-03-01',     'region' =&gt; 'YOUR-REGION',     'signature_version' =&gt; 'v4' ]);</pre> |
| Python-Boto SDK | <p>boto デフォルト設定ファイルに以下を指定します。</p> <pre>[s3] use-sigv4 = True</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

|          |                                                                                                                                                                                                                                                                                                                                                        |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SDK      | リクエスト認証に署名バージョン 4 を使用するようにリクエストする                                                                                                                                                                                                                                                                                                                      |
| Ruby SDK | <p>Ruby SDK - バージョン 1: クライアントを構築するときに <code>:s3_signature_version</code> パラメータを <code>:v4</code> に設定します。</p> <pre>s3 = AWS::S3::Client.new(:s3_signature_version =&gt; :v4)</pre> <p>Ruby SDK - バージョン 3: クライアントを構築するときに <code>signature_version</code> パラメータを <code>v4</code> に設定します。</p> <pre>s3 = Aws::S3::Client.new(signature_version: 'v4')</pre> |
| .NET SDK | <p>Amazon S3 クライアントを作成する前に、コードに以下を追加します。</p> <pre>AWSConfigsS3.UseSignatureVersion4 = true;</pre> <p>または、以下を設定ファイルに追加します:</p> <pre>&lt;appSettings&gt;   &lt;add key="AWS.S3.UseSignatureVersion4" value="true" /&gt; &lt;/appSettings&gt;</pre>                                                                                                       |

## Amazon S3 用の AWS 署名バージョン 2 がオフになっている (廃止)

署名バージョン 2 は Amazon S3 でオフになっています (廃止)。Amazon S3 は署名バージョン 4 を使用して署名された API リクエストのみを受け付けます。

このセクションでは、署名バージョン 2 のサポート終了に関するよくある質問への回答が掲載されています。

署名バージョン 2 または 4 とは何ですか。また、署名リクエストとはどういう意味ですか。

署名バージョン 2 または署名バージョン 4 の署名プロセスは、Amazon S3 API リクエストを認証するために使用されます。リクエストの署名により、Amazon S3 によってリクエストの送信者を識別し、悪意のある人物からリクエストを保護することができます。

AWS リクエストの署名の詳細については、「AWS 全般のリファレンス」の「[AWS API リクエストの署名](#)」を参照してください。

どの更新を実行していますか。

現在、署名バージョン 2 および署名バージョン 4 のプロセスを使用して署名された Amazon S3 API リクエストがサポートされています。今後、Amazon S3 は署名バージョン 4 を使用して署名したリクエストのみを受け入れるようになります。

AWS リクエストの署名の詳細については、「AWS 全般のリファレンス」の「[Signature Version 4 の変更](#)」を参照してください。

なぜこの更新を実行するのですか。

署名バージョン 4 は、シークレットアクセスキーに代わって、署名されたキーを使用することでセキュリティの向上を提供します。Signature Version 4 は現在すべての AWS リージョン でサポートされていますが、Signature Version 2 は 2014 年 1 月以前にリリースしたリージョンにおいてのみサポートされます。この更新によって、すべてのリージョンを通してより一貫したエクスペリエンスを提供することができます。

署名バージョン 4 を使用していることはどのように確認できますか。また、どの更新が必要ですか。

リクエストを署名するために使用する署名バージョンは、通常の場合、クライアント側のツールあるいは SDK によって設定されます。デフォルトでは、最新バージョンの AWS SDK は Signature Version 4 を使用します。サードパーティのソフトウェアについては、お使いのソフトウェアに該当するサポートチームにお問い合わせいただき、必要なバージョンを確認してください。Amazon S3 にダイレクト REST コールを送信する場合、アプリケーションを変更して署名バージョン 4 の署名プロセスを使用する必要があります。

Signature Version 4 に移行するときに使用する AWS SDK のバージョンについての詳細は、[署名バージョン 2 から署名バージョン 4 への移行](#) を参照してください。

Amazon S3 REST API で Signature Version 4 を使用方法については、Amazon Simple Storage Service API リファレンスの[リクエストの認証 \(AWSSignature Version 4\)](#) を参照してください。

更新しない場合には、どうなりますか。

以降に署名バージョン 2 で署名されたリクエストは、Amazon S3 における認証で失敗します。リクエスト送信者には、リクエストが署名バージョン 4 で署名されている必要があることのエラーが表示されます。

7 日以上にわたって署名することが必要である署名付き URL を使用していても、変更することが必要ですか。

7 日以上にわたって署名することが必要である署名付き URL を使用している場合、現在のところ対応する必要はありません。引き続き AWS Signature Version 2 を使用して署名し、署名付き URL を認証できます。今後、署名付き URL シナリオにおける署名バージョン 4 への移行方法に関する詳細をフォローアップして、提供する予定です。

## 詳細

- Signature Version 4 の使用方法の詳細については、[AWS API リクエストの署名](#)を参照してください。
- 署名バージョン 2 と 署名バージョン 4 の変更点のリストについては、「[署名バージョン 4 の変更](#)」を参照してください。
- AWS フォーラムの投稿である [Amazon S3 API リクエストの署名における AWS Signature Version 2 から AWS Signature Version 4 への移行](#)を参照してください。
- ご質問やご意見がありましたら、[サポート](#) までお問い合わせください。

## 署名バージョン 2 から署名バージョン 4 への移行

Amazon S3 API リクエスト認証に署名バージョン 2 を現在使用している場合は、署名バージョン 4 の使用に移行する必要があります。「[Amazon S3 用の AWS 署名バージョン 2 がオフになっている\(廃止\)](#)」で説明されているとおり、署名バージョン 2 のサポートは終了します。

Amazon S3 REST API で Signature Version 4 を使用方法については、Amazon Simple Storage Service API リファレンスの[リクエストの認証 \(AWSSignature Version 4\)](#)を参照してください。

次の表では、署名バージョン 4 (SigV4) を使用するために必要な最低限のバージョンの SDK の一覧を示しています。AWS Java、JavaScript (Node.js)、あるいは Python (Boto/CLI) SDK を使った署名付き URL を使用している場合は、正しい AWS リージョン を設定し、クライアント設定で Signature Version 4 を設定する必要があります。クライアント設定で SigV4 を設定する詳細については、「[リクエスト認証での署名バージョンの指定](#)」を参照してください。

| この SDK または製品を使用する場合       | この SDK バージョンにアップグレード             | Sigv4 を使用するために、クライアントでコード変更が必要ですか。 | SDK ドキュメントへのリンク                        |
|---------------------------|----------------------------------|------------------------------------|----------------------------------------|
| AWS SDK for Java v1       | Java 1.11.201 以降または v2 にアップグレード。 | はい                                 | <a href="#">リクエスト認証での署名バージョンの指定</a>    |
| AWS SDK for Java v2       | SDK のアップグレードは不要です。               | いいえ                                | <a href="#">AWS SDK for Java</a>       |
| AWS SDK for .NET v1       | 3.1.10 以降にアップグレード。               | はい                                 | <a href="#">AWS SDK for .NET</a>       |
| AWS SDK for .NET v2       | 3.1.10 以降にアップグレード。               | いいえ                                | <a href="#">AWS SDK for .NET v2</a>    |
| AWS SDK for .NET v3       | 3.3.0.0 以降にアップグレード。              | はい                                 | <a href="#">AWS SDK for .NET v3</a>    |
| AWS SDK for JavaScript v1 | 2.68.0 以降にアップグレード。               | はい                                 | <a href="#">AWS SDK for JavaScript</a> |
| AWS SDK for JavaScript v2 | 2.68.0 以降にアップグレード。               | はい                                 | <a href="#">AWS SDK for JavaScript</a> |



| この SDK または製品を使用する場合       | この SDK バージョンにアップグレード                                                             | Sigv4 を使用するために、クライアントでコード変更が必要ですか。 | SDK ドキュメントへのリンク                        |
|---------------------------|----------------------------------------------------------------------------------|------------------------------------|----------------------------------------|
| AWS SDK for JavaScript v3 | 現在のところ、何もする必要はありません。主要バージョン V3 in Q3 2019 にアップグレード。                              | いいえ                                | <a href="#">AWS SDK for JavaScript</a> |
| AWS SDK for PHP v1        | 最新バージョンの PHP、または少なくとも S3 クライアントの設定で署名パラメータを v4 に設定した v2.7.4 にアップグレードすることをお勧めします。 | はい                                 | <a href="#">AWS SDK for PHP</a>        |

| この SDK または製品を使用する場合 | この SDK バージョンにアップグレード                                                             | Sigv4 を使用するために、クライアントでコード変更が必要ですか。 | SDK ドキュメントへのリンク                             |
|---------------------|----------------------------------------------------------------------------------|------------------------------------|---------------------------------------------|
| AWS SDK for PHP v2  | 最新バージョンの PHP、または少なくとも S3 クライアントの設定で署名パラメータを v4 に設定した v2.7.4 にアップグレードすることをお勧めします。 | いいえ                                | <a href="#">AWS SDK for PHP</a>             |
| AWS SDK for PHP v3  | SDK のアップグレードは不要です。                                                               | いいえ                                | <a href="#">AWS SDK for PHP</a>             |
| Boto2               | Boto2 v2.49.0 にアップグレード。                                                          | はい                                 | <a href="#">Boto 2 アップグレード</a>              |
| Boto3               | 1.5.71 (Botocore)、1.4.6 (Boto3) にアップグレード。                                        | はい                                 | <a href="#">Boto 3 - AWS SDK for Python</a> |

| この SDK または製品を使用する場合 | この SDK バージョンにアップグレード | Sigv4 を使用するために、クライアントでコード変更が必要ですか。 | SDK ドキュメントへのリンク                                    |
|---------------------|----------------------|------------------------------------|----------------------------------------------------|
| AWS CLI             | 1.11.108 にアップグレード。   | はい                                 | <a href="#">AWS Command Line Interface</a>         |
| AWS CLI v2 (プレビュー)  | SDK のアップグレードは不要です。   | いいえ                                | <a href="#">AWS Command Line Interface バージョン 2</a> |
| AWS SDK for Ruby v1 | Ruby V3 にアップグレード。    | はい                                 | <a href="#">Ruby V3 for AWS</a>                    |
| AWS SDK for Ruby v2 | Ruby V3 にアップグレード。    | はい                                 | <a href="#">Ruby V3 for AWS</a>                    |
| AWS SDK for Ruby v3 | SDK のアップグレードは不要です。   | いいえ                                | <a href="#">Ruby V3 for AWS</a>                    |
| Go                  | SDK のアップグレードは不要です。   | いいえ                                | <a href="#">AWS SDK for Go</a>                     |
| C++                 | SDK のアップグレードは不要です。   | いいえ                                | <a href="#">AWS SDK for C++</a>                    |

AWS Tools for Windows PowerShell または AWS Tools for PowerShell Core

3.3.0.0 より前のモジュールバージョンを使用している場合は、3.3.0.0 にアップグレードする必要があります。

バージョン情報を取得するには、Get-Module コマンドレットを使用します。

```
Get-Module -Name AWSPowerShell  
Get-Module -Name AWSPowerShell.NetCore
```

3.3.0.0 バージョンに更新するには、Update-Module コマンドレットを使用します。

```
Update-Module -Name AWSPowerShell  
Update-Module -Name AWSPowerShell.NetCore
```

署名バージョン 2 をトラフィックで送信する、7 日間以上で有効な署名付き URL を使用できます。

## AWS サポート の Amazon S3 リクエスト ID の取得

Amazon S3 でエラーまたは予期しない動作が発生したために、AWS サポート に問い合わせるときは、失敗したアクションに関連するリクエスト ID を提供する必要があります。AWS サポート はこれらのリクエスト ID を使用して、発生している問題を解決します。

リクエスト ID はペアになっていて、Amazon S3 が処理するすべての応答で返され (エラーの応答を含む)、詳細なログを通じてアクセスできます。リクエスト ID を取得するための一般的な方法がいくつかあります。例えば、S3 サーバーアクセスログや AWS CloudTrail イベントまたはデータイベントなどがあります。

これらのログを回復したら、2 つの値をコピーして記録しておきます。この値は、連絡するとき必要になります。AWS サポート AWS サポート へのお問い合わせの詳細については、[AWS サポート ドキュメント](#)の「[AWS へのお問い合わせ](#)」を参照してください。

### トピック

- [AWS CLI を使用したリクエスト ID の取得](#)
- [Windows PowerShell を使用してリクエスト ID を取得する](#)
- [AWS CloudTrail データイベントを使用してリクエスト ID を取得する](#)

- [S3 サーバーアクセスログ記録を使用してリクエスト ID を取得する](#)
- [HTTP を使用したリクエスト ID の取得](#)
- [ウェブブラウザを使用したオブジェクトリクエスト ID の取得](#)
- [AWS SDK を使用したリクエスト ID の取得](#)

## AWS CLI を使用したリクエスト ID の取得

AWS Command Line Interface (AWS CLI) を使用してリクエスト ID を取得するには、`--debug` をコマンドに追加してください。例えば、次のようにデバッグログに `x-amz-request-id` と `x-amz-id-2` が表示されます。

```
...
2025-04-30 14:35:28,572 - MainThread - botocore.parsers - DEBUG - Response headers:
{'x-amz-id-2': 'a+zm50vDJH3LLmCiXvwEo0u0PtPS/qCJaBvB2ZMH9dzyzTiJhiLZkBFFoRfsPf0KztUKT/garCI=', 'x-amz-request-id': 'N4NMN0MJ4VDFZMX9', 'Date': 'Wed, 30 Apr 2025 21:35:29 GMT', 'Content-Type': 'application/xml', 'Transfer-Encoding': 'chunked', 'Server': 'AmazonS3'}
...
```

## Windows PowerShell を使用してリクエスト ID を取得する

Windows PowerShell を使用してログを回復する方法については、「AWS デベロッパーツールブログ」の「[AWS Tools for Windows PowerShell における応答ログ](#)」を参照してください。

## AWS CloudTrail データイベントを使用してリクエスト ID を取得する

S3 オブジェクトレベルの API オペレーションをログに記録するように CloudTrail データイベントで設定された Amazon S3 バケットは、Amazon S3 でユーザー、ロール、または AWS サービスによって実行されるアクションに関する詳細情報を提供します。Athena で CloudTrail イベントをクエリすることで、S3 リクエスト ID を特定できます。

詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[CloudTrail を使用した Amazon S3 リクエストの特定](#)」を参照してください。

## S3 サーバーアクセスログ記録を使用してリクエスト ID を取得する

S3 サーバーアクセスログ用に設定された Amazon S3 バケットは、バケットに対して行われた各リクエストの詳細な記録を提供します。Athena を使用してサーバーアクセスログをクエリすることで、S3 リクエスト ID を特定できます。

詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[Amazon Athena を使用したリクエストのアクセスログのクエリ](#)」を参照してください。

## HTTP を使用したリクエスト ID の取得

ターゲットアプリケーションに到達する前に HTTP リクエストのビットを記録することで、リクエスト ID、x-amz-request-id、および x-amz-id-2 を取得することができます。HTTP リクエストの詳細なログを復元するために使用できる、さまざまなサードパーティー製ツールがあります。別の Amazon S3 HTTP リクエストを送信するにあたって、信頼しているツールを選択して実行し、Amazon S3 トラフィックが流れるポートをリッスンします。

HTTP リクエストの場合、リクエスト ID のペアは次のようになります。

```
x-amz-request-id: 79104EXAMPLEB723
x-amz-id-2: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km
```

### Note

HTTPS リクエストで暗号化され、ほとんどのパケットキャプチャで非表示になります。

## ウェブブラウザを使用したオブジェクトリクエスト ID の取得

ほとんどのウェブブラウザには、リクエストヘッダーを確認できる開発者ツールがあります。

ウェブブラウザに基づくリクエストでエラーが返される場合、リクエスト ID のペアは次の例のようになります。

```
<Error><Code>AccessDenied</Code><Message>Access Denied</Message>
<RequestId>79104EXAMPLEB723</RequestId><HostId>IOWQ4fDEXAMPLEQM
+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km</HostId></Error>
```

成功したリクエストからリクエスト ID のペアを取得するには、開発者ツールを使用して、HTTP レスポンスヘッダーを表示する必要があります。

## AWS SDK を使用したリクエスト ID の取得

以下のセクションでは、さまざまな AWS SDK を使用したログ記録の設定に関する情報を示します。

**Note**

すべてのリクエストと応答で詳細ログを有効にすることができますが、大きなリクエストまたは応答はアプリケーションで大幅な遅延を発生させる可能性があるため、本稼働システムでログ記録を有効にすることはお勧めしません。

AWS SDK リクエストの場合、リクエスト ID のペアは次のようになります。

```
Status Code: 403, AWS Service: Amazon S3, AWS Request ID: 79104EXAMPLEB723
AWS Error Code: AccessDenied AWS Error Message: Access Denied
S3 Extended Request ID: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK
+Jd1vEXAMPLEa3Km
```

**C++**

ロガーのタイプと詳細度は、SDKOptions 引数で SDK 初期化中に指定されます。次の例では、詳細レベルを LogLevel::Debug として指定します。

デフォルトのロガーはファイルシステムに書き込み、ファイルは次の規則 aws\_sdk\_YYYY-MM-DD-HH.log を使用して名前が付けられます。ロガーは 1 時間ごとに新しいファイルを作成します。

```
Aws::SDKOptions options;
options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
Aws::InitAPI(options);
// ...
Aws::ShutdownAPI(options);
```

詳細については、AWS GitHub の SDK for C++ Wiki の「[ログ記録を有効にするには](#)」を参照してください。

**Go**

ログ記録の設定は、SDK for Go を使用して行うことができます。詳細については、「AWS SDK for Go v2 デベロッパーガイド」の「[ログ記録](#)」を参照してください。

**Java**

特定のリクエストまたは応答のログ記録を有効にして、関連するヘッダーのみを取得して返すことができます。これを行うには、com.amazonaws.services.s3.S3ResponseMetadata

クラスをインポートします。その後、実際のリクエストを実行する前に変数にリクエストを保存できます。記録されたリクエストまたは応答を取得するには、`getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()` を呼び出します。

```
PutObjectRequest req = new PutObjectRequest(bucketName, key, createSampleFile());
s3.putObject(req);
S3ResponseMetadata md = s3.getCachedResponseMetadata(req);
System.out.println("Host ID: " + md.getHostId() + " RequestID: " +
    md.getRequestId());
```

または、Java のすべてのリクエストと応答の詳細ログ記録を使用できます。詳細については、AWS SDK for Java デベロッパーガイドで「[詳細なワイヤログ記録](#)」を参照してください。

## JavaScript

AWS SDK for JavaScript には組み込みロガーが実装されているため、これを使用して行った API コールをログに記録できます。ロガーをオンにしてコンソールにログエントリを出力するには、コードに次のステートメントを追加します。

```
AWS.config.logger = console;
```

詳細については、「AWS SDK for JavaScript デベロッパーガイド」の「[AWS SDK for JavaScript 呼び出しのログ記録](#)」を参照してください。

## Kotlin

AWS SDK for Kotlin を使用すると、コードまたは環境設定を使用して、ワイヤレベルのメッセージのログモードを指定できます。HTTP リクエストと HTTP レスポンスのログモードを設定できます。

追加のログ記録をオプトインするには、サービスクライアントを構築するときに `logMode` プロパティを設定します。

```
import aws.smithy.kotlin.runtime.client.LogMode

// ...

val client = S3Client {
    // ...
    logMode = LogMode.LogRequestWithBody + LogMode.LogResponse
```



```
}
```

または、環境変数を使用してログモードを設定することもできます。

```
export SDK_LOG_MODE=LogRequestWithBody|LogResponse
```

詳細については、「AWS SDK for Kotlin デベロッパーガイド」の「[ログ記録](#)」を参照してください。

## .NET

組み込みの System.Diagnostics ログ作成ツールを使用して、SDK for .NET のログ記録を設定できます。詳細については、AWS デベロッパーブログの投稿である「[SDK for .NET によるログ記録](#)」を参照してください。

### Note

デフォルトでは、返されたログにはエラー情報のみが含まれます。リクエスト ID を取得するには、設定ファイルに AWSLogMetrics (およびオプションで AWSResponseLogging) を追加する必要があります。

## PHP

クライアントのコンストラクタでデバッグオプションを true に設定することで、送信されるデータなどのデバッグ情報を取得できます。

```
$s3Client = new Aws\S3\S3Client([
    'region' => 'us-standard',
    'version' => '2006-03-01',
    'debug'   => true
]);
```

詳細については、AWS SDK for PHP デベロッパーガイドの「[送信されるデータを表示する方法](#)」を参照してください。

## Python (Boto3)

AWS SDK for Python (Boto3) を使用すると、特定の応答をログに記録できます。この機能を使用して、関連するヘッダーのみを取得できます。次のコードは、レスポンスの一部をファイルに記録する方法を示しています。

```
import logging
import boto3
logging.basicConfig(filename='logfile.txt', level=logging.INFO)
logger = logging.getLogger(__name__)
s3 = boto3.resource('s3')
response = s3.Bucket(bucket_name).Object(object_key).put()
logger.info("HTTPStatusCode: %s", response['ResponseMetadata']['HTTPStatusCode'])
logger.info("RequestId: %s", response['ResponseMetadata']['RequestId'])
logger.info("HostId: %s", response['ResponseMetadata']['HostId'])
logger.info("Date: %s", response['ResponseMetadata']['HTTPHeaders']['date'])
```

例外を発生時にキャッチし、関連情報をログに記録することもできます。詳細については、AWSSDK for Python (Boto) API リファレンスの「[エラー応答から有用な情報を識別する](#)」を参照してください。

さらに、次のコードを使用して、詳細なデバッグログを出力するように Boto3 を設定できます。

```
import logging
import boto3
boto3.set_stream_logger('', logging.DEBUG)
```

詳細については、AWS SDK for Python (Boto) API リファレンスの「[set\\_stream\\_logger](#)」を参照してください。

## Ruby

SDK for Ruby バージョン 3 を使用して、リクエスト ID を取得できます。次のコードを使用して、クライアントで HTTP ワイヤログ記録を有効にできます。

```
s3 = Aws::S3::Client.new(http_wire_trace: true)
```

### Note

ワイヤログ記録には、アクセスキー ID などの機密情報が含まれる場合があります。機密情報は、AWS サポートと共有する前にサニタイズする必要があります。

リクエスト context オブジェクトのリクエスト ID は、リクエストレスポンスまたはエラーでも確認できます。

```
# Finding the request ID from an error:
```

```
begin
  s3.put_object(bucket: 'bucket', key: 'key', body: 'test')
rescue Aws::S3::Errors::ServiceError => e
  puts e.context[:request_id]
  puts e.context[:s3_host_id]
end

# Finding the request ID from a successful call:
resp = s3.put_object(bucket: 'bucket', key: 'key', body: 'test')
puts resp.context[:request_id]
puts resp.context[:s3_host_id]
```

詳細については、「AWS SDK for Ruby デベロッパーガイド」の「[AWS SDK for Ruby クライアントからのワイヤトレース情報を使用したデバッグ](#)」を参照してください。

## Rust

ログ記録を有効にするには、Rust アプリケーションで tracing-subscriber クレートを追加して初期化します。

トレースライブラリを Cargo.toml ファイルに追加します。

```
tracing-subscriber = { version = "0.3", features = ["env-filter"] }
```

次に、Rust コードで、SDK オペレーションを呼び出す前に、メイン関数でロガーを初期化します。

```
tracing_subscriber::fmt::init();
```

詳細については、「AWS SDK for Rust デベロッパーガイド」の「[AWS SDK for Rust におけるログ記録の設定と使用](#)」を参照してください。

## S3 Tables でサポートされている Amazon S3 オブジェクトレベルの API オペレーション

次の表に、S3 Tables でサポートされている S3 オブジェクトレベルの API オペレーションと、対応するヘッダーを示します。Amazon S3 Tables API の一覧については、「[Amazon S3 Tables](#)」を参照してください。Amazon S3 Tables の詳細については、「Amazon S3 ユーザーガイド」の「[Amazon S3 Tables とテーブルバケットの使用](#)」を参照してください。

| オブジェクトレベルの API                          | サポートされているヘッダー               | 注意事項                                                         |  |
|-----------------------------------------|-----------------------------|--------------------------------------------------------------|--|
| <a href="#">AbortMultipartUpload</a>    | x-amz-expected-bucket-owner | なし                                                           |  |
| <a href="#">CompleteMultipartUpload</a> | x-amz-checksum-crc32        | なし                                                           |  |
|                                         | x-amz-checksum-crc32c       | なし                                                           |  |
|                                         | x-amz-checksum-sha1         | なし                                                           |  |
|                                         | x-amz-checksum-sha256       | なし                                                           |  |
|                                         | x-amz-expected-bucket-owner | なし                                                           |  |
|                                         | If-Match                    | なし                                                           |  |
|                                         | If-None-Match               | なし                                                           |  |
| <a href="#">CreateMultipartUpload</a>   | x-amz-acl: ACL              | S3 Tables の場合、デフォルト値は bucket-owner-full-control であり、変更できません。 |  |
|                                         | Cache-Control               | なし                                                           |  |
|                                         | Content-Disposition         | なし                                                           |  |
|                                         | Content-Encoding            | なし                                                           |  |

| オブジェクトレベルの API            | サポートされているヘッダー                | 注意事項                                                 |  |
|---------------------------|------------------------------|------------------------------------------------------|--|
|                           | Content-Language             | なし                                                   |  |
|                           | Content-Type                 | なし                                                   |  |
|                           | Expires                      | なし                                                   |  |
|                           | x-amz-checksum-algorithm     | なし                                                   |  |
|                           | x-amz-storage-class          | S3 Tables の場合、デフォルト値は STANDARD であり、変更できません。          |  |
|                           | x-amz-server-side-encryption | S3 Tables の場合、デフォルト値は (SSE-S3) (AES256) であり、変更できません。 |  |
| <a href="#">GetObject</a> | If-Match                     | なし                                                   |  |
|                           | If-Modified-Since            | なし                                                   |  |
|                           | If-None-Match                | なし                                                   |  |
|                           | If-Unmodified-Since          | なし                                                   |  |
|                           | Range                        | なし                                                   |  |
|                           | x-amz-expected-bucket-owner  | なし                                                   |  |
|                           | x-amz-checksum-mode          | なし                                                   |  |

| オブジェクトレベルの API             | サポートされているヘッダー               | 注意事項                                                         |  |
|----------------------------|-----------------------------|--------------------------------------------------------------|--|
| <a href="#">HeadObject</a> | If-Match                    | なし                                                           |  |
|                            | If-Modified-Since           | なし                                                           |  |
|                            | If-None-Match               | なし                                                           |  |
|                            | If-Unmodified-Since         | なし                                                           |  |
|                            | Range                       | なし                                                           |  |
|                            | x-amz-checksum-mode         | なし                                                           |  |
|                            | x-amz-expected-bucket-owner | なし                                                           |  |
| <a href="#">ListParts</a>  | x-amz-expected-bucket-owner | なし                                                           |  |
| <a href="#">PutObject</a>  | x-amz-acl: ACL              | S3 Tables の場合、デフォルト値は bucket-owner-full-control であり、変更できません。 |  |
|                            | Cache-Control               | なし                                                           |  |
|                            | Content-Disposition         | なし                                                           |  |
|                            | Content-Encoding            | なし                                                           |  |
|                            | Content-Language            | なし                                                           |  |
|                            | Content-Length              | なし                                                           |  |

| オブジェクトレベルの API | サポートされているヘッダー                | 注意事項                                                 |  |
|----------------|------------------------------|------------------------------------------------------|--|
|                | Content-MD5                  | なし                                                   |  |
|                | Content-Type                 | なし                                                   |  |
|                | x-amz-sdk-checksum-algorithm | なし                                                   |  |
|                | x-amz-checksum-crc32         | なし                                                   |  |
|                | x-amz-checksum-crc32c        | なし                                                   |  |
|                | x-amz-checksum-sha1          | なし                                                   |  |
|                | x-amz-checksum-sha256        | なし                                                   |  |
|                | Expires                      | なし                                                   |  |
|                | If-None-Match                | なし                                                   |  |
|                | x-amz-expected-bucket-owner  | なし                                                   |  |
|                | x-amz-storage-class          | S3 Tables の場合、デフォルト値は STANDARD であり、変更できません。          |  |
|                | x-amz-server-side-encryption | S3 Tables の場合、デフォルト値は (SSE-S3) (AES256) であり、変更できません。 |  |

| オブジェクトレベルの API             | サポートされているヘッダー               | 注意事項 |  |
|----------------------------|-----------------------------|------|--|
| <a href="#">UploadPart</a> | Content-Length              | なし   |  |
|                            | Content-MD5                 | なし   |  |
|                            | x-amz-checksum-crc32        | なし   |  |
|                            | x-amz-checksum-crc32c       | なし   |  |
|                            | x-amz-checksum-sha1         | なし   |  |
|                            | x-amz-checksum-sha256       | なし   |  |
|                            | x-amz-expected-bucket-owner | なし   |  |



# AWS SDK を使用した Amazon S3 のコード例

次のコード例は、AWS ソフトウェア開発キット (SDK) による Amazon S3 の使用方法を示しています。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## コードの例

- [AWS SDK を使用した Amazon S3 のコード例](#)
  - [AWS SDK を使用した Amazon S3 の基本的なコード例](#)
    - [Hello Amazon S3](#)
    - [AWS SDK を使用した Amazon S3 の基本的な使用方法を学ぶ](#)
    - [AWS SDK を使用した Amazon S3 のアクション](#)
      - [AWS SDK または CLI で AbortMultipartUpload を使用する](#)
      - [AWS SDK または CLI で CompleteMultipartUpload を使用する](#)
      - [AWS SDK または CLI で CopyObject を使用する](#)
      - [AWS SDK または CLI で CreateBucket を使用する](#)
      - [AWS SDK で CreateMultiRegionAccessPoint を使用する](#)
      - [AWS SDK または CLI で CreateMultipartUpload を使用する](#)
      - [AWS SDK で CreatePresignedPost を使用する](#)
      - [AWS SDK または CLI で DeleteBucket を使用する](#)
      - [CLI で DeleteBucketAnalyticsConfiguration を使用する](#)
      - [AWS SDK または CLI で DeleteBucketCors を使用する](#)
      - [CLI で DeleteBucketEncryption を使用する](#)
      - [CLI で DeleteBucketInventoryConfiguration を使用する](#)
      - [AWS SDK または CLI で DeleteBucketLifecycle を使用する](#)
      - [CLI で DeleteBucketMetricsConfiguration を使用する](#)
      - [AWS SDK または CLI で DeleteBucketPolicy を使用する](#)
      - [CLI で DeleteBucketReplication を使用する](#)
      - [CLI で DeleteBucketTagging を使用する](#)

- [AWS SDK または CLI で DeleteBucketWebsite を使用する](#)
- [AWS SDK または CLI で DeleteObject を使用する](#)
- [CLI で DeleteObjectTagging を使用する](#)
- [AWS SDK または CLI で DeleteObjects を使用する](#)
- [CLI で DeletePublicAccessBlock を使用する](#)
- [CLI で GetBucketAccelerateConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketAcl を使用する](#)
- [CLI で GetBucketAnalyticsConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketCors を使用する](#)
- [AWS SDK または CLI で GetBucketEncryption を使用する](#)
- [CLI で GetBucketInventoryConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketLifecycleConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketLocation を使用する](#)
- [CLI で GetBucketLogging を使用する](#)
- [CLI で GetBucketMetricsConfiguration を使用する](#)
- [CLI で GetBucketNotification を使用する](#)
- [AWS SDK または CLI で GetBucketPolicy を使用する](#)
- [CLI で GetBucketPolicyStatus を使用する](#)
- [AWS SDK または CLI で GetBucketReplication を使用する](#)
- [CLI で GetBucketRequestPayment を使用する](#)
- [CLI で GetBucketTagging を使用する](#)
- [CLI で GetBucketVersioning を使用する](#)
- [AWS SDK または CLI で GetBucketWebsite を使用する](#)
- [AWS SDK または CLI で GetObject を使用する](#)
- [AWS SDK または CLI で GetObjectAcl を使用する](#)
- [AWS SDK または CLI で GetObjectAttributes を使用する](#)
- [AWS SDK または CLI で GetObjectLegalHold を使用する](#)
- [AWS SDK または CLI で GetObjectLockConfiguration を使用する](#)
- [AWS SDK または CLI で GetObjectRetention を使用する](#)
- [CLI で GetObjectTagging を使用する](#)

- [CLI で GetPublicAccessBlock を使用する](#)
- [AWS SDK または CLI で HeadBucket を使用する](#)
- [AWS SDK または CLI で HeadObject を使用する](#)
- [CLI で ListBucketAnalyticsConfigurations を使用する](#)
- [CLI で ListBucketInventoryConfigurations を使用する](#)
- [AWS SDK または CLI で ListBuckets を使用する](#)
- [AWS SDK または CLI で ListMultipartUploads を使用する](#)
- [AWS SDK または CLI で ListObjectVersions を使用する](#)
- [CLI で ListObjects を使用する](#)
- [AWS SDK または CLI で ListObjectsV2 を使用する](#)
- [AWS SDK または CLI で PutBucketAccelerateConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketAcl を使用する](#)
- [AWS SDK または CLI で PutBucketCors を使用する](#)
- [AWS SDK または CLI で PutBucketEncryption を使用する](#)
- [AWS SDK または CLI で PutBucketLifecycleConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketLogging を使用する](#)
- [CLI で PutBucketNotification を使用する](#)
- [AWS SDK または CLI で PutBucketNotificationConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketPolicy を使用する](#)
- [AWS SDK または CLI で PutBucketReplication を使用する](#)
- [CLI で PutBucketRequestPayment を使用する](#)
- [CLI で PutBucketTagging を使用する](#)
- [AWS SDK または CLI で PutBucketVersioning を使用する](#)
- [AWS SDK または CLI で PutBucketWebsite を使用する](#)
- [AWS SDK または CLI で PutObject を使用する](#)
- [AWS SDK または CLI で PutObjectAcl を使用する](#)
- [AWS SDK または CLI で PutObjectLegalHold を使用する](#)
- [AWS SDK または CLI で PutObjectLockConfiguration を使用する](#)
- [AWS SDK または CLI で PutObjectRetention を使用する](#)
- [AWS SDK または CLI で RestoreObject を使用する](#)

- [AWS SDK または CLI で SelectObjectContent を使用する](#)
- [AWS SDK または CLI で UploadPart を使用する](#)
- [AWS SDK または CLI で UploadPartCopy を使用する](#)
- [AWS SDK を使用した Amazon S3 のシナリオ](#)
  - [バケットが存在するかどうかを確認する](#)
  - [オブジェクトが存在するかどうかを確認します。](#)
  - [AWS SDK を使用して、テキストを音声に変換し、テキストに戻す](#)
  - [AWS SDK を使用して Amazon S3 の署名付き URL を作成する](#)
  - [ユーザーがラベルを使用して写真を管理できる写真アセット管理アプリケーションの作成](#)
  - [AWS SDK を使用して Amazon S3 オブジェクトを一覧表示するウェブページ](#)
  - [Amazon Textract エクスプローラーアプリケーションを作成する](#)
  - [AWS SDK を使用して、指定した Amazon S3 バケットからすべてのオブジェクトを削除します。](#)
  - [AWS SDK を使用して Amazon S3 への不完全なマルチパートアップロードを削除する](#)
  - [AWS SDK を使用して、Amazon Rekognition で画像内の PPE を検出する](#)
  - [AWS SDK を使用して、画像から抽出されたテキスト内のエンティティを検出する](#)
  - [AWS SDK を使用して、画像内の顔を検出する](#)
  - [AWS SDK を使用して、Amazon Rekognition で画像内のオブジェクトを検出する](#)
  - [AWS SDK を使用して、Amazon Rekognition で、動画内の人やオブジェクトを検出する](#)
  - [Amazon Simple Storage Service \(Amazon S3\) バケットから S3 の「ディレクトリ」をダウンロードする](#)
  - [Amazon Simple Storage Service \(Amazon S3\) バケットからすべてのオブジェクトを、ローカルディレクトリにダウンロードする](#)
  - [サイズが不明なストリームを、AWS SDK を使用して Amazon S3 オブジェクトからダウンロードする](#)
  - [AWS SDK を使用してマルチリージョンアクセスポイントから Amazon S3 オブジェクトを取得する](#)
  - [AWS SDK を使用して、Amazon S3 バケットから If-Modified-Since を指定してオブジェクトを取得する](#)
  - [AWS SDK を使用して Amazon S3 オブジェクトの暗号化の使用を開始する](#)
  - [AWS SDK を使用して Amazon S3 オブジェクトのタグの使用を開始する](#)

- [ビッグデータ処理クラスターの使用開始](#)
- [設定管理の使用開始](#)
- [ドキュメントテキスト抽出の使用開始](#)
- [機械学習特徴量ストアの使用開始](#)
- [オブジェクトストレージの使用開始](#)
- [クエリ分析の使用開始](#)
- [AWS SDK を使用して Amazon S3 オブジェクトロック機能进行操作する](#)
- [AWS SDK を使用して Amazon S3 の条件付きリクエストを行う](#)
- [AWS SDK を使用して Amazon S3 バケットのアクセスコントロールリスト \(ACL\) を管理する](#)
- [AWS SDK で、Amazon S3 を使用して大量の Amazon SQS メッセージを管理する](#)
- [AWS SDK を使用して、バージョン管理されている Amazon S3 オブジェクトを Lambda 関数でバッチで管理する](#)
- [AWS SDK を使用して Amazon S3 の URI を解析する](#)
- [AWS SDK を使用して Amazon S3 オブジェクトのマルチパートコピーを実行する](#)
- [AWS SDK を使用して Amazon S3 イベント通知を受信して処理する](#)
- [AWS SDK を使用して、EXIF とその他の画像情報を保存する](#)
- [AWS SDK を使用して S3 イベント通知を Amazon EventBridge に送信する](#)
- [AWS SDK を使用して Amazon S3 オブジェクトのアップロードまたはダウンロードを追跡する](#)
- [S3 Object Lambda でアプリケーションのデータを変換する](#)
- [AWS SDK による単体テストと統合テストのアプローチ例](#)
- [ローカルディレクトリを Amazon Simple Storage Service \(Amazon S3\) バケットに、再帰的にアップロードする](#)
- [AWS SDK を使用して Amazon S3 との間で大きなファイルをアップロードまたはダウンロードする](#)
- [サイズが不明なストリームを AWS SDK を使用して Amazon S3 オブジェクトにアップロードする](#)
- [チェックサムを使用した AWS SDK での Amazon S3 オブジェクトの操作](#)
- [AWS SDK を使用して Amazon S3 オブジェクトの整合性機能进行操作する](#)
- [AWS SDK を使用して、Amazon S3 のバージョン管理されたオブジェクトを操作する](#)

- [Amazon S3 のサーバーレス例](#)
  - [Amazon S3 トリガーから Lambda 関数を呼び出す](#)
- [AWS SDK を使用した Amazon S3 Control のコード例](#)
- [AWS SDK を使用した Amazon S3 Control の基本的な例](#)
  - [Hello Amazon S3 Control](#)
  - [AWS SDK を使用して Amazon S3 Control の基本を学ぶ](#)
  - [AWS SDK を使用した Amazon S3 Control のアクション](#)
    - [AWS SDK または CLI で CreateJob を使用する](#)
    - [AWS SDK で DeleteJobTagging を使用する](#)
    - [AWS SDK または CLI で DescribeJob を使用する](#)
    - [AWS SDK で GetJobTagging を使用する](#)
    - [AWS SDK で PutJobTagging を使用する](#)
    - [AWS SDK または CLI で UpdateJobPriority を使用する](#)
    - [AWS SDK または CLI で UpdateJobStatus を使用する](#)
- [AWS SDK を使用した S3 ディレクトリバケットのコード例](#)
- [AWS SDK を使用した S3 ディレクトリバケットの基本的な例](#)
  - [Hello Amazon S3 ディレクトリバケット](#)
  - [AWS SDK を使用して S3 ディレクトリバケットの基本を学ぶ](#)
  - [AWS SDK を使用した S3 ディレクトリバケットのアクション](#)
    - [AWS SDK で AbortMultipartUpload を使用する](#)
    - [AWS SDK で CompleteMultipartUpload を使用する](#)
    - [AWS SDK で CopyObject を使用する](#)
    - [AWS SDK で CreateBucket を使用する](#)
    - [AWS SDK で CreateMultipartUpload を使用する](#)
    - [AWS SDK で CreateSession を使用する](#)
    - [AWS SDK で DeleteBucket を使用する](#)
    - [AWS SDK で DeleteBucketEncryption を使用する](#)
    - [AWS SDK で DeleteBucketPolicy を使用する](#)
    - [AWS SDK で DeleteObject を使用する](#)
    - [AWS SDK で DeleteObjects を使用する](#)

- [AWS SDK で GetBucketEncryption を使用する](#)
- [AWS SDK で GetBucketPolicy を使用する](#)
- [AWS SDK で GetObject を使用する](#)
- [AWS SDK で GetObjectAttributes を使用する](#)
- [AWS SDK で HeadBucket を使用する](#)
- [AWS SDK で HeadObject を使用する](#)
- [AWS SDK で ListDirectoryBuckets を使用する](#)
- [AWS SDK で ListMultipartUploads を使用する](#)
- [AWS SDK で ListObjectsV2 を使用する](#)
- [AWS SDK で ListParts を使用する](#)
- [AWS SDK で PutBucketEncryption を使用する](#)
- [AWS SDK で PutBucketPolicy を使用する](#)
- [AWS SDK で PutObject を使用する](#)
- [AWS SDK で UploadPart を使用する](#)
- [AWS SDK で UploadPartCopy を使用する](#)
- [AWS SDK を使用する S3 ディレクトリバケットのシナリオ](#)
  - [AWS SDK を使用して、オブジェクトを取得するための Amazon S3 ディレクトリバケットの署名付き URL を作成する](#)

## AWS SDK を使用した Amazon S3 のコード例

次のコード例は、AWS ソフトウェア開発キット (SDK) による Amazon S3 の使用方法を示しています。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。



AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## Amazon S3

- [AWS SDK を使用した Amazon S3 の基本的なコード例](#)
  - [Hello Amazon S3](#)
  - [AWS SDK を使用した Amazon S3 の基本的な使用方法を学ぶ](#)
  - [AWS SDK を使用した Amazon S3 のアクション](#)
    - [AWS SDK または CLI で AbortMultipartUpload を使用する](#)
    - [AWS SDK または CLI で CompleteMultipartUpload を使用する](#)
    - [AWS SDK または CLI で CopyObject を使用する](#)
    - [AWS SDK または CLI で CreateBucket を使用する](#)
    - [AWS SDK で CreateMultiRegionAccessPoint を使用する](#)
    - [AWS SDK または CLI で CreateMultipartUpload を使用する](#)
    - [AWS SDK で CreatePresignedPost を使用する](#)
    - [AWS SDK または CLI で DeleteBucket を使用する](#)
    - [CLI で DeleteBucketAnalyticsConfiguration を使用する](#)
    - [AWS SDK または CLI で DeleteBucketCors を使用する](#)
    - [CLI で DeleteBucketEncryption を使用する](#)
    - [CLI で DeleteBucketInventoryConfiguration を使用する](#)
    - [AWS SDK または CLI で DeleteBucketLifecycle を使用する](#)
    - [CLI で DeleteBucketMetricsConfiguration を使用する](#)
    - [AWS SDK または CLI で DeleteBucketPolicy を使用する](#)
    - [CLI で DeleteBucketReplication を使用する](#)
    - [CLI で DeleteBucketTagging を使用する](#)
    - [AWS SDK または CLI で DeleteBucketWebsite を使用する](#)
    - [AWS SDK または CLI で DeleteObject を使用する](#)
    - [CLI で DeleteObjectTagging を使用する](#)
    - [AWS SDK または CLI で DeleteObjects を使用する](#)
    - [CLI で DeletePublicAccessBlock を使用する](#)



- [CLI で GetBucketAccelerateConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketAcl を使用する](#)
- [CLI で GetBucketAnalyticsConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketCors を使用する](#)
- [AWS SDK または CLI で GetBucketEncryption を使用する](#)
- [CLI で GetBucketInventoryConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketLifecycleConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketLocation を使用する](#)
- [CLI で GetBucketLogging を使用する](#)
- [CLI で GetBucketMetricsConfiguration を使用する](#)
- [CLI で GetBucketNotification を使用する](#)
- [AWS SDK または CLI で GetBucketPolicy を使用する](#)
- [CLI で GetBucketPolicyStatus を使用する](#)
- [AWS SDK または CLI で GetBucketReplication を使用する](#)
- [CLI で GetBucketRequestPayment を使用する](#)
- [CLI で GetBucketTagging を使用する](#)
- [CLI で GetBucketVersioning を使用する](#)
- [AWS SDK または CLI で GetBucketWebsite を使用する](#)
- [AWS SDK または CLI で GetObject を使用する](#)
- [AWS SDK または CLI で GetObjectAcl を使用する](#)
- [AWS SDK または CLI で GetObjectAttributes を使用する](#)
- [AWS SDK または CLI で GetObjectLegalHold を使用する](#)
- [AWS SDK または CLI で GetObjectLockConfiguration を使用する](#)
- [AWS SDK または CLI で GetObjectRetention を使用する](#)
- [CLI で GetObjectTagging を使用する](#)
- [CLI で GetPublicAccessBlock を使用する](#)
- [AWS SDK または CLI で HeadBucket を使用する](#)
- [AWS SDK または CLI で HeadObject を使用する](#)
- [CLI で ListBucketAnalyticsConfigurations を使用する](#)
- [CLI で ListBucketInventoryConfigurations を使用する](#)

- [AWS SDK または CLI で ListBuckets を使用する](#)
- [AWS SDK または CLI で ListMultipartUploads を使用する](#)
- [AWS SDK または CLI で ListObjectVersions を使用する](#)
- [CLI で ListObjects を使用する](#)
- [AWS SDK または CLI で ListObjectsV2 を使用する](#)
- [AWS SDK または CLI で PutBucketAccelerateConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketAcl を使用する](#)
- [AWS SDK または CLI で PutBucketCors を使用する](#)
- [AWS SDK または CLI で PutBucketEncryption を使用する](#)
- [AWS SDK または CLI で PutBucketLifecycleConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketLogging を使用する](#)
- [CLI で PutBucketNotification を使用する](#)
- [AWS SDK または CLI で PutBucketNotificationConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketPolicy を使用する](#)
- [AWS SDK または CLI で PutBucketReplication を使用する](#)
- [CLI で PutBucketRequestPayment を使用する](#)
- [CLI で PutBucketTagging を使用する](#)
- [AWS SDK または CLI で PutBucketVersioning を使用する](#)
- [AWS SDK または CLI で PutBucketWebsite を使用する](#)
- [AWS SDK または CLI で PutObject を使用する](#)
- [AWS SDK または CLI で PutObjectAcl を使用する](#)
- [AWS SDK または CLI で PutObjectLegalHold を使用する](#)
- [AWS SDK または CLI で PutObjectLockConfiguration を使用する](#)
- [AWS SDK または CLI で PutObjectRetention を使用する](#)
- [AWS SDK または CLI で RestoreObject を使用する](#)
- [AWS SDK または CLI で SelectObjectContent を使用する](#)
- [AWS SDK または CLI で UploadPart を使用する](#)
- [AWS SDK または CLI で UploadPartCopy を使用する](#)
- [AWS SDK を使用した Amazon S3 のシナリオ](#)
  - [バケットが存在するかどうかを確認する](#)

- [オブジェクトが存在するかどうかを確認します。](#)
- [AWS SDK を使用して、テキストを音声に変換し、テキストに戻す](#)
- [AWS SDK を使用して Amazon S3 の署名付き URL を作成する](#)
- [ユーザーがラベルを使用して写真を管理できる写真アセット管理アプリケーションの作成](#)
- [AWS SDK を使用して Amazon S3 オブジェクトを一覧表示するウェブページ](#)
- [Amazon Textract エクスプローラーアプリケーションを作成する](#)
- [AWS SDK を使用して、指定した Amazon S3 バケットからすべてのオブジェクトを削除します。](#)
- [AWS SDK を使用して Amazon S3 への不完全なマルチパートアップロードを削除する](#)
- [AWS SDK を使用して、Amazon Rekognition で画像内の PPE を検出する](#)
- [AWS SDK を使用して、画像から抽出されたテキスト内のエンティティを検出する](#)
- [AWS SDK を使用して、画像内の顔を検出する](#)
- [AWS SDK を使用して、Amazon Rekognition で画像内のオブジェクトを検出する](#)
- [AWS SDK を使用して、Amazon Rekognition で、動画内の人やオブジェクトを検出する](#)
- [Amazon Simple Storage Service \(Amazon S3\) バケットから S3 の「ディレクトリ」をダウンロードする](#)
- [Amazon Simple Storage Service \(Amazon S3\) バケットからすべてのオブジェクトを、ローカルディレクトリにダウンロードする](#)
- [サイズが不明なストリームを、AWS SDK を使用して Amazon S3 オブジェクトからダウンロードする](#)
- [AWS SDK を使用してマルチリージョンアクセスポイントから Amazon S3 オブジェクトを取得する](#)
- [AWS SDK を使用して、Amazon S3 バケットから If-Modified-Since を指定してオブジェクトを取得する](#)
- [AWS SDK を使用して Amazon S3 オブジェクトの暗号化の使用を開始する](#)
- [AWS SDK を使用して Amazon S3 オブジェクトのタグの使用を開始する](#)
- [ビッグデータ処理クラスターの使用開始](#)
- [設定管理の使用開始](#)
- [ドキュメントテキスト抽出の使用開始](#)
- [機械学習特徴量ストアの使用開始](#)

- [クエリ分析の使用開始](#)
- [AWS SDK を使用して Amazon S3 オブジェクトロック機能を実操作する](#)
- [AWS SDK を使用して Amazon S3 の条件付きリクエストを行う](#)
- [AWS SDK を使用して Amazon S3 バケットのアクセスコントロールリスト \(ACL\) を管理する](#)
- [AWS SDK で、Amazon S3 を使用して大量の Amazon SQS メッセージを管理する](#)
- [AWS SDK を使用して、バージョン管理されている Amazon S3 オブジェクトを Lambda 関数でバッチで管理する](#)
- [AWS SDK を使用して Amazon S3 の URI を解析する](#)
- [AWS SDK を使用して Amazon S3 オブジェクトのマルチパートコピーを実行する](#)
- [AWS SDK を使用して Amazon S3 イベント通知を受信して処理する](#)
- [AWS SDK を使用して、EXIF とその他の画像情報を保存する](#)
- [AWS SDK を使用して S3 イベント通知を Amazon EventBridge に送信する](#)
- [AWS SDK を使用して Amazon S3 オブジェクトのアップロードまたはダウンロードを追跡する](#)
- [S3 Object Lambda でアプリケーションのデータを変換する](#)
- [AWS SDK による単体テストと統合テストのアプローチ例](#)
- [ローカルディレクトリを Amazon Simple Storage Service \(Amazon S3\) バケットに、再帰的にアップロードする](#)
- [AWS SDK を使用して Amazon S3 との間で大きなファイルをアップロードまたはダウンロードする](#)
- [サイズが不明なストリームを AWS SDK を使用して Amazon S3 オブジェクトにアップロードする](#)
- [チェックサムを使用した AWS SDK での Amazon S3 オブジェクトの操作](#)
- [AWS SDK を使用して Amazon S3 オブジェクトの整合性機能を実操作する](#)
- [AWS SDK を使用して、Amazon S3 のバージョン管理されたオブジェクトを実操作する](#)
- [Amazon S3 のサーバーレス例](#)
  - [Amazon S3 トリガーから Lambda 関数を呼び出す](#)

## AWS SDK を使用した Amazon S3 の基本的なコード例

次のコード例は、AWS SDK を使用した Amazon Simple Storage Service の基本的な使用方法を説明しています。

## 例

- [Hello Amazon S3](#)
- [AWS SDK を使用した Amazon S3 の基本的な使用方法を学ぶ](#)
- [AWS SDK を使用した Amazon S3 のアクション](#)
  - [AWS SDK または CLI で AbortMultipartUpload を使用する](#)
  - [AWS SDK または CLI で CompleteMultipartUpload を使用する](#)
  - [AWS SDK または CLI で CopyObject を使用する](#)
  - [AWS SDK または CLI で CreateBucket を使用する](#)
  - [AWS SDK で CreateMultiRegionAccessPoint を使用する](#)
  - [AWS SDK または CLI で CreateMultipartUpload を使用する](#)
  - [AWS SDK で CreatePresignedPost を使用する](#)
  - [AWS SDK または CLI で DeleteBucket を使用する](#)
  - [CLI で DeleteBucketAnalyticsConfiguration を使用する](#)
  - [AWS SDK または CLI で DeleteBucketCors を使用する](#)
  - [CLI で DeleteBucketEncryption を使用する](#)
  - [CLI で DeleteBucketInventoryConfiguration を使用する](#)
  - [AWS SDK または CLI で DeleteBucketLifecycle を使用する](#)
  - [CLI で DeleteBucketMetricsConfiguration を使用する](#)
  - [AWS SDK または CLI で DeleteBucketPolicy を使用する](#)
  - [CLI で DeleteBucketReplication を使用する](#)
  - [CLI で DeleteBucketTagging を使用する](#)
  - [AWS SDK または CLI で DeleteBucketWebsite を使用する](#)
  - [AWS SDK または CLI で DeleteObject を使用する](#)
  - [CLI で DeleteObjectTagging を使用する](#)
  - [AWS SDK または CLI で DeleteObjects を使用する](#)
  - [CLI で DeletePublicAccessBlock を使用する](#)
  - [CLI で GetBucketAccelerateConfiguration を使用する](#)
  - [AWS SDK または CLI で GetBucketAcl を使用する](#)
  - [CLI で GetBucketAnalyticsConfiguration を使用する](#)
  - [AWS SDK または CLI で GetBucketCors を使用する](#)

- [AWS SDK または CLI で GetBucketEncryption を使用する](#)
- [CLI で GetBucketInventoryConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketLifecycleConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketLocation を使用する](#)
- [CLI で GetBucketLogging を使用する](#)
- [CLI で GetBucketMetricsConfiguration を使用する](#)
- [CLI で GetBucketNotification を使用する](#)
- [AWS SDK または CLI で GetBucketPolicy を使用する](#)
- [CLI で GetBucketPolicyStatus を使用する](#)
- [AWS SDK または CLI で GetBucketReplication を使用する](#)
- [CLI で GetBucketRequestPayment を使用する](#)
- [CLI で GetBucketTagging を使用する](#)
- [CLI で GetBucketVersioning を使用する](#)
- [AWS SDK または CLI で GetBucketWebsite を使用する](#)
- [AWS SDK または CLI で GetObject を使用する](#)
- [AWS SDK または CLI で GetObjectAcl を使用する](#)
- [AWS SDK または CLI で GetObjectAttributes を使用する](#)
- [AWS SDK または CLI で GetObjectLegalHold を使用する](#)
- [AWS SDK または CLI で GetObjectLockConfiguration を使用する](#)
- [AWS SDK または CLI で GetObjectRetention を使用する](#)
- [CLI で GetObjectTagging を使用する](#)
- [CLI で GetPublicAccessBlock を使用する](#)
- [AWS SDK または CLI で HeadBucket を使用する](#)
- [AWS SDK または CLI で HeadObject を使用する](#)
- [CLI で ListBucketAnalyticsConfigurations を使用する](#)
- [CLI で ListBucketInventoryConfigurations を使用する](#)
- [AWS SDK または CLI で ListBuckets を使用する](#)
- [AWS SDK または CLI で ListMultipartUploads を使用する](#)
- [AWS SDK または CLI で ListObjectVersions を使用する](#)
- [CLI で ListObjects を使用する](#)

- [AWS SDK または CLI で ListObjectsV2 を使用する](#)
- [AWS SDK または CLI で PutBucketAccelerateConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketAcl を使用する](#)
- [AWS SDK または CLI で PutBucketCors を使用する](#)
- [AWS SDK または CLI で PutBucketEncryption を使用する](#)
- [AWS SDK または CLI で PutBucketLifecycleConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketLogging を使用する](#)
- [CLI で PutBucketNotification を使用する](#)
- [AWS SDK または CLI で PutBucketNotificationConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketPolicy を使用する](#)
- [AWS SDK または CLI で PutBucketReplication を使用する](#)
- [CLI で PutBucketRequestPayment を使用する](#)
- [CLI で PutBucketTagging を使用する](#)
- [AWS SDK または CLI で PutBucketVersioning を使用する](#)
- [AWS SDK または CLI で PutBucketWebsite を使用する](#)
- [AWS SDK または CLI で PutObject を使用する](#)
- [AWS SDK または CLI で PutObjectAcl を使用する](#)
- [AWS SDK または CLI で PutObjectLegalHold を使用する](#)
- [AWS SDK または CLI で PutObjectLockConfiguration を使用する](#)
- [AWS SDK または CLI で PutObjectRetention を使用する](#)
- [AWS SDK または CLI で RestoreObject を使用する](#)
- [AWS SDK または CLI で SelectObjectContent を使用する](#)
- [AWS SDK または CLI で UploadPart を使用する](#)
- [AWS SDK または CLI で UploadPartCopy を使用する](#)

## Hello Amazon S3

次のコード例は、Amazon S3 の使用を開始する方法を示しています。



## .NET

### SDK for .NET (v4)

#### Note

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

```
/// <summary>
/// Hello Amazon Simple Storage Service
/// (Amazon S3) example.
/// </summary>
public class HelloS3
{
    /// <summary>
    /// Main method to run the Hello S3 example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var s3Client = new AmazonS3Client();

        try
        {
            Console.WriteLine("Hello Amazon S3! Let's list your buckets:");
            Console.WriteLine(new string('-', 80));

            // Use the built-in paginator to list buckets
            var request = new ListBucketsRequest();
            var paginator = s3Client.Paginators.ListBuckets(request);

            var buckets = new List<S3Bucket>();

            await foreach (var response in paginator.Responses)
            {
                buckets.AddRange(response.Buckets);
            }

            if (buckets.Any())
```



```
        {
            Console.WriteLine($"Found {buckets.Count} S3 buckets:");
            Console.WriteLine();

            foreach (var bucket in buckets)
            {
                Console.WriteLine($"- Bucket Name: {bucket.BucketName}");
                Console.WriteLine($"  Creation Date:
{bucket.CreationDate:yyyy-MM-dd HH:mm:ss UTC}");
                Console.WriteLine();
            }
        }
        else
        {
            Console.WriteLine("No S3 buckets found in your account.");
        }

        Console.WriteLine("Hello S3 completed successfully.");
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"S3 service error occurred: {ex.Message}");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Couldn't list S3 buckets. Here's why:
{ex.Message}");
    }
}
}
```

- APIの詳細については、AWS SDK for .NET API リファレンスの「[ListBuckets](#)」を参照してください。

## C++

## SDK for C++

 Note

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

CMakeLists.txt CMake ファイルのコード。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
  libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
      "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
  endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
  # Copy relevant AWS SDK for C++ libraries into the current binary directory
  for running and debugging.
```

```
# set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
may need to uncomment this
# and set the proper subdirectory to the executables' location.

AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})
```

hello\_s3.cpp ソースファイルのコード。

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 * A "Hello S3" starter application which initializes an Amazon Simple Storage
 * Service (Amazon S3) client
 * and lists the Amazon S3 buckets in the selected region.
 *
 * main function
 *
 * Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
```

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

// You don't normally have to test that you are authenticated. But the
// S3 service permits anonymous requests, thus the s3Client will return "success"
// and 0 buckets even if you are unauthenticated, which can be confusing to a new
// user.

auto provider =
    Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
auto creds = provider->GetAWSCredentials();
if (creds.IsEmpty()) {
    std::cerr << "Failed authentication" << std::endl;
}

Aws::S3::S3Client s3Client(clientConfig);
auto outcome = s3Client.ListBuckets();

if (!outcome.IsSuccess()) {
    std::cerr << "Failed with error: " << outcome.GetError() <<
std::endl;
    result = 1;
} else {
    std::cout << "Found " << outcome.GetResult().GetBuckets().size()
        << " buckets\n";
    for (auto &bucket: outcome.GetResult().GetBuckets()) {
        std::cout << bucket.GetName() << std::endl;
    }
}

}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}
```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[ListBuckets](#)」を参照してください。

## Go

## SDK for Go V2

 Note

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

```
package main

import (
    "context"
    "errors"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    s3Client := s3.NewFromConfig(sdkConfig)
    count := 10
    fmt.Printf("Let's list up to %v buckets for your account.\n", count)
    result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        var ae smithy.APIError
```



```
/**
 * Before running this Java V2 code 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 HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    /**
     * Lists all the S3 buckets associated with the provided AWS S3 client.
     *
     * @param s3 the S3Client instance used to interact with the AWS S3 service
     */
    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- APIの詳細については、AWS SDK for Java 2.x API リファレンスの「[ListBuckets](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the S3 buckets in your configured AWS account.
 */
export const helloS3 = async () => {
  // When no region or credentials are provided, the SDK will use the
  // region and credentials from the local AWS config.
  const client = new S3Client({});

  try {
    /**
     * @type { import("@aws-sdk/client-s3").Bucket[] }
     */
    const buckets = [];

    for await (const page of paginateListBuckets({ client }, {})) {
      buckets.push(...page.Buckets);
    }
    console.log("Buckets: ");
    console.log(buckets.map((bucket) => bucket.Name).join("\n"));
    return buckets;
  } catch (caught) {
    // ListBuckets does not throw any modeled errors. Any error caught
    // here will be something generic like `AccessDenied`.
    if (caught instanceof S3ServiceException) {
      console.error(`${caught.name}: ${caught.message}`);
    } else {
```



```
        // Something besides S3 failed.  
        throw caught;  
    }  
}  
};
```

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

## PHP

### SDK for PHP

#### Note

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

```
use Aws\S3\S3Client;  
  
$client = new S3Client(['region' => 'us-west-2']);  
$results = $client->listBuckets();  
var_dump($results);
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

```
import boto3

def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) client and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """

    # Create an S3 client.
    s3_client = boto3.client("s3")

    print("Hello, Amazon S3! Let's list your buckets:")

    # Create a paginator for the list_buckets operation.
    paginator = s3_client.get_paginator("list_buckets")

    # Use the paginator to get a list of all buckets.
    response_iterator = paginator.paginate(
        PaginationConfig={
            "PageSize": 50, # Adjust PageSize as needed.
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response.
    buckets_found = False
    for page in response_iterator:
        if "Buckets" in page and page["Buckets"]:
            buckets_found = True
            for bucket in page["Buckets"]:
                print(f"\t{bucket['Name']}")

    if not buckets_found:
        print("No buckets found!")

if __name__ == "__main__":
    hello_s3()
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end
  end

  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end
```

```
if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
  manager.list_buckets
end
```

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

## Rust

### SDK for Rust

#### Note

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

```
/// S3 Hello World Example using the AWS SDK for Rust.
///
/// This example lists the objects in a bucket, uploads an object to that bucket,
/// and then retrieves the object and prints some S3 information about the
/// object.
/// This shows a number of S3 features, including how to use built-in paginators
/// for large data sets.
///
/// # Arguments
///
/// * `client` - an S3 client configured appropriately for the environment.
/// * `bucket` - the bucket name that the object will be uploaded to. Must be
///   present in the region the `client` is configured to use.
/// * `filename` - a reference to a path that will be read and uploaded to S3.
/// * `key` - the string key that the object will be uploaded as inside the
///   bucket.
async fn list_bucket_and_upload_object(
  client: &aws_sdk_s3::Client,
  bucket: &str,
```

```
    filepath: &Path,
    key: &str,
) -> Result<(), S3ExampleError> {
    // List the buckets in this account
    let mut objects = client
        .list_objects_v2()
        .bucket(bucket)
        .into_paginator()
        .send();

    println!("key\tetag\tlast_modified\tstorage_class");
    while let Some(Ok(object)) = objects.next().await {
        for item in object.contents() {
            println!(
                "{}\t{}\t{}\t{}",
                item.key().unwrap_or_default(),
                item.e_tag().unwrap_or_default(),
                item.last_modified()
                    .map(|lm| format!("{lm}"))
                    .unwrap_or_default(),
                item.storage_class()
                    .map(|sc| format!("{sc}"))
                    .unwrap_or_default()
            );
        }
    }
}

// Prepare a ByteStream around the file, and upload the object using that
// ByteStream.
let body = aws_sdk_s3::primitives::ByteStream::from_path(filepath)
    .await
    .map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to create bytestream for {filepath:?} ({err:?})"
        ))
    })?;
let resp = client
    .put_object()
    .bucket(bucket)
    .key(key)
    .body(body)
    .send()
    .await?;
```

```
println!(
    "Upload success. Version: {:?}" ,
    resp.version_id()
        .expect("S3 Object upload missing version ID")
);

// Retrieve the just-uploaded object.
let resp = client.get_object().bucket(bucket).key(key).send().await?;
println!("etag: {}", resp.e_tag().unwrap_or("(missing)"));
println!("version: {}", resp.version_id().unwrap_or("(missing)"));

Ok(())
}
```

S3ExampleError ユーティリティ。

```
/// S3ExampleError provides a From<T: ProvideErrorMetadata> impl to extract
/// client-specific error details. This serves as a consistent backup to handling
/// specific service errors, depending on what is needed by the scenario.
/// It is used throughout the code examples for the AWS SDK for Rust.
#[derive(Debug)]
pub struct S3ExampleError(String);
impl S3ExampleError {
    pub fn new(value: impl Into<String>) -> Self {
        S3ExampleError(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        S3ExampleError(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: aws_sdk_s3::error::ProvideErrorMetadata> From<T> for S3ExampleError {
    fn from(value: T) -> Self {
        S3ExampleError(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
        ))
    }
}
```

```
        .map(String::from)
        .unwrap_or("missing reason".into()),
    ))
}

impl std::error::Error for S3ExampleError {}

impl std::fmt::Display for S3ExampleError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用した Amazon S3 の基本的な使用方法を学ぶ

次のコード例は、以下を実行する方法を示しています。

- バケットを作成し、そこにファイルをアップロードします。
- バケットからオブジェクトをダウンロードします。
- バケット内のサブフォルダにオブジェクトをコピーします。
- バケット内のオブジェクトを一覧表示します。
- バケットオブジェクトとバケットを削除します。

## .NET

### SDK for .NET (v4)

#### Note

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

Amazon S3 機能を示すインタラクティブなシナリオを実行します。

```
public class S3_Basics
{
    public static bool IsInteractive = true;
    public static string BucketName = null!;
    public static string TempFilePath = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static ILogger<S3_Basics> _logger = null!;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
            {
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddLogging(builder => builder.AddConsole());
            })
            .Build();

        _logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<S3_Basics>();

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();

        var sepBar = new string('-', 45);

        Console.WriteLine(sepBar);
        Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
        Console.WriteLine("procedures. This application will:");
        Console.WriteLine("\n\t1. Create a bucket");
        Console.WriteLine("\n\t2. Upload an object to the new bucket");
```



```
        Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
        Console.WriteLine("\n\t4. List the items in the new bucket");
        Console.WriteLine("\n\t5. Delete all the items in the bucket");
        Console.WriteLine("\n\t6. Delete the bucket");
        Console.WriteLine(sepBar);

        await RunScenario(_s3Wrapper, _logger);

        Console.WriteLine(sepBar);
        Console.WriteLine("The Amazon S3 scenario has successfully completed.");
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Run the S3 Basics scenario with injected dependencies.
    /// </summary>
    /// <param name="s3Wrapper">The S3 wrapper instance.</param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    public static async Task RunScenario(S3Wrapper s3Wrapper, ILogger<S3_Basics>
scenarioLogger)
    {
        string bucketName = BucketName;
        string filePath = TempFilePath;
        string keyName = string.Empty;

        var sepBar = new string('-', 45);

        try
        {
            // Create a bucket.
            Console.WriteLine($"{sepBar}");
            Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");
            Console.WriteLine(sepBar);

            if (IsInteractive)
            {
                Console.Write("Please enter a name for the new bucket: ");
                bucketName = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Using bucket name: {bucketName}");
            }
        }
    }
}
```

```
    }

    var success = await s3Wrapper.CreateBucketAsync(bucketName);
    if (success)
    {
        Console.WriteLine($"Successfully created bucket: {bucketName}.
\n");
    }
    else
    {
        Console.WriteLine($"Could not create bucket: {bucketName}.\n");
    }

    Console.WriteLine(sepBar);
    Console.WriteLine("Upload a file to the new bucket.");
    Console.WriteLine(sepBar);

    if (IsInteractive)
    {
        // Get the local path and filename for the file to upload.
        while (string.IsNullOrEmpty(filePath))
        {
            Console.Write("Please enter the path and filename of the file
to upload: ");
            filePath = Console.ReadLine();

            // Confirm that the file exists on the local computer.
            if (!File.Exists(filePath))
            {
                Console.WriteLine($"Couldn't find {filePath}. Try again.
\n");
                filePath = string.Empty;
            }
        }
    }
    else
    {
        // Use the public variable if set, otherwise create a temp file
        if (!string.IsNullOrEmpty(TempFilePath))
        {
            filePath = TempFilePath;
            Console.WriteLine($"Using provided test file: {filePath}");
        }
        else
```

```
        {
            // Create a temporary test file for non-interactive mode
            filePath = Path.GetTempFileName();
            var testContent = "This is a test file for S3 basics
scenario.\nGenerated on: " + DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss
UTC");

            await File.WriteAllTextAsync(filePath, testContent);
            Console.WriteLine($"Created temporary test file:
{filePath}");
        }
    }

    // Get the file name from the full path.
    keyName = Path.GetFileName(filePath);

    success = await s3Wrapper.UploadFileAsync(bucketName, keyName,
filePath);

    if (success)
    {
        Console.WriteLine($"Successfully uploaded {keyName} from
{filePath} to {bucketName}.\n");
    }
    else
    {
        Console.WriteLine($"Could not upload {keyName}.\n");
    }

    // Set up download path
    string downloadPath = string.Empty;

    if (IsInteractive)
    {
        // Now get a new location where we can save the file.
        while (string.IsNullOrEmpty(downloadPath))
        {
            // First get the path to which the file will be downloaded.
            Console.Write("Please enter the path where the file will be
downloaded: ");

            downloadPath = Console.ReadLine();

            // Confirm that the file doesn't already exist on the local
            computer.

            if (File.Exists($"{downloadPath}\\{keyName}"))
```

```
        {
            Console.WriteLine($"Sorry, the file already exists in
that location.\n");
            downloadPath = string.Empty;
        }
    }
else
{
    downloadPath = Path.GetTempPath();
    var downloadFile = Path.Combine(downloadPath, keyName);
    if (File.Exists(downloadFile))
    {
        File.Delete(downloadFile);
    }

    Console.WriteLine($"Using download path: {downloadPath}");
}

// Download an object from a bucket.
success = await s3Wrapper.DownloadObjectFromBucketAsync(bucketName,
keyName, downloadPath);

if (success)
{
    Console.WriteLine($"Successfully downloaded {keyName}.\n");
}
else
{
    Console.WriteLine($"Sorry, could not download {keyName}.\n");
}

// Copy the object to a different folder in the bucket.
string folderName = string.Empty;

if (IsInteractive)
{
    while (string.IsNullOrEmpty(folderName))
    {
        Console.Write("Please enter the name of the folder to copy
your object to: ");
        folderName = Console.ReadLine();
    }
}
```

```
        else
        {
            folderName = "test-folder";
            Console.WriteLine($"Using folder name: {folderName}");
        }

        await s3Wrapper.CopyObjectInBucketAsync(bucketName, keyName,
folderName);

        // List the objects in the bucket.
        await s3Wrapper.ListBucketContentsAsync(bucketName);

        // Delete the contents of the bucket.
        if (IsInteractive)
        {
            Console.WriteLine("Press <Enter> when you are ready to delete the
bucket contents.");
            _ = Console.ReadLine();
        }

        var deleteContentsSuccess = await
s3Wrapper.DeleteBucketContentsAsync(bucketName);
        if (deleteContentsSuccess)
        {
            Console.WriteLine($"Successfully deleted contents of
{bucketName}.\n");
        }
        else
        {
            Console.WriteLine($"Sorry, could not delete contents of
{bucketName}.\n");
        }

        if (IsInteractive)
        {
            // Deleting the bucket too quickly after separately deleting its
contents can
            // cause an error that the bucket isn't empty. To delete contents
and bucket in one
            // operation, use AmazonS3Util.DeleteS3BucketWithObjectsAsync
            Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
            _ = Console.ReadLine();
        }
    }
```

```
        else
        {
            // Add a small delay for non-interactive mode to ensure objects
are fully deleted.
            Console.WriteLine("Waiting a moment for objects to be fully
deleted...");
            await Task.Delay(2000);
        }

        // Delete the bucket.
        var deleteSuccess = await s3Wrapper.DeleteBucketAsync(bucketName);
        if (deleteSuccess)
        {
            Console.WriteLine($"Successfully deleted {bucketName}.\n");
        }
        else
        {
            Console.WriteLine($"Sorry, could not delete {bucketName}.\n");
        }

        // Clean up temporary files in non-interactive mode
        if (!IsInteractive)
        {
            try
            {
                if (File.Exists(filePath))
                {
                    File.Delete(filePath);
                    Console.WriteLine("Cleaned up temporary test file.");
                }

                var downloadFile = Path.Combine(downloadPath, keyName);
                if (File.Exists(downloadFile))
                {
                    File.Delete(downloadFile);
                    Console.WriteLine("Cleaned up downloaded test file.");
                }
            }
            catch (Exception ex)
            {
                scenarioLogger.LogWarning(ex, "Failed to clean up temporary
files.");
            }
        }
    }
```

```
    }
    catch (Exception ex)
    {
        scenarioLogger.LogError(ex, "An error occurred during the S3 scenario
execution.");

        // Clean up on error - delete bucket if it exists
        try
        {
            if (!string.IsNullOrEmpty(bucketName))
            {
                await s3Wrapper.DeleteBucketContentsAsync(bucketName);
                await s3Wrapper.DeleteBucketAsync(bucketName);
            }
        }
        catch (Exception cleanupEx)
        {
            scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
        }

        // Clean up temporary files in non-interactive mode
        if (!IsInteractive)
        {
            try
            {
                if (!string.IsNullOrEmpty(filePath) && File.Exists(filePath))
                {
                    File.Delete(filePath);
                }
            }
            catch (Exception fileCleanupEx)
            {
                scenarioLogger.LogWarning(fileCleanupEx, "Failed to clean up
temporary files during error handling.");
            }
        }

        throw;
    }
}
```

## Amazon S3 SDK メソッドのラッパークラス。

```
using Amazon.S3;
using Amazon.S3.Model;

namespace S3_Actions;

/// <summary>
/// This class contains all of the methods for working with Amazon Simple
/// Storage Service (Amazon S3) buckets.
/// </summary>
public class S3Wrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Initializes a new instance of the <see cref="S3Wrapper"/> class.
    /// </summary>
    /// <param name="amazonS3">An initialized Amazon S3 client object.</param>
    public S3Wrapper(IAmazonS3 amazonS3)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Shows how to create a new Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>A boolean value representing the success or failure of
    /// the bucket creation process.</returns>
    public async Task<bool> CreateBucketAsync(string bucketName)
    {
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
            };

            var response = await _amazonS3.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
}
```



```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Shows how to upload a file from the local computer to an Amazon S3
    /// bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to which the object
    /// will be uploaded.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object
    /// on the local computer to upload.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }
}
```

```
/// <summary>
/// Shows how to download an object from an Amazon S3 bucket to the
/// local computer.
/// </summary>
/// <param name="bucketName">The name of the bucket where the object is
/// currently stored.</param>
/// <param name="objectName">The name of the object to download.</param>
/// <param name="filePath">The path, including filename, where the
/// downloaded object will be stored.</param>
/// <returns>A boolean value indicating the success or failure of the
/// download process.</returns>
public async Task<bool> DownloadObjectFromBucketAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    var request = new GetObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
    };

    using GetObjectResponse response = await
        _amazonS3.GetObjectAsync(request);

    try
    {
        // Save object to local file
        await response.WriteResponseStreamToFileAsync($"{filePath}\
        \{objectName}", true, CancellationTokens.None);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error saving {objectName}: {ex.Message}");
        return false;
    }
}

/// <summary>
```

```
/// Copies an object in an Amazon S3 bucket to a folder within the
/// same bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// object to copy is located.</param>
/// <param name="objectName">The object to be copied.</param>
/// <param name="folderName">The folder to which the object will
/// be copied.</param>
/// <returns>A boolean value that indicates the success or failure of
/// the copy operation.</returns>
public async Task<bool> CopyObjectInBucketAsync(
    string bucketName,
    string objectName,
    string folderName)
{
    try
    {
        var request = new CopyObjectRequest
        {
            SourceBucket = bucketName,
            SourceKey = objectName,
            DestinationBucket = bucketName,
            DestinationKey = $"{folderName}\\{objectName}",
        };
        var response = await _amazonS3.CopyObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error copying object: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Shows how to list the objects in an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket for which to list.
/// <param name="printList">True to print out the list.
/// <returns>The collection of objects.</returns>
public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
    bool printList = true)
```

```
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
        };

        if (printList)
        {
            Console.WriteLine("-----");
            Console.WriteLine($"Listing the contents of {bucketName}:");
            Console.WriteLine("-----");
        }

        var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
        {
            BucketName = bucketName,
        });
        var s3objects = new List<S3Object>();
        await foreach (var response in listObjectsV2Paginator.Responses)
        {
            if (response.S3Objects != null)
            {
                s3objects.AddRange(response.S3Objects);
            }
        }

        if (printList)
        {
            Console.WriteLine($"Number of Objects: {s3objects.Count}");
            foreach (var entry in s3objects)
            {
                Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
            }
        }

        return s3objects;
    }
    catch (AmazonS3Exception ex)
    {

```

```

        Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' getting list of objects.");
        return null;
    }
}

/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>
public async Task<bool> DeleteBucketContentsAsync(string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
    try
    {
        // Delete all objects in the bucket.
        var deleteList = await ListBucketContentsAsync(bucketName, false);
        if (deleteList != null && deleteList.Any())
        {
            await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
            });
        }

        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting objects: {ex.Message}");
        return false;
    }
}

/// <summary>

```

```
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest { BucketName = bucketName, };

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting bucket: {ex.Message}");
        return false;
    }
}
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)

## Bash

### AWS CLIBash スクリプトを使用した

#### Note

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

```
#####
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#     0 - If successful.
#     1 - If an error occurred.
#####
function s3_getting_started() {
    {
        if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
            cd bucket-lifecycle-operations || exit

            source ./bucket_operations.sh
            cd ..
        fi
    }

    echo_repeat "*" 88
    echo "Welcome to the Amazon S3 getting started demo."
    echo_repeat "*" 88
    echo "A unique bucket will be created by appending a Universally Unique
Identifier to a bucket name prefix."
    echo -n "Enter a prefix for the S3 bucket that will be used in this demo: "
    get_input
    bucket_name_prefix=$get_input_result
    local bucket_name
    bucket_name=$(generate_random_name "$bucket_name_prefix")

    local region_code
    region_code=$(aws configure get region)
```

```
if create_bucket -b "$bucket_name" -r "$region_code"; then
    echo "Created demo bucket named $bucket_name"
else
    errecho "The bucket failed to create. This demo will exit."
    return 1
fi

local file_name
while [ -z "$file_name" ]; do
    echo -n "Enter a file you want to upload to your bucket: "
    get_input
    file_name=$get_input_result

    if [ ! -f "$file_name" ]; then
        echo "Could not find file $file_name. Are you sure it exists?"
        file_name=""
    fi
done

local key
key="$(basename "$file_name")"

local result=0
if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
    echo "Uploaded file $file_name into bucket $bucket_name with key $key."
else
    result=1
fi

local destination_file
destination_file="$file_name.download"
if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
    if download_object_from_bucket "$bucket_name" "$destination_file" "$key";
then
        echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
    else
        result=1
    fi
fi
fi
```



```
if yes_no_input "Would you like to copy $key a new object key in your bucket?
(y/n) "; then
    local to_key
    to_key="demo/$key"
    if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
        echo "Copied $key in the bucket $bucket_name to the $to_key."
    else
        result=1
    fi
fi

local bucket_items
bucket_items=$(list_items_in_bucket "$bucket_name")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    result=1
fi

echo "Your bucket contains the following items."
echo -e "Name\t\tSize"
echo "$bucket_items"

if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
(y/n) "; then
    bucket_items=$(echo "$bucket_items" | cut -f 1)

    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
        echo "The following items were deleted from the bucket $bucket_name"
        echo "$bucket_items"
    else
        result=1
    fi

    if delete_bucket "$bucket_name"; then
        echo "Deleted the bucket $bucket_name"
    else
        result=1
    fi
fi

return $result
}
```

このシナリオで使用する Amazon S3 関数。

```
#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
#                        create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally
unique."
        echo "  [-r region_code]  The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
            ;;
        esac
    done
```

```

        return 0
        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done

if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
fi

local bucket_config_arg
# A location constraint for "us-east-1" returns an error.
if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
fi

iecho "Parameters:\n"
iecho "    Bucket name:  $bucket_name"
iecho "    Region code:  $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
fi

```

```

}

#####
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.
#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

    response=$(aws s3api put-object \
        --bucket "$bucket_name" \
        --body "$source_file" \
        --key "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}

#####
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#     $1 - The name of the bucket to download the object from.
#     $2 - The path and file name to store the downloaded bucket.
#     $3 - The key (name) of the object in the bucket.
#

```

```

# Returns:
#      0 - If successful.
#      1 - If it fails.
#####
function download_object_from_bucket() {
    local bucket_name=$1
    local destination_file_name=$2
    local object_name=$3
    local response

    response=$(aws s3api get-object \
        --bucket "$bucket_name" \
        --key "$object_name" \
        "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}

#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#      $1 - The name of the bucket to copy the file from and to.
#      $2 - The key of the source file to copy.
#      $3 - The key of the destination file.
#
# Returns:
#      0 - If successful.
#      1 - If it fails.
#####
function copy_item_in_bucket() {
    local bucket_name=$1
    local source_key=$2
    local destination_key=$3
    local response

    response=$(aws s3api copy-object \
        --bucket "$bucket_name" \

```

```

--copy-source "$bucket_name/$source_key" \
--key "$destination_key")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
    return 1
fi
}

#####
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     The list of files in text format.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function list_items_in_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api list-objects \
        --bucket "$bucket_name" \
        --output text \
        --query 'Contents[].{Key: Key, Size: Size}')

    # shellcheck disable=SC2181
    if [[ ${?} -eq 0 ]]; then
        echo "$response"
    else
        errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
        return 1
    fi
}

```

```
#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - A list of keys in the bucket to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":["
    for key in $keys; do
        delete_items="$delete_items{\"Key\": \"$key\"},"
    done
    delete_items=${delete_items%?} # Remove the final comma.
    delete_items="$delete_items]"

    response=$(aws s3api delete-objects \
        --bucket "$bucket_name" \
        --delete "$delete_items")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
        return 1
    fi
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
```

```
# Parameters:
#     $1 - The name of the bucket.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api delete-bucket \
        --bucket "$bucket_name")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
        return 1
    fi
}
```

- API の詳細については、「AWS CLI コマンドリファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)



## C++

## SDK for C++

 Note

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

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
            \param bucketName: The S3 bucket's name.
            \param client: An S3 client.
            \return bool: Function succeeded.
        */
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
```

```

        \param bucketName: The S3 bucket's name.
        \param key: The key for the object in the S3 bucket.
        \param client: An S3 client.
        \return bool: Function succeeded.
    */
    static bool
    deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
&key,
                           Aws::S3::S3Client &client);
    }
}

//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
    \param bucketNamePrefix: A prefix for a bucket name.
    \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
    \param saveFilePath: Path for saving a downloaded S3 object.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &bucketNamePrefix,
      const Aws::String &uploadFilePath,
      const Aws::String &saveFilePath,
      const Aws::Client::ClientConfiguration
&clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: <bucketNamePrefix> + "-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = bucketNamePrefix +
        Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(

```

```

    Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
        clientConfig.region));
    request.WithCreateBucketConfiguration(createBucketConfiguration);
}

    Aws::S3::Model::CreateBucketOutcome outcome =
client.CreateBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        return false;
    } else {
        std::cout << "Created the bucket, '" << bucketName <<
            "', in the region, '" << clientConfig.region << "'." <<
std::endl;
    }
}

// 2. Upload a local file to the bucket.
Aws::String key = "key-for-test";
{
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    std::shared_ptr<Aws::FStream> input_data =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
            uploadFilePath,
            std::ios_base::in |
            std::ios_base::binary);

    if (!input_data->is_open()) {
        std::cerr << "Error: unable to open file, '" << uploadFilePath <<
            "'." << std::endl;
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    }

    request.SetBody(input_data);
}

```

```

    Aws::S3::Model::PutObjectOutcome outcome =
        client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
            outcome.GetError().GetMessage() << std::endl;
        AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    } else {
        std::cout << "Added the object with the key, '" << key
            << "', to the bucket, '"
            << bucketName << "'." << std::endl;
    }
}

// 3. Download the object to a local file.
{
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Downloaded the object with the key, '" << key
            << "', in the bucket, '"
            << bucketName << "'." << std::endl;

        Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
            GetBody();
        Aws::OFStream outStream(saveFilePath,
                                std::ios_base::out | std::ios_base::binary);
        if (!outStream.is_open()) {
            std::cout << "Error: unable to open file, '" << saveFilePath <<
                "'."
                << std::endl;

```

```
        } else {
            outputStream << ioStream.rdbuf();
            std::cout << "Wrote the downloaded object to the file '"
                      << saveFilePath << "'.\" << std::endl;
        }
    }
}

// 4. Copy the object to a different "folder" in the bucket.
Aws::String copiedToKey = "test-folder/" + key;
{
    Aws::S3::Model::CopyObjectRequest request;
    request.WithBucket(bucketName)
           .WithKey(copiedToKey)
           .WithCopySource(bucketName + "/" + key);

    Aws::S3::Model::CopyObjectOutcome outcome =
        client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error: copyObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Copied the object with the key, '" << key
                  << "', to the key, '" << copiedToKey
                  << "', in the bucket, '" << bucketName << "'.\" << std::endl;
    }
}

// 5. List objects in the bucket.
{
    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken;
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }
        Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
            request);

        if (!outcome.IsSuccess()) {
```

```

        std::cerr << "Error: ListObjects: " <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    } else {
        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();
        allObjects.insert(allObjects.end(), objects.begin(),
objects.end());
        continuationToken = outcome.GetResult().GetContinuationToken();
    }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " objects in the bucket, '" <<
bucketName
        << "':" << std::endl;

    for (Aws::S3::Model::Object &object: allObjects) {
        std::cout << "        '" << object.GetKey() << "'" << std::endl;
    }
}

// 6. Delete all objects in the bucket.
// All objects in the bucket must be deleted before deleting the bucket.
AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

// 7. Delete the bucket.
return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {

```

```
        std::cout << "Deleted the object with the key, '" << key
                    << "', from the bucket, '"
                    << bucketName << "'.'" << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
&client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                    err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'.'" << std::endl;
    }
    return outcome.IsSuccess();
}
```

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

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

## Go

## SDK for Go V2

 Note

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

シナリオで使用するバケットアクションとオブジェクトアクションをラップする構造体を定義します。

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "io"
    "log"
    "os"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}
```



```
// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
    error) {
    var err error
    var output *s3.ListBucketsOutput
    var buckets []types.Bucket
    bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
        &s3.ListBucketsInput{})
    for bucketPaginator.HasMorePages() {
        output, err = bucketPaginator.NextPage(ctx)
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
                fmt.Println("You don't have permission to list buckets for this account.")
                err = apiErr
            } else {
                log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
            }
            break
        } else {
            buckets = append(buckets, output.Buckets...)
        }
    }
    return buckets, err
}

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
    (bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
    exists := true
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
            }
            err = nil
        }
    }
}
```

```
    default:
        log.Printf("Either you don't have access to bucket %v or another error
occurred. "+
        "Here's what happened: %v\n", bucketName, err)
    }
}
} else {
    log.Printf("Bucket %v exists and you already own it.", bucketName)
}

return exists, err
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    })
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", name)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", name)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
        }
    }
    return err
}
```

```
// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
            Body:   file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
                log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
                    fileName, bucketName, objectKey, err)
            }
        } else {
            err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                    aws.String(objectKey)}, time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
            }
        }
    }
    return err
}

// UploadLargeObject uses an upload manager to upload data to an object in a
// bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
```

```

func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
    string, objectKey string, largeObject []byte) error {
    largeBuffer := bytes.NewReader(largeObject)
    var partMiBs int64 = 10
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
        u.PartSize = partMiBs * 1024 * 1024
    })
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    largeBuffer,
    })
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
            log.Printf("Error while uploading object to %s. The object is too large.\n"+
                "The maximum size for a multipart upload is 5TB.", bucketName)
        } else {
            log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
                bucketName, objectKey, err)
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }

    return err
}

// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey

```

```

    if errors.As(err, &noKey) {
        log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
            objectKey, bucketName)
        err = noKey
    } else {
        log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
            objectKey, err)
    }
    return err
}
defer result.Body.Close()
file, err := os.Create(fileName)
if err != nil {
    log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
    return err
}
defer file.Close()
body, err := io.ReadAll(result.Body)
if err != nil {
    log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
        err)
}
_, err = file.Write(body)
return err
}

// DownloadLargeObject uses a download manager to download an object from a
// bucket.
// The download manager gets the data in parts and writes them to a buffer until
// all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
    string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10
    downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
    {
        d.PartSize = partMiBs * 1024 * 1024
    })
    buffer := manager.NewWriteAtBuffer([]byte{})
    _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
}

```

```
    })
    if err != nil {
        log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return buffer.Bytes(), err
}

// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(ctx context.Context, bucketName string,
    objectKey string, folderName string) error {
    objectDest := fmt.Sprintf("%v/%v", folderName, objectKey)
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(bucketName),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
        Key:         aws.String(objectDest),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
                active tier.\n",
                objectKey, bucketName)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                aws.String(objectDest)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectDest)
        }
    }
    return err
}

// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
```

```
    Bucket:      aws.String(destinationBucket),
    CopySource:  aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
    Key:         aws.String(objectKey),
  })
  if err != nil {
    var notActive *types.ObjectNotInActiveTierError
    if errors.As(err, &notActive) {
      log.Printf("Couldn't copy object %s from %s because the object isn't in the
active tier.\n",
        objectKey, sourceBucket)
      err = notActive
    }
  } else {
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
      ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
aws.String(objectKey)}, time.Minute)
    if err != nil {
      log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
  }
  return err
}

// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
([]types.Object, error) {
  var err error
  var output *s3.ListObjectsV2Output
  input := &s3.ListObjectsV2Input{
    Bucket: aws.String(bucketName),
  }
  var objects []types.Object
  objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
  for objectPaginator.HasMorePages() {
    output, err = objectPaginator.NextPage(ctx)
    if err != nil {
      var noBucket *types.NoSuchBucket
      if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucketName)
        err = noBucket
      }
    }
    break
  }
}
```

```
    } else {
        objects = append(objects, output.Contents...)
    }
}
return objects, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(ctx context.Context, bucketName string,
    objectKeys []string) error {
    var objectIds []types.ObjectIdentifier
    for _, key := range objectKeys {
        objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
    }
    output, err := basics.S3Client.DeleteObjects(ctx, &s3.DeleteObjectsInput{
        Bucket: aws.String(bucketName),
        Delete: &types.Delete{Objects: objectIds, Quiet: aws.Bool(true)},
    })
    if err != nil || len(output.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucketName)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
        }
        } else if len(output.Errors) > 0 {
            for _, outErr := range output.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *output.Errors[0].Message)
        }
    } else {
        for _, delObjs := range output.Deleted {
            err = s3.NewObjectNotExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key: delObjs.Key},
                time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                    *delObjs.Key)
            } else {
                log.Printf("Deleted %s.\n", *delObjs.Key)
            }
        }
    }
}
```



```
    }
    }
}
return err
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is
// returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
    return err
}
```

S3 バケットとオブジェクトの操作方法を示すインタラクティブなシナリオを実行します。

```
import (
    "context"
    "fmt"
```

```
"log"
"os"
"strings"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunGetStartedScenario is an interactive example that shows you how to use
// Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
// objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Download an object to a local file.
// 4. Copy an object to a different folder in the bucket.
// 5. List objects in the bucket.
// 6. Delete all objects in the bucket.
// 7. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../..demotools folder of this repo.
func RunGetStartedScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := questioner.(*demotools.MockQuestioner)
            if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
                log.Println(r)
            }
        }
    }()

    log.Println(strings.Repeat("-", 88))
}
```



```
err = bucketBasics.UploadFile(ctx, bucketName, smallKey, smallFile)
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
demotools.NotEmpty{})
err = bucketBasics.DownloadFile(ctx, bucketName, smallKey, downloadFileName)
if err != nil {
    panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(ctx, bucketName, smallKey, folderName)
if err != nil {
    panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(ctx, bucketName)
if err != nil {
    panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {
    objKeys = append(objKeys, *object.Key)
    log.Printf("\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
"contents? (y/n)", "y") {
    log.Println("Deleting objects.")
    err = bucketBasics.DeleteObjects(ctx, bucketName, objKeys)
```

```
if err != nil {
    panic(err)
}
log.Println("Deleting bucket.")
err = bucketBasics.DeleteBucket(ctx, bucketName)
if err != nil {
    panic(err)
}
log.Printf("Deleting downloaded file %v.\n", downloadFileName)
err = os.Remove(downloadFileName)
if err != nil {
    panic(err)
}
} else {
    log.Println("Okay. Don't forget to delete objects from your bucket to avoid charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)

## Java

### SDK for Java 2.x

#### Note

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

シナリオ例。

```
import java.io.IOException;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Copy the object to another Amazon S3 bucket using multi copy.
 * 8. Deletes the object from the Amazon S3 bucket.
 * 9. Deletes the Amazon S3 bucket.
 */
```

```

public class S3Scenario {

    public static Scanner scanner = new Scanner(System.in);
    static S3Actions s3Actions = new S3Actions();
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    private static final Logger logger =
LoggerFactory.getLogger(S3Scenario.class);
    public static void main(String[] args) throws IOException {
        final String usage = ""
            Usage:
                <bucketName> <key> <objectPath> <savePath> <toBucket>

        Where:
            bucketName - The name of the S3 bucket.
            key - The unique identifier for the object stored in the S3
bucket.
            objectPath - The full file path of the object within the S3
bucket (e.g., "documents/reports/annual_report.pdf").
            savePath - The local file path where the object will be
downloaded and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
            toBucket - The name of the S3 bucket to which the object will be
copied.

        """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon Simple Storage Service (S3) example
scenario.");
        logger.info("
            Amazon S3 is a highly scalable and durable object storage
            service provided by Amazon Web Services (AWS). It is designed to
            store and retrieve

```

any amount of data, from anywhere on the web, at any time.

The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a set of methods to programmatically interact with the Amazon S3 (Simple Storage Service) service. This allows developers to automate the management and manipulation of S3 buckets and objects as part of their application deployment pipelines. With S3, teams can focus on building and deploying their applications without having to worry about the underlying storage infrastructure required to host and manage large amounts of data.

This scenario walks you through how to perform key operations for this service.

Let's get started...

```
""");
waitForInputToContinue(scanner);
logger.info(DASHES);

try {
    // Run the methods that belong to this scenario.
    runScenario(bucketName, key, objectPath, savePath, toBucket);

} catch (Throwable rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof S3Exception kmsEx) {
        logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
}

private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
    logger.info(DASHES);
    logger.info("1. Create an Amazon S3 bucket.");
    try {
        CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
        future.join();
    }
```



```
        waitForInputToContinue(scanner);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code  
{})", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }

    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("2. Upload a local file to the Amazon S3 bucket.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
        future.join();
        logger.info("File uploaded successfully to {}/{}", bucketName, key);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code  
{})", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }

    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("3. Download the object to another local file.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
```

```
        future.join();
        logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("4. Perform a multipart upload.");
    waitForInputToContinue(scanner);
    String multipartKey = "multiPartKey";
    try {
        // Call the multipartUpload method
        CompletableFuture<Void> future =
s3Actions.multipartUpload(bucketName, multipartKey);
        future.join();
        logger.info("Multipart upload completed successfully for bucket '{} '
and key '{}'", bucketName, multipartKey);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("5. List all objects located in the Amazon S3 bucket.");
```

```
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
            future.join();
            logger.info("Object listing completed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Copy the object to another Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
s3Actions.performMultiCopy(toBucket, bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
```

```
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("9. Delete the Amazon S3 bucket.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
s3Actions.deleteBucketAsync(bucketName);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("You successfully completed the Amazon S3 scenario.");
    logger.info(DASHES);
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        logger.info("");
    }
}
```

```
        logger.info("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            logger.info("Continuing with the program...");
            logger.info("");
            break;
        } else {
            // Handle invalid input.
            logger.info("Invalid input. Please try again.");
        }
    }
}
```

オペレーションを含むラッパークラス。

```
public class S3Actions {

    private static final Logger logger =
        LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
             * The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
             * version 2,
             * and it is designed to provide a high-performance, asynchronous HTTP
             * client for interacting with AWS services.
             * It uses the Netty framework to handle the underlying network
             * communication and the Java NIO API to
             * provide a non-blocking, event-driven approach to HTTP requests and
             * responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50) // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60)) // Set the connection
                timeout.
                .readTimeout(Duration.ofSeconds(60)) // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60)) // Set the write timeout.
            ;
            s3AsyncClient = AmazonS3AsyncClient.builder().withAsyncHttpClient(httpClient).build();
        }
    }
}
```

```

        .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2)) // Set the overall API
call timeout.
        .apiCallAttemptTimeout(Duration.ofSeconds(90)) // Set the
individual call attempt timeout.
        .retryStrategy(RetryMode.STANDARD)
        .build();

        s3AsyncClient = S3AsyncClient.builder()
        .region(Region.US_EAST_1)
        .httpClient(httpClient)
        .overrideConfiguration(overrideConfig)
        .build();
    }
    return s3AsyncClient;
}

/**
 * Creates an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes when the bucket is
created and ready
 * @throws RuntimeException if there is a failure while creating the bucket
 */
public CompletableFuture<Void> createBucketAsync(String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
    return response.thenCompose(resp -> {
        S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =

```

```

        s3Waiter.waitForBucketExists(bucketRequestWait);
        return waiterResponseFuture.thenAccept(waiterResponse -> {
            waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                logger.info(bucketName + " is ready");
            });
        });
    }).whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to create bucket", ex);
        }
    });
}

/**
 * Uploads a local file to an AWS S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (object name) to use for the uploaded file
 * @param objectPath  the local file path of the file to be uploaded
 * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
 */
public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
    return response.whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to upload file", ex);
        }
    });
}

/**

```



```

    * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
    and writes them to a local file.
    *
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path        the local file path where the object's bytes will be
    written
    * @return a {@link CompletableFuture} that completes when the object bytes
    have been written to the local file
    */
    public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
    keyName, String path) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        CompletableFuture<ResponseBytes<GetObjectResponse>> response =
        getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
        return response.thenAccept(objectBytes -> {
            try {
                byte[] data = objectBytes.asByteArray();
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
ex);
            }
        });
    }

    /**
    * Asynchronously lists all objects in the specified S3 bucket.
    *
    * @param bucketName the name of the S3 bucket to list objects for
    * @return a {@link CompletableFuture} that completes when all objects have
    been listed
    */

```

```

    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey the key (name) of the object to be copied
     * @param toBucket the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as
a {@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3
exception occurred during the copy
     */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {

```

```

        if (copyRes != null) {
            logger.info("The " + objectKey + " was copied to " + toBucket);
        } else {
            throw new RuntimeException("An S3 exception occurred during
copy", ex);
        }
    });

    return response.thenApply(CopyObjectResponse::copyObjectResult)
        .thenApply(Object::toString);
}

/**
 * Performs a multipart upload to an Amazon S3 bucket.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (name) of the file to be uploaded
 * @return a {@link CompletableFuture} that completes when the multipart
upload is successful
 */
public CompletableFuture<Void> multipartUpload(String bucketName, String key)
{
    int mB = 1024 * 1024;

    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    return
getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
        .thenCompose(createResponse -> {
            String uploadId = createResponse.uploadId();
            System.out.println("Upload ID: " + uploadId);

            // Upload part 1.
            UploadPartRequest uploadPartRequest1 =
UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(1)
                .contentType((long) (5 * mB)) // Specify the content length

```

```
        .build());

        CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(1)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Upload part 2.
        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .partNumber(2)
        .contentLength((long) (3 * mB))
        .build();

        CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(2)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Combine the results of both parts.
        return CompletableFuture.allOf(part1Future, part2Future)
        .thenCompose(v -> {
            CompletedPart part1 = part1Future.join();
            CompletedPart part2 = part2Future.join();

            CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
                .build();

            CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
                .bucket(bucketName)
```

```

        .key(key)
        .uploadId(uploadId)
        .multipartUpload(completedMultipartUpload)
        .build();

        // Complete the multipart upload
        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
    });
})
    .thenAccept(response -> System.out.println("Multipart upload
completed successfully"))
    .exceptionally(ex -> {
        System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
        throw new RuntimeException(ex);
    });
}

/**
 * Deletes an object from an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the key (file name) of the object to be deleted
 * @return a {@link CompletableFuture} that completes when the object has
been deleted
 */
public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(key + " was deleted");
        } else {
            throw new RuntimeException("An S3 exception occurred during
delete", ex);
        }
    })
}

```

```
    });

    return response.thenApply(r -> null);
}

/**
 * Deletes an S3 bucket asynchronously.
 *
 * @param bucket the name of the bucket to be deleted
 * @return a {@link CompletableFuture} that completes when the bucket
deletion is successful, or throws a {@link RuntimeException}
 * if an error occurs during the deletion process
 */
public CompletableFuture<Void> deleteBucketAsync(String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(bucket + " was deleted.");
        } else {
            throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
        }
    });
    return response.thenApply(r -> null);
}

public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(toBucket)
        .key(key)
        .build();

    getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
        .thenApply(createMultipartUploadResponse -> {
            String uploadId = createMultipartUploadResponse.uploadId();
            System.out.println("Upload ID: " + uploadId);
        });
}
```

```

        UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
    .sourceBucket(bucketName)
    .destinationBucket(toBucket)
    .sourceKey(key)
    .destinationKey(key)
    .uploadId(uploadId) // Use the valid uploadId.
    .partNumber(1) // Ensure the part number is correct.
    .copySourceRange("bytes=0-1023") // Adjust range as needed
    .build();

        return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
    })
    .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
    .whenComplete((uploadPartCopyResponse, exception) -> {
        if (exception != null) {
            // Handle any exceptions.
            logger.error("Error during upload part copy: " +
exception.getMessage());
        } else {
            // Successfully completed the upload part copy.
            System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
        }
    });
    return null;
}

private static ByteBuffer getRandomByteBuffer(int size) {
    ByteBuffer buffer = ByteBuffer.allocate(size);
    for (int i = 0; i < size; i++) {
        buffer.put((byte) (Math.random() * 256));
    }
    buffer.flip();
    return buffer;
}
}
}

```

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

- [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

まず、必要なモジュールをすべてインポートします。

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "node:url";
import { readdirSync, readFileSync, writeFileSync } from "node:fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```



前述のインポートでは、いくつかのヘルパーユーティリティを参照しています。これらのユーティリティは、このセクションの冒頭でリンクされている GitHub リポジトリのローカルにあります。参考までに、これらのユーティリティの以下の実装を参照してください。

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox, password } from "@inquirer/prompts";

export class Prompter {
  /**
   * @param {{ message: string, choices: { name: string, value: string }[] }}
   options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {{ message: string }} options
   */
  password(options) {
    return password({ ...options, mask: true });
  }

  /**
   * @param {string} prompt
   */
  checkContinue = async (prompt = "") => {
    const prefix = prompt && `${prompt} `;
    const ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
  };
}
```

```

/**
 * @param {{ message: string }} options
 */
confirm(options) {
  return confirm(options);
}

/**
 * @param {{ message: string, choices: { name: string, value: string }[] }}
options
 */
checkbox(options) {
  return checkbox(options);
}
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n  ${text}\n${rule}\n`;
};

```

オブジェクトは「バケット」に保存されます。新しいバケットを作成する関数を定義しましょう。

```

export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};

```

バケットには「オブジェクト」が含まれています。この関数は、ディレクトリの内容をバケットにオブジェクトとしてアップロードします。

```

export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {

```

```
const filePath = `${folderPath}/${key}`;
const fileContent = readFileSync(filePath);
return {
  Key: key,
  Body: fileContent,
};
});

for (const file of files) {
  await s3Client.send(
    new PutObjectCommand({
      Bucket: bucketName,
      Body: file.Body,
      Key: file.Key,
    }),
  );
  console.log(`${file.Key} uploaded successfully.`);
}
};
```

オブジェクトをアップロードしたら、正しくアップロードされたことを確認します。そのためには `ListObjects` を使用できます。ここでは 'Key' プロパティを使用しますが、レスポンスには他にも便利なプロパティがあります。

```
export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(`${contentsList}\n`);
};
```

バケットから別のバケットにオブジェクトをコピーしたい場合があります。そのためには、`CopyObject` コマンドを使用してください。

```
export const copyFileFromBucket = async ({ destinationBucket }) => {
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });
};
```

```
if (!proceed) {
  return;
}
const copy = async () => {
  try {
    const sourceBucket = await prompter.input({
      message: "Enter source bucket name:",
    });
    const sourceKey = await prompter.input({
      message: "Enter source key:",
    });
    const destinationKey = await prompter.input({
      message: "Enter destination key:",
    });

    const command = new CopyObjectCommand({
      Bucket: destinationBucket,
      CopySource: `${sourceBucket}/${sourceKey}`,
      Key: destinationKey,
    });
    await s3Client.send(command);
    await copyFileFromBucket({ destinationBucket });
  } catch (err) {
    console.error("Copy error.");
    console.error(err);
    const retryAnswer = await prompter.confirm({ message: "Try again?" });
    if (retryAnswer) {
      await copy();
    }
  }
};
await copy();
};
```

バケットから複数のオブジェクトを取得するための SDK メソッドはありません。代わりに、ダウンロードして繰り返し処理するオブジェクトのリストを作成します。

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
```

```
    message: "Enter destination path for files:",
  });

  for (const content of Contents) {
    const obj = await s3Client.send(
      new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
    );
    writeFileSync(
      `${path}/${content.Key}`,
      await obj.Body.transformToByteArray(),
    );
  }
  console.log("Files downloaded successfully.\n");
};
```

では、リソースをクリーンアップしましょう。バケットを削除するには、そのバケットを空にしておく必要があります。これら 2 つの関数はバケットを空にして削除します。

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};
```

「main」関数はすべてをまとめます。このファイルを直接実行すると、main 関数が呼び出されます。

```
const main = async () => {
```

```
const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
  import.meta.url,
)}../../../../resources/sample_files/.sample_media`;

try {
  console.log(wrapText("Welcome to the Amazon S3 getting started example.));
  console.log("Let's create a bucket.");
  const bucketName = await createBucket();
  await prompter.confirm({ message: continueMessage });

  console.log(wrapText("File upload.));
  console.log(
    "I have some default files ready to go. You can edit the source code to
provide your own.",
  );
  await uploadFilesToBucket({
    bucketName,
    folderPath: OBJECT_DIRECTORY,
  });

  await listFilesInBucket({ bucketName });
  await prompter.confirm({ message: continueMessage });

  console.log(wrapText("Copy files.));
  await copyFileFromBucket({ destinationBucket: bucketName });
  await listFilesInBucket({ bucketName });
  await prompter.confirm({ message: continueMessage });

  console.log(wrapText("Download files.));
  await downloadFilesFromBucket({ bucketName });

  console.log(wrapText("Clean up.));
  await emptyBucket({ bucketName });
  await deleteBucket({ bucketName });
} catch (err) {
  console.error(err);
}
};
```

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

- [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

## Kotlin

### SDK for Kotlin

#### Note

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

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for
example, C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
```

```
val key = args[1]
val objectPath = args[2]
val savePath = args[3]
val toBucket = args[4]

// Create an Amazon S3 bucket.
createBucket(bucketName)

// Update a local file to the Amazon S3 bucket.
putObject(bucketName, key, objectPath)

// Download the object to another local file.
getObjectFromMap(bucketName, key, savePath)

// List all objects located in the Amazon S3 bucket.
listBucketObs(bucketName)

// Copy the object to another Amazon S3 bucket
copyBucketOb(bucketName, key, toBucket)

// Delete the object from the Amazon S3 bucket.
deleteBucketObs(bucketName, key)

// Delete the Amazon S3 bucket.
deleteBucket(bucketName)
println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
```



```
) {  
    val metadataVal = mutableMapOf<String, String>()  
    metadataVal["myVal"] = "test"  
  
    val request =  
        PutObjectRequest {  
            bucket = bucketName  
            key = objectKey  
            metadata = metadataVal  
            this.body = Paths.get(objectPath).asByteStream()  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        val response = s3.putObject(request)  
        println("Tag information is ${response.eTag}")  
    }  
}  
  
suspend fun getObjectFromMrap(  
    bucketName: String,  
    keyName: String,  
    path: String,  
) {  
    val request =  
        GetObjectRequest {  
            key = keyName  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.getObject(request) { resp ->  
            val myFile = File(path)  
            resp.body?.writeToFile(myFile)  
            println("Successfully read $keyName from $bucketName")  
        }  
    }  
}  
  
suspend fun listBucketObs(bucketName: String) {  
    val request =  
        ListObjectsRequest {  
            bucket = bucketName  
        }  
}
```

```
S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->

    val response = s3.listObjects(request)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
```

```
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)

## PHP

## SDK for PHP

 Note

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

```
echo("\n");
echo("-----\n");
print("Welcome to the Amazon S3 getting started demo using PHP!\n");
echo("-----\n");

$region = 'us-west-2';

$this->s3client = new S3Client([
    'region' => $region,
]);
/* Inline declaration example
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
*/

$this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $this->bucketName \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}

$fileName = __DIR__ . "/local-file-" . uniqid();
try {
    $this->s3client->putObject([
        'Bucket' => $this->bucketName,
```

```
        'Key' => $fileName,
        'SourceFile' => __DIR__ . '/testfile.txt'
    ]);
    echo "Uploaded $fileName to $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with file upload before continuing.");
}

try {
    $file = $this->s3client->getObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with object copying before continuing.");
}

try {
    $contents = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
```

```
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with listing objects before continuing.");
}

try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $this->s3client->deleteObjects([
        'Bucket' => $this->bucketName,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    if (isset($check['Contents']) && count($check['Contents']) > 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $fileName from $this->bucketName with error:
    " . $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}

try {
    $this->s3client->deleteBucket([
        'Bucket' => $this->bucketName,
    ]);
    echo "Deleted bucket $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $this->bucketName with error: " . $exception-
    >getMessage();
    exit("Please fix error with bucket deletion before continuing.");
}
```

```
echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)

## Python

### SDK for Python (Boto3)

#### Note

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

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError

def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)
```

```

bucket_name = f"amzn-s3-demo-bucket-{uuid.uuid4()}"
bucket = s3_resource.Bucket(bucket_name)
try:
    bucket.create(
        CreateBucketConfiguration={
            "LocationConstraint": s3_resource.meta.client.meta.region_name
        }
    )
    print(f"Created demo bucket named {bucket.name}.")
except ClientError as err:
    print(f"Tried and failed to create demo bucket {bucket_name}.")
    print(f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}")
    print(f"\nCan't continue the demo without a bucket!")
    return

file_name = None
while file_name is None:
    file_name = input("\nEnter a file you want to upload to your bucket: ")
    if not os.path.exists(file_name):
        print(f"Couldn't find file {file_name}. Are you sure it exists?")
        file_name = None

obj = bucket.Object(os.path.basename(file_name))
try:
    obj.upload_file(file_name)
    print(
        f"Uploaded file {file_name} into bucket {bucket.name} with key
{obj.key}."
    )
except S3UploadFailedError as err:
    print(f"Couldn't upload file {file_name} to {bucket.name}.")
    print(f"\t{err}")

answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
if answer.lower() == "y":
    data = io.BytesIO()
    try:
        obj.download_fileobj(data)
        data.seek(0)
        print(f"Got your object. Here are the first 20 bytes:\n")
        print(f"\t{data.read(20)}")
    except ClientError as err:

```



```

        print(f"Couldn't download {obj.key}.")
        print(
            f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
        )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)?
"
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

```

```
)

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

if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)

## Ruby

### SDK for Ruby

#### Note

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

```
require 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
```

```
@s3_resource = s3_resource
end

# Creates a bucket with a random name in the currently configured account and
# AWS Region.
#
# @return [Aws::S3::Bucket] The newly created bucket.
def create_bucket
  bucket = @s3_resource.create_bucket(
    bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
    create_bucket_configuration: {
      location_constraint: 'us-east-1' # NOTE: only certain regions permitted
    }
  )
  puts("Created demo bucket named #{bucket.name}.")
rescue Aws::Errors::ServiceError => e
  puts('Tried and failed to create demo bucket.')
  puts("\t#{e.code}: #{e.message}")
  puts("\nCan't continue the demo without a bucket!")
  raise
else
  bucket
end

# Requests a file name from the user.
#
# @return The name of the file.
def create_file
  File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
end

# Uploads a file to an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
# @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
file.
def upload_file(bucket)
  File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
  s3_object = bucket.object(File.basename('demo.txt'))
  s3_object.upload_file('demo.txt')
  puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
rescue Aws::Errors::ServiceError => e
```

```

    puts("Couldn't upload file demo.txt to #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    s3_object
end

# Downloads an Amazon S3 object to a file.
#
# @param s3_object [Aws::S3::Object] The object to download.
def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
        puts('Enter a name for the downloaded file: ')
        file_name = gets.chomp
        s3_object.download_file(file_name)
        puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
rescue Aws::Errors::ServiceError => e
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
end

# Copies an Amazon S3 object to a subfolder within the same bucket.
#
# @param source_object [Aws::S3::Object] The source object to copy.
# @return [Aws::S3::Object, nil] The destination object.
def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your
bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
        dest_object = source_object.bucket.object("demo-folder/
#{source_object.key}")
        dest_object.copy_from(source_object)
        puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
end

```

```
else
  dest_object
end

# Lists the objects in an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to query.
def list_objects(bucket)
  puts("\nYour bucket contains the following objects:")
  bucket.objects.each do |obj|
    puts("\t#{obj.key}")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list the objects in bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
```

```
s3_object = scenario.upload_file(bucket)
scenario.download_file(s3_object)
scenario.copy_object(s3_object)
scenario.list_objects(bucket)
scenario.delete_bucket(bucket)

puts('Thanks for watching!')
puts('-' * 88)
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo!')
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME
== __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)

## Rust

### SDK for Rust

#### Note

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

シナリオを実行するバイナリクレートのコード。

```
#![allow(clippy::result_large_err)]

//! Purpose
//! Shows how to use the AWS SDK for Rust to get started using
//! Amazon Simple Storage Service (Amazon S3). Create a bucket, move objects
    into and out of it,
//! and delete all resources at the end of the demo.
//!
//! This example follows the steps in "Getting started with Amazon S3" in the
    Amazon S3
//! user guide.
//! - https://docs.aws.amazon.com/AmazonS3/latest/userguide/GetStartedWithS3.html

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_code_examples::error::S3ExampleError;
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    let shared_config =
aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);
    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name,
key, target_key).await
    {
        eprintln!("{:?}", e);
    };

    Ok(())
}

async fn run_s3_operations(
    region: Region,
    client: Client,
```

```

    bucket_name: String,
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), S3ExampleError> {
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;
    let run_example: Result<(), S3ExampleError> = (async {
        s3_code_examples::upload_object(&client, &bucket_name, &file_name,
&key).await?;
        let _object = s3_code_examples::download_object(&client, &bucket_name,
&key).await;
        s3_code_examples::copy_object(&client, &bucket_name, &bucket_name, &key,
&target_key)
            .await?;
        s3_code_examples::list_objects(&client, &bucket_name).await?;
        s3_code_examples::clear_bucket(&client, &bucket_name).await?;
        Ok(())
    })
    .await;
    if let Err(err) = run_example {
        eprintln!("Failed to complete getting-started example: {err:?}");
    }
    s3_code_examples::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}

```

シナリオで使われる一般的なアクション。

```

pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
S3ExampleError> {
    let constraint =
aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()

```



```
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
    task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}

pub async fn download_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::get_object::GetObjectOutput, S3ExampleError> {
```

```
        client
            .get_object()
            .bucket(bucket_name)
            .key(key)
            .send()
            .await
            .map_err(S3ExampleError::from)
    }

    /// Copy an object from one bucket to another.
    pub async fn copy_object(
        client: &aws_sdk_s3::Client,
        source_bucket: &str,
        destination_bucket: &str,
        source_object: &str,
        destination_object: &str,
    ) -> Result<(), S3ExampleError> {
        let source_key = format!("{source_bucket}/{source_object}");
        let response = client
            .copy_object()
            .copy_source(&source_key)
            .bucket(destination_bucket)
            .key(destination_object)
            .send()
            .await?;

        println!(
            "Copied from {source_key} to {destination_bucket}/{destination_object}
            with etag {}",
            response
                .copy_object_result
                .unwrap_or_else(||
                    aws_sdk_s3::types::CopyObjectResult::builder().build())
                .e_tag()
                .unwrap_or("missing")
        );
        Ok(())
    }

    pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
        Result<(), S3ExampleError> {
        let mut response = client
            .list_objects_v2()
            .bucket(bucket.to_owned())
```

```
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

while let Some(result) = response.next().await {
    match result {
        Ok(output) => {
            for object in output.contents() {
                println!(" - {}", object.key().unwrap_or("Unknown"));
            }
        }
        Err(err) => {
            eprintln!("{err:?}")
        }
    }
}

Ok(())
}

/// Given a bucket, remove all objects in the bucket, and then ensure no objects
/// remain in the bucket.
pub async fn clear_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<Vec<String>, S3ExampleError> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    // delete_objects no longer needs to be mutable.
    let objects_to_delete: Vec<String> = objects
        .contents()
        .iter()
        .filter_map(|obj| obj.key())
        .map(String::from)
        .collect();

    if objects_to_delete.is_empty() {
        return Ok(vec![]);
    }

    let return_keys = objects_to_delete.clone();

    delete_objects(client, bucket_name, objects_to_delete).await?;
```

```
let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

eprintln!("{objects:?}");

match objects.key_count {
    Some(0) => Ok(return_keys),
    _ => Err(S3ExampleError::new(
        "There were still objects left in the bucket.",
    )),
}
}

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
```

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

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)

- [ListObjectsV2](#)
- [PutObject](#)

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    " determine our region from our session
    DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
    DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
    " When in the us-east-1 region, you must not specify a constraint
    " In all other regions, specify the region as the constraint
    IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
    ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
    ENDIF.

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
CATCH /aws1/cx_s3_bucketalrdyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
CATCH /aws1/cx_s3_bktalrdyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.
```

```
"Upload an object to an S3 bucket."
TRY.
    "Get contents of file from application server."
    DATA lv_file_content TYPE xstring.
    OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
    READ DATASET iv_key INTO lv_file_content.
    CLOSE DATASET iv_key.

    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_file_content ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Get an object from a bucket. "
TRY.
    DATA(lo_result) = lo_s3->getobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key ).
    DATA(lv_object_data) = lo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" Copy an object to a subfolder in a bucket. "
TRY.
    lo_s3->copyobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
        iv_copysource = |{ iv_bucket_name }/{ iv_key }| ).
    MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" List objects in the bucket. "
```

```

TRY.
    DATA(lo_list) = lo_s3->listobjects(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

DATA text TYPE string VALUE 'Object List - '.
DATA lv_object_key TYPE /aws1/s3_objectkey.
LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
    lv_object_key = lo_object->get_key( ).
    CONCATENATE lv_object_key ', ' INTO text.
ENDLOOP.
MESSAGE text TYPE 'I'.

" Delete the objects in a bucket. "
TRY.
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key ).
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }| ).
    MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Delete the bucket. "
TRY.
    lo_s3->deletebucket(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

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

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

## Swift

### SDK for Swift

#### Note

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

```
import AWSS3

import Foundation
import AWSS3
import Smithy
import ClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let configuration: S3Client.S3ClientConfiguration
    let client: S3Client

    enum HandlerError: Error {
        case getObjectBody(String)
        case readGetObjectBody(String)
        case missingContents(String)
    }

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
    drive the AWS calls
    /// used for the example.
    ///
```



```
/// - Returns: A new ``ServiceHandler`` object, ready to be called to
///           execute AWS operations.
public init() async throws {
    do {
        configuration = try await S3Client.S3ClientConfiguration()
        // configuration.region = "us-east-2" // Uncomment this to set the
region programmatically.
        client = S3Client(config: configuration)
    }
    catch {
        print("ERROR: ", dump(error, name: "Initializing S3 client"))
        throw error
    }
}

/// Create a new user given the specified name.
///
/// - Parameters:
///   - name: Name of the bucket to create.
/// Throws an exception if an error occurs.
public func createBucket(name: String) async throws {
    var input = CreateBucketInput(
        bucket: name
    )

    // For regions other than "us-east-1", you must set the
locationConstraint in the createBucketConfiguration.
    // For more information, see LocationConstraint in the S3 API guide.
    // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
    if let region = configuration.region {
        if region != "us-east-1" {
            input.createBucketConfiguration =
S3ClientTypes.CreateBucketConfiguration(locationConstraint:
S3ClientTypes.BucketLocationConstraint(rawValue: region))
        }
    }

    do {
        _ = try await client.createBucket(input: input)
    }
    catch let error as BucketAlreadyOwnedByYou {
```

```
        print("The bucket '\(name)' already exists and is owned by you. You
may wish to ignore this exception.")
        throw error
    }
    catch {
        print("ERROR: ", dump(error, name: "Creating a bucket"))
        throw error
    }
}

/// Delete a bucket.
/// - Parameter name: Name of the bucket to delete.
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    do {
        _ = try await client.deleteBucket(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a bucket"))
        throw error
    }
}

/// Upload a file from local storage to the bucket.
/// - Parameters:
///   - bucket: Name of the bucket to upload the file to.
///   - key: Name of the file to create.
///   - file: Path name of the file to upload.
public func uploadFile(bucket: String, key: String, file: String) async
throws {
    let fileUrl = URL(fileURLWithPath: file)
    do {
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        _ = try await client.putObject(input: input)
```

```
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object. "))
        throw error
    }
}

/// Create a file in the specified bucket with the given name. The new
/// file's contents are uploaded from a `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket to create a file in.
///   - key: Name of the file to create.
///   - data: A `Data` object to write into the new file.
public func createFile(bucket: String, key: String, withData data: Data)
async throws {
    let dataStream = ByteStream.data(data)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object. "))
        throw error
    }
}

/// Download the named file to the given directory on the local device.
///
/// - Parameters:
///   - bucket: Name of the bucket that contains the file to be copied.
///   - key: The name of the file to copy from the bucket.
///   - to: The path of the directory on the local device where you want to
///     download the file.
public func downloadFile(bucket: String, key: String, to: String) async
throws {
    let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)
```

```
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file. "))
            throw error
        }
    }
}

/// Read the specified file from the given S3 bucket into a Swift
/// `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket containing the file to read.
///   - key: Name of the file within the bucket to read.
///
/// - Returns: A `Data` object containing the complete file data.
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }
    }
```

```
        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        return data
    }
    catch {
        print("ERROR: ", dump(error, name: "Reading a file. "))
        throw error
    }
}

/// Copy a file from one bucket to another.
///
/// - Parameters:
///   - sourceBucket: Name of the bucket containing the source file.
///   - name: Name of the source file.
///   - destBucket: Name of the bucket to copy the file into.
public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
    let srcUrl = ("\(sourceBucket)/
\((name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

    let input = CopyObjectInput(
        bucket: destBucket,
        copySource: srcUrl,
        key: name
    )
    do {
        _ = try await client.copyObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Copying an object. "))
        throw error
    }
}

/// Deletes the specified file from Amazon S3.
///
/// - Parameters:
///   - bucket: Name of the bucket containing the file to delete.
///   - key: Name of the file to delete.
```

```
///
public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a file. "))
        throw error
    }
}

/// Returns an array of strings, each naming one file in the
/// specified bucket.
///
/// - Parameter bucket: Name of the bucket to get a file listing for.
/// - Returns: An array of `String` objects, each giving the name of
///            one file contained in the bucket.
public func listBucketFiles(bucket: String) async throws -> [String] {
    do {
        let input = ListObjectsV2Input(
            bucket: bucket
        )

        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'continuationToken' in
        "ListObjectsV2Output".
        let output = client.listObjectsV2Paginated(input: input)
        var names: [String] = []

        for try await page in output {
            guard let objList = page.contents else {
                print("ERROR: listObjectsV2Paginated returned nil contents.")
                continue
            }

            for obj in objList {
                if let objName = obj.key {
                    names.append(objName)
                }
            }
        }
    }
}
```

```
        }
    }

    return names
}
catch {
    print("ERROR: ", dump(error, name: "Listing objects. "))
    throw error
}
}
```

```
import AWSS3

import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
bucket")
    var downloadDir: String

    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: ""
    )
}
```

```
    Performs the following Amazon S3 commands:

    * `CreateBucket`
    * `PutObject`
    * `GetObject`
    * `CopyObject`
    * `ListObjects`
    * `DeleteObjects`
    * `DeleteBucket`
    """"
)

/// Called by ``main()`` to do the actual running of the AWS
/// example.
func runAsync() async throws {
    let serviceHandler = try await ServiceHandler()

    // 1. Create the bucket.
    print("Creating the bucket \(bucketName)...")
    try await serviceHandler.createBucket(name: bucketName)

    // 2. Upload a file to the bucket.
    print("Uploading the file \(uploadSource)...")
    try await serviceHandler.uploadFile(bucket: bucketName, key: objName,
file: uploadSource)

    // 3. Download the file.
    print("Downloading the file \(objName) to \(downloadDir)...")
    try await serviceHandler.downloadFile(bucket: bucketName, key: objName,
to: downloadDir)

    // 4. Copy the file to another bucket.
    print("Copying the file to the bucket \(destBucket)...")
    try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

    // 5. List the contents of the bucket.

    print("Getting a list of the files in the bucket \(bucketName)")
    let fileList = try await serviceHandler.listBucketFiles(bucket:
bucketName)
    let numFiles = fileList.count
    if numFiles != 0 {
```



```
        print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
        \(bucketName):")
        for name in fileList {
            print("  \(name)")
        }
    } else {
        print("No files found in bucket \(bucketName)")
    }

    // 6. Delete the objects from the bucket.

    print("Deleting the file \(objName) from the bucket \(bucketName)...")
    try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
    print("Deleting the file \(objName) from the bucket \(destBucket)...")
    try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

    // 7. Delete the bucket.
    print("Deleting the bucket \(bucketName)...")
    try await serviceHandler.deleteBucket(name: bucketName)

    print("Done.")
}
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- API の詳細については、AWS SDK for Swift API リファレンスの以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjectsV2](#)
  - [PutObject](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用した Amazon S3 のアクション

次のコード例では、AWS SDK を使用して個々の Amazon S3 アクションを実行する方法を示しています。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

これらの抜粋は、Amazon S3 API を呼び出し、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。アクションは [AWS SDK を使用した Amazon S3 のシナリオ](#) のコンテキスト内で確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。完全なリストについては、「[Amazon Simple Storage Service API リファレンス](#)」を参照してください。

### 例

- [AWS SDK または CLI で AbortMultipartUpload を使用する](#)
- [AWS SDK または CLI で CompleteMultipartUpload を使用する](#)
- [AWS SDK または CLI で CopyObject を使用する](#)
- [AWS SDK または CLI で CreateBucket を使用する](#)
- [AWS SDK で CreateMultiRegionAccessPoint を使用する](#)
- [AWS SDK または CLI で CreateMultipartUpload を使用する](#)
- [AWS SDK で CreatePresignedPost を使用する](#)

- [AWS SDK または CLI で DeleteBucket を使用する](#)
- [CLI で DeleteBucketAnalyticsConfiguration を使用する](#)
- [AWS SDK または CLI で DeleteBucketCors を使用する](#)
- [CLI で DeleteBucketEncryption を使用する](#)
- [CLI で DeleteBucketInventoryConfiguration を使用する](#)
- [AWS SDK または CLI で DeleteBucketLifecycle を使用する](#)
- [CLI で DeleteBucketMetricsConfiguration を使用する](#)
- [AWS SDK または CLI で DeleteBucketPolicy を使用する](#)
- [CLI で DeleteBucketReplication を使用する](#)
- [CLI で DeleteBucketTagging を使用する](#)
- [AWS SDK または CLI で DeleteBucketWebsite を使用する](#)
- [AWS SDK または CLI で DeleteObject を使用する](#)
- [CLI で DeleteObjectTagging を使用する](#)
- [AWS SDK または CLI で DeleteObjects を使用する](#)
- [CLI で DeletePublicAccessBlock を使用する](#)
- [CLI で GetBucketAccelerateConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketAcl を使用する](#)
- [CLI で GetBucketAnalyticsConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketCors を使用する](#)
- [AWS SDK または CLI で GetBucketEncryption を使用する](#)
- [CLI で GetBucketInventoryConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketLifecycleConfiguration を使用する](#)
- [AWS SDK または CLI で GetBucketLocation を使用する](#)
- [CLI で GetBucketLogging を使用する](#)
- [CLI で GetBucketMetricsConfiguration を使用する](#)
- [CLI で GetBucketNotification を使用する](#)
- [AWS SDK または CLI で GetBucketPolicy を使用する](#)
- [CLI で GetBucketPolicyStatus を使用する](#)
- [AWS SDK または CLI で GetBucketReplication を使用する](#)
- [CLI で GetBucketRequestPayment を使用する](#)

- [CLI で GetBucketTagging を使用する](#)
- [CLI で GetBucketVersioning を使用する](#)
- [AWS SDK または CLI で GetBucketWebsite を使用する](#)
- [AWS SDK または CLI で GetObject を使用する](#)
- [AWS SDK または CLI で GetObjectAcl を使用する](#)
- [AWS SDK または CLI で GetObjectAttributes を使用する](#)
- [AWS SDK または CLI で GetObjectLegalHold を使用する](#)
- [AWS SDK または CLI で GetObjectLockConfiguration を使用する](#)
- [AWS SDK または CLI で GetObjectRetention を使用する](#)
- [CLI で GetObjectTagging を使用する](#)
- [CLI で GetPublicAccessBlock を使用する](#)
- [AWS SDK または CLI で HeadBucket を使用する](#)
- [AWS SDK または CLI で HeadObject を使用する](#)
- [CLI で ListBucketAnalyticsConfigurations を使用する](#)
- [CLI で ListBucketInventoryConfigurations を使用する](#)
- [AWS SDK または CLI で ListBuckets を使用する](#)
- [AWS SDK または CLI で ListMultipartUploads を使用する](#)
- [AWS SDK または CLI で ListObjectVersions を使用する](#)
- [CLI で ListObjects を使用する](#)
- [AWS SDK または CLI で ListObjectsV2 を使用する](#)
- [AWS SDK または CLI で PutBucketAccelerateConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketAcl を使用する](#)
- [AWS SDK または CLI で PutBucketCors を使用する](#)
- [AWS SDK または CLI で PutBucketEncryption を使用する](#)
- [AWS SDK または CLI で PutBucketLifecycleConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketLogging を使用する](#)
- [CLI で PutBucketNotification を使用する](#)
- [AWS SDK または CLI で PutBucketNotificationConfiguration を使用する](#)
- [AWS SDK または CLI で PutBucketPolicy を使用する](#)
- [AWS SDK または CLI で PutBucketReplication を使用する](#)

- [CLI で PutBucketRequestPayment を使用する](#)
- [CLI で PutBucketTagging を使用する](#)
- [AWS SDK または CLI で PutBucketVersioning を使用する](#)
- [AWS SDK または CLI で PutBucketWebsite を使用する](#)
- [AWS SDK または CLI で PutObject を使用する](#)
- [AWS SDK または CLI で PutObjectAcl を使用する](#)
- [AWS SDK または CLI で PutObjectLegalHold を使用する](#)
- [AWS SDK または CLI で PutObjectLockConfiguration を使用する](#)
- [AWS SDK または CLI で PutObjectRetention を使用する](#)
- [AWS SDK または CLI で RestoreObject を使用する](#)
- [AWS SDK または CLI で SelectObjectContent を使用する](#)
- [AWS SDK または CLI で UploadPart を使用する](#)
- [AWS SDK または CLI で UploadPartCopy を使用する](#)

AWS SDK または CLI で **AbortMultipartUpload** を使用する

次のサンプルコードは、AbortMultipartUpload を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [不完全なマルチパートアップロードを特定する](#)
- [Amazon S3 オブジェクトの整合性の操作](#)

C++

SDK for C++

 Note

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

```
///  
//! Abort a multipart upload to an S3 bucket.
```

```

/ * !
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
                                       const Aws::String &uploadID,
                                       const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
        client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

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

## CLI

### AWS CLI

指定したマルチパートアップロードを中止するには

次の abort-multipart-upload コマンドは、バケット amzn-s3-demo-bucket 内のキー multipart/01 のマルチパートアップロードを中止します。

```
aws s3api abort-multipart-upload \  
  --bucket amzn-s3-demo-bucket \  
  --key multipart/01 \  
  --upload-  
id dfRtDYU0WMCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZlJF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P
```

このコマンドに必要なアップロード ID create-multipart-upload はによって出力され、list-multipart-uploads で取得することもできます。

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

## Java

### SDK for Java 2.x

#### Note

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

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.core.exception.SdkException;  
import software.amazon.awssdk.core.sync.RequestBody;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;  
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;  
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;  
import software.amazon.awssdk.services.s3.model.CompletedPart;  
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;  
import software.amazon.awssdk.services.s3.model.LifecycleRule;  
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;  
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;  
import software.amazon.awssdk.services.s3.model.MultipartUpload;  
import  
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
```

```
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code 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 AbortMultipartUploadExamples {
    static final S3Client s3Client = S3Client.create();
    static final String classPathFilePath = "/multipartUploadFiles/s3-userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    private static final Logger logger =
        LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.
        String key = UUID.randomUUID().toString();
    }
}
```



```

        createBucket(bucketName);
    try {
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread");
        abortIncompleteMultipartUploadsFromList(bucketName);
        abortMultipartUploadUsingUploadId(bucketName, key);
        abortMultipartUploadsUsingLifecycleConfig(bucketName);
    } catch (S3Exception e) {
        logger.error(e.getMessage());
    } finally {
        deleteResources(bucketName, key);
    }
}

/**
 * Aborts all incomplete multipart uploads from the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket
 */
public static void abortIncompleteMultipartUploadsFromList(String bucketName)
{
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(upload.key())
            .expectedBucketOwner(accountId)
            .uploadId(upload.uploadId())
            .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}

```

```
    }  
}  
  
/**  
 * Aborts incomplete multipart uploads older than the specified point in  
time.  
 *  
 * @param bucketName the name of the S3 bucket  
 * @param pointInTime the cutoff time; uploads initiated before this are  
aborted  
 */  
static void abortIncompleteMultipartUploadsOlderThan(String bucketName,  
Instant pointInTime) {  
    ListMultipartUploadsRequest listMultipartUploadsRequest =  
ListMultipartUploadsRequest.builder()  
        .bucket(bucketName)  
        .build();  
  
    ListMultipartUploadsResponse response =  
s3Client.listMultipartUploads(listMultipartUploadsRequest);  
    List<MultipartUpload> uploads = response.uploads();  
  
    AbortMultipartUploadRequest abortMultipartUploadRequest;  
    for (MultipartUpload upload : uploads) {  
        logger.info("Found multipartUpload with upload ID [{}], initiated  
[{}]", upload.uploadId(), upload.initiated());  
        if (upload.initiated().isBefore(pointInTime)) {  
            abortMultipartUploadRequest =  
AbortMultipartUploadRequest.builder()  
                .bucket(bucketName)  
                .key(upload.key())  
                .expectedBucketOwner(accountId)  
                .uploadId(upload.uploadId())  
                .build();  
  
            AbortMultipartUploadResponse abortMultipartUploadResponse =  
s3Client.abortMultipartUpload(abortMultipartUploadRequest);  
            if  
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {  
                logger.info("Upload ID [{}] to bucket [{}] successfully  
aborted.", upload.uploadId(), bucketName);  
            }  
        }  
    }  
}
```

```

    }

    /**
     * Aborts a multipart upload using the upload ID.
     *
     * @param bucketName the name of the S3 bucket
     * @param key         the object key
     */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key)
    {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
            .uploadId(uploadId)
            .bucket(bucketName)
            .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
        }
    }

    /**
     * Configures a lifecycle rule to abort incomplete multipart uploads after 7
    days.
     *
     * @param bucketName the name of the S3 bucket
     */
    static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
        Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
            .abortIncompleteMultipartUpload(b -> b.
                daysAfterInitiation(7))
            .status("Enabled")
            .filter(SdkBuilder::build) // Filter element is required.
            .build());

        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
            .bucket(bucketName)
            .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {

```

```
        logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
    } else {
        logger.error("Unsuccessfully applied rule. HTTP status code is [{ }]",
response.sdkHttpResponse().statusCode());
    }
}

/*****
Multipart upload methods
*****/

static void initiateAndInterruptMultiPartUpload(String bucketName, String
key, String threadName) {
    Runnable upload = () -> {
        try {
            doMultipartUpload(bucketName, key);
        } catch (SdkException e) {
            logger.error(e.getMessage());
        }
    };
    Thread uploadThread = new Thread(upload, threadName);
    uploadThread.start();
    try {
        Thread.sleep(Duration.ofSeconds(1).toMillis());
    } catch (InterruptedException e) {
        logger.error(e.getMessage());
    }
    uploadThread.interrupt();
}

static Instant initiateAndInterruptTwoUploads(String bucketName, String key)
{
    initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread1");
    try {
        Thread.sleep(Duration.ofSeconds(5).toMillis());
    } catch (InterruptedException e) {
        logger.error(e.getMessage());
    }
    Instant secondUploadInstant = Instant.now();
    initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread2");
    return secondUploadInstant;
}
```

```
static void doMultipartUpload(String bucketName, String key) {
    String uploadId = step1CreateMultipartUpload(bucketName, key);
    List<CompletedPart> completedParts = step2UploadParts(bucketName, key,
uploadId);
    step3CompleteMultipartUpload(bucketName, key, uploadId, completedParts);
}

static String step1CreateMultipartUpload(String bucketName, String key) {
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    return createMultipartUploadResponse.uploadId();
}

static List<CompletedPart> step2UploadParts(String bucketName, String key,
String uploadId) {
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip();
            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
                RequestBody.fromByteBuffer(bb));

            CompletedPart part = CompletedPart.builder()
                .partNumber(partNumber)
                .eTag(partResponse.eTag())
                .build();
        }
    }
}
```

```
        completedParts.add(part);
        logger.info("Part {} upload", partNumber);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException | S3Exception e) {
    logger.error(e.getMessage());
    return null;
}
return completedParts;
}

static void step3CompleteMultipartUpload(String bucketName, String key,
String uploadId, List<CompletedPart> completedParts) {
    s3Client.completeMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)

        .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}

static String startUploadReturningUploadId(String bucketName, String key) {
    String uploadId = step1CreateMultipartUpload(bucketName, key);
    doMultipartUploadWithUploadId(bucketName, key, uploadId);
    return uploadId;
}

static void doMultipartUploadWithUploadId(String bucketName, String key,
String uploadId) {
    new Thread(() -> {
        try {
            List<CompletedPart> completedParts = step2UploadParts(bucketName,
key, uploadId);
            step3CompleteMultipartUpload(bucketName, key, uploadId,
completedParts);
        } catch (SdkException e) {
            logger.error(e.getMessage());
        }
    }, "upload thread").start();
    try {
        Thread.sleep(Duration.ofSeconds(2L).toMillis());
    }
```

```
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Resource handling methods
     */

    static void createBucket(String bucketName) {
        logger.info("Creating bucket: [{}]", bucketName);
        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

    static void deleteResources(String bucketName, String key) {
        logger.info("Deleting resources ...");
        s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
        s3Client.deleteBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
        }
        logger.info("Resources deleted.");
    }

    private static String getAccountId() {
        try (StsClient stsClient = StsClient.create()) {
            return stsClient.getCallerIdentity().account();
        }
    }

    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL =
        PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
            Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
    }
}
```

```
    }  
    return fullFilePath;  
  }  
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、5 日より前に作成されたマルチパートアップロードを中止します。

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

例 2: このコマンドは、2014 年 1 月 2 日より前に作成されたマルチパートアップロードを中止します。

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

例 3: このコマンドは、2014 年 1 月 2 日 10:45:37 より前に作成されたマルチパートアップロードを中止します。

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[AbortMultipartUpload](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、5 日より前に作成されたマルチパートアップロードを中止します。

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

例 2: このコマンドは、2014 年 1 月 2 日より前に作成されたマルチパートアップロードを中止します。



```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

例 3: このコマンドは、2014 年 1 月 2 日 10:45:37 より前に作成されたマルチパートアップロードを中止します。

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[AbortMultipartUpload](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CompleteMultipartUpload** を使用する

次のサンプルコードは、CompleteMultipartUpload を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [マルチパートコピーを実行する](#)
- [チェックサムの使用](#)
- [Amazon S3 オブジェクトの整合性の操作](#)

C++

SDK for C++

 Note

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

```
//! Complete a multipart upload to an S3 bucket.
```

```

/ * !
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

    const Aws::String &key,

    const Aws::String &uploadID,

    const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

    const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
        client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

```

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

## CLI

## AWS CLI

次のコマンドは、バケット `amzn-s3-demo-bucket` 内のキー `multipart/01` のマルチパートアップロードを完了します。

```
aws s3api complete-multipart-upload --multipart-upload file://  
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-  
id dfRtDYU0WwCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

このコマンドに必要なアップロード ID `create-multipart-upload` はによって出力され、`list-multipart-uploads` で取得することもできます。

上記のコマンドのマルチパートアップロードオプションは、完全なファイルに再構成する必要があるマルチパートアップロードの部分を記述した JSON 構造を採用しています。この例では、`file://` プレフィックスを使用して、`mpustruct` という名前のローカルフォルダーにあるファイルから JSON 構造を読み込みます。

`mpustruct`:

```
{  
  "Parts": [  
    {  
      "ETag": "e868e0f4719e394144ef36531ee6824c",  
      "PartNumber": 1  
    },  
    {  
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",  
      "PartNumber": 2  
    },  
    {  
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",  
      "PartNumber": 3  
    }  
  ]  
}
```

各パートの ETag 値は、`upload-part` コマンドを使用してパートをアップロードするたびに出力されます。また、`list-parts` を呼び出して取得したり、各パートの MD5 チェックサムを取得して計算したりすることもできます。

出力:

```
{
  "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
  "Bucket": "amzn-s3-demo-bucket",
  "Location": "https://amzn-s3-demo-bucket.s3.amazonaws.com/multipart%2F01",
  "Key": "multipart/01"
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[CompleteMultipartUpload](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

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

```
// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;
```

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
```

```
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
```

```
        .part_number(part_number)
        .build(),
    );
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CopyObject** を使用する

次のサンプルコードは、CopyObject を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [暗号化の開始方法](#)
- [オブジェクトストレージの使用開始](#)
- [条件付きリクエストの実行](#)

.NET

SDK for .NET (v4)

 Note

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

```
/// <summary>
/// Copies an object in an Amazon S3 bucket to a folder within the
/// same bucket.
```

```
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// object to copy is located.</param>
/// <param name="objectName">The object to be copied.</param>
/// <param name="folderName">The folder to which the object will
/// be copied.</param>
/// <returns>A boolean value that indicates the success or failure of
/// the copy operation.</returns>
public async Task<bool> CopyObjectInBucketAsync(
    string bucketName,
    string objectName,
    string folderName)
{
    try
    {
        var request = new CopyObjectRequest
        {
            SourceBucket = bucketName,
            SourceKey = objectName,
            DestinationBucket = bucketName,
            DestinationKey = $"{folderName}\\{objectName}",
        };
        var response = await _amazonS3.CopyObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error copying object: '{ex.Message}'");
        return false;
    }
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[CopyObject](#)」を参照してください。

SDK for .NET

#### Note

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

条件付きリクエストを使用してオブジェクトをコピーします。

```
/// <summary>
/// Copies an object from one Amazon S3 bucket to another with a conditional
request.
/// </summary>
/// <param name="sourceKey">The key of the source object to copy.</param>
/// <param name="destKey">The key of the destination object.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="destBucket">The destination bucket of the object.</param>
/// <param name="conditionType">The type of condition to apply, e.g.
'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional copy is successful, False otherwise.</
returns>
public async Task<bool> CopyObjectConditional(string sourceKey, string
destKey, string sourceBucket, string destBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
        var copyObjectRequest = new CopyObjectRequest
        {
            DestinationBucket = destBucket,
            DestinationKey = destKey,
            SourceBucket = sourceBucket,
            SourceKey = sourceKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                copyObjectRequest.ETagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfModifiedSince:
```



```
        copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
        break;
        case S3ConditionType.IfUnmodifiedSince:
            copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
            break;
        default:
            throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
    }

    await _amazonS3.CopyObjectAsync(copyObjectRequest);
    _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
    return true;
}
catch (AmazonS3Exception e)
{
    if (e.ErrorCode == "PreconditionFailed")
    {
        _logger.LogError("Conditional copy failed: Precondition failed");
    }
    else if (e.ErrorCode == "304")
    {
        _logger.LogError("Conditional copy failed: Object not modified");
    }
    else
    {
        _logger.LogError($"Unexpected error: {e.ErrorCode}");
        throw;
    }
    return false;
}
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[CopyObject](#)」を参照してください。

## Bash

### AWS CLIBash スクリプトを使用した

#### Note

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

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file from and to.
#     $2 - The key of the source file to copy.
#     $3 - The key of the destination file.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_item_in_bucket() {
    local bucket_name=$1
    local source_key=$2
    local destination_key=$3
    local response

    response=$(aws s3api copy-object \
        --bucket "$bucket_name" \
        --copy-source "$bucket_name/$source_key" \
```

```
--key "$destination_key")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
    return 1
fi
}
```

- APIの詳細については、AWS CLI コマンドリファレンスの「[CopyObject](#)」を参照してください。

## C++

### SDK for C++

#### Note

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

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
    &fromBucket, const Aws::String &toBucket,
                        const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
        .WithKey(objectKey)
        .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
```

```
std::cout << "Successfully copied " << objectKey << " from " <<
fromBucket <<
    " to " << toBucket << "." << std::endl;
}

return outcome.IsSuccess();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[CopyObject](#)」を参照してください。

## CLI

### AWS CLI

次のコマンドは、bucket-1 から bucket-2 にオブジェクトをコピーします。

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --
bucket bucket-2
```

出力:

```
{
  "CopyObjectResult": {
    "LastModified": "2015-11-10T01:07:25.000Z",
    "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""
  },
  "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifM1kA"
}
```

- API の詳細については、「AWS CLI Command Reference」の「[CopyObject](#)」を参照してください。

## Go

## SDK for Go V2

 Note

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

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}  
  
// CopyToBucket copies an object in a bucket to another bucket.
```

```
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
destinationBucket string, objectKey string) error {
_, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
    Bucket:      aws.String(destinationBucket),
    CopySource:  aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
    Key:         aws.String(objectKey),
})
if err != nil {
    var notActive *types.ObjectNotInActiveTierError
    if errors.As(err, &notActive) {
        log.Printf("Couldn't copy object %s from %s because the object isn't in the
active tier.\n",
            objectKey, sourceBucket)
        err = notActive
    }
} else {
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
aws.String(objectKey)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
}
return err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

[S3Client](#) を使用してオブジェクトをコピーします。

```

/**
 * Asynchronously copies an object from one S3 bucket to another.
 *
 * @param fromBucket the name of the source S3 bucket
 * @param objectKey the key (name) of the object to be copied
 * @param toBucket the name of the destination S3 bucket
 * @return a {@link CompletableFuture} that completes with the copy result as
a {@link String}
 * @throws RuntimeException if the URL could not be encoded or an S3
exception occurred during the copy
 */
public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
    response.whenComplete((copyRes, ex) -> {
        if (copyRes != null) {
            logger.info("The " + objectKey + " was copied to " + toBucket);
        } else {
            throw new RuntimeException("An S3 exception occurred during
copy", ex);
        }
    });

    return response.thenApply(CopyObjectResponse::copyObjectResult)
        .thenApply(Object::toString);
}

```

[S3TransferManager](#) を使用して、1 つのバケットから別のバケットへ [オブジェクトをコピー](#) します。 [完全なファイル](#) と [テスト](#) を表示します。

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;

```

```
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

public String copyObject(S3TransferManager transferManager, String
bucketName,
    String key, String destinationBucket, String destinationKey) {
    CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
        .sourceBucket(bucketName)
        .sourceKey(key)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    CopyRequest copyRequest = CopyRequest.builder()
        .copyObjectRequest(copyObjectRequest)
        .build();

    Copy copy = transferManager.copy(copyRequest);

    CompletedCopy completedCopy = copy.completionFuture().join();
    return completedCopy.response().copyObjectResult().eTag();
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[CopyObject](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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



オブジェクトをコピーします。

```
import {
  S3Client,
  CopyObjectCommand,
  ObjectNotInActiveTierError,
  waitUntilObjectExists,
} from "@aws-sdk/client-s3";

/**
 * Copy an S3 object from one bucket to another.
 *
 * @param {{
 *   sourceBucket: string,
 *   sourceKey: string,
 *   destinationBucket: string,
 *   destinationKey: string }} config
 */
export const main = async ({
  sourceBucket,
  sourceKey,
  destinationBucket,
  destinationKey,
}) => {
  const client = new S3Client({});

  try {
    await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucket}/${sourceKey}`,
        Bucket: destinationBucket,
        Key: destinationKey,
      }),
    );
    await waitUntilObjectExists(
      { client },
      { Bucket: destinationBucket, Key: destinationKey },
    );
    console.log(
      `Successfully copied ${sourceBucket}/${sourceKey} to ${destinationBucket}/${destinationKey}`,
    );
  } catch (caught) {
    if (caught instanceof ObjectNotInActiveTierError) {
```

```
        console.error(
            `Could not copy ${sourceKey} from ${sourceBucket}. Object is not in the
            active tier.` ,
        );
    } else {
        throw caught;
    }
}
};
```

ETag が指定されたものと一致しないことを条件にオブジェクトをコピーします。

```
import {
    CopyObjectCommand,
    NoSuchKey,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
    type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
    destinationBucketName: string, eTag: string }}
 */
export const main = async ({
    sourceBucketName,
    sourceKeyName,
    destinationBucketName,
    eTag,
}) => {
    const client = new S3Client({});
    const name = data.name;
    try {
        const response = await client.send(
            new CopyObjectCommand({
```

```
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfMatch: eTag,
    )),
    );
    console.log("Successfully copied object to bucket.");
} catch (caught) {
    if (caught instanceof NoSuchKey) {
        console.error(
            `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Unable to copy object "${sourceKeyName}" to bucket
"${sourceBucketName}": ${caught.name}: ${caught.message}`
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        sourceBucketName: {
            type: "string",
            required: true,
        },
        sourceKeyName: {
            type: "string",
            required: true,
        },
        destinationBucketName: {
            type: "string",
            required: true,
        },
    };
};
```

```

    },
    eTag: {
      type: "string",
      required: true,
    },
  },
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}

```

ETag が指定されたものと一致しないことを条件にオブジェクトをコピーします。

```

import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string, eTag: string }}
 */
export const main = async ({

```

```
    sourceBucketName,
    sourceKeyName,
    destinationBucketName,
    eTag,
  }) => {
    const client = new S3Client({});
    const name = data.name;

    try {
      const response = await client.send(
        new CopyObjectCommand({
          CopySource: `${sourceBucketName}/${sourceKeyName}`,
          Bucket: destinationBucketName,
          Key: `${name}${sourceKeyName}`,
          CopySourceIfNoneMatch: eTag,
        }),
      );
      console.log("Successfully copied object to bucket.");
    } catch (caught) {
      if (caught instanceof NoSuchKey) {
        console.error(
          `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`,
        );
      } else if (caught instanceof S3ServiceException) {
        console.error(
          `Unable to copy object "${sourceKeyName}" to bucket
"${sourceBucketName}": ${caught.name}: ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  };

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utills/util-node.js";

const loadArgs = () => {
  const options = {
```

```
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

特定の期間内に作成または変更されたという条件に基づいて、オブジェクトをコピーします。

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);

  const name = data.name;
  const client = new S3Client({});
  const copySource = `${sourceBucketName}/${sourceKeyName}`;
  const copiedKey = name + sourceKeyName;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
        CopySourceIfModifiedSince: date,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
        "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while copying object from ${sourceBucketName}.
        ${caught.name}: ${caught.message}`,
      );
    }
  }
}
```

```
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```



特定の期間内に作成または変更されていないという条件に基づいて、オブジェクトをコピーします。

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);
  const client = new S3Client({});
  const name = data.name;
  const copiedKey = name + sourceKeyName;
  const copySource = `${sourceBucketName}/${sourceKeyName}`;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
        CopySourceIfUnmodifiedSince: date,
      }),
    );
  }
```

```
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while copying object from ${sourceBucketName}.
${caught.name}: ${caught.message}`
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};
```

```
if (isMain(import.meta.url)) {  
  const { errors, results } = loadArgs();  
  if (!errors) {  
    main(results.values);  
  } else {  
    console.error(errors.join("\n"));  
  }  
}
```

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

## Kotlin

### SDK for Kotlin

#### Note

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

```
suspend fun copyBucketObject(  
    fromBucket: String,  
    objectKey: String,  
    toBucket: String,  
) {  
    var encodedUrl = ""  
    try {  
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",  
StandardCharsets.UTF_8.toString())  
    } catch (e: UnsupportedOperationException) {  
        println("URL could not be encoded: " + e.message)  
    }  
  
    val request =  
        CopyObjectRequest {  
            copySource = encodedUrl  
            bucket = toBucket
```

```
        key = objectKey
    }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

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

## PHP

### SDK for PHP

#### Note

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

オブジェクトの単純なコピー。

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $fileName with error: " . $exception-
    >getMessage();
    exit("Please fix error with object copying before continuing.");
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」から同じバケットにコピーします。ただし、新しいキー「sample-copy.txt」を使用します。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey sample-copy.txt
```

例 2: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」からバケット「backup-files」にコピーします。ただし、キーは「sample-copy.txt」を使用します。

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

例 3: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」からダウンロードして、「local-sample.txt」という名前のローカルファイルに保存します。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-sample.txt
```

例 4: 指定したファイルに 1 つのオブジェクトをダウンロードします。ダウンロードしたファイルは c:\downloads\data\archive.zip に保存されます

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:\downloads
```

例 5: 指定した key prefix と一致するすべてのオブジェクトをローカルフォルダにダウンロードします。相対的なキー階層は、ダウンロードの場所全体のサブフォルダとして保存されます。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:\downloads
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[CopyObject](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」から同じバケットにコピーします。ただし、新しいキー「sample-copy.txt」を使用します。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey sample-copy.txt
```

例 2: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」からバケット「backup-files」にコピーします。ただし、キーは「sample-copy.txt」を使用します。

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

例 3: このコマンドは、オブジェクト「sample.txt」をバケット「test-files」からダウンロードして、「local-sample.txt」という名前のローカルファイルに保存します。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-sample.txt
```

例 4: 指定したファイルに 1 つのオブジェクトをダウンロードします。ダウンロードしたファイルは c:\downloads\data\archive.zip に保存されます

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:\downloads
```

例 5: 指定した key prefix と一致するすべてのオブジェクトをローカルフォルダにダウンロードします。相対的なキー階層は、ダウンロードの場所全体のサブフォルダとして保存されます。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:\downloads
```

例 6: マルチパート並列ダウンロードを使用して 1 つのオブジェクトをローカルファイルにダウンロードし、大きなファイルのパフォーマンスを向上させます。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-sample.txt -UseMultipartDownload
```

例 7: カスタム 64 MB のパートサイズで RANGE モードを使用して大きなオブジェクトをダウンロードします。RANGE モードは HTTP バイト範囲リクエストを使用し、アップロード方法に関係なくすべてのオブジェクトで機能します。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

例 8: マルチパートダウンロードと同時ファイルダウンロードを使用して、キープレフィックスに一致するすべてのオブジェクトをローカルフォルダにダウンロードします。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

例 9: マルチパートダウンロードと 20 の同時接続を使用して単一のオブジェクトをダウンロードし、高帯域幅ネットワークの最大スループットを実現します。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

例 10: マルチパートダウンロードを使用して、キープレフィックスに一致するすべてのオブジェクトをローカルフォルダにダウンロードし、中止するのではなく、個々のファイルの失敗をスキップして続行します。

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[CopyObject](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
        key.
                           This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
```



```

        dest_object.key,
    )
except ClientError:
    logger.exception(
        "Couldn't copy object from %s/%s to %s/%s.",
        self.object.bucket_name,
        self.object.key,
        dest_object.bucket_name,
        dest_object.key,
    )
    raise

```

条件付きリクエストを使用してオブジェクトをコピーします。

```

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
        request.

```

```
:param source_key: The key of the source object to copy.
:param dest_key: The key of the destination object.
:param source_bucket: The source bucket of the object.
:param dest_bucket: The destination bucket of the object.
:param condition_type: The type of condition to apply, e.g.
'CopySourceIfMatch', 'CopySourceIfNoneMatch',
'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
:param condition_value: The value to use for the condition.
"""
try:
    self.s3.copy_object(
        Bucket=dest_bucket,
        Key=dest_key,
        CopySource={"Bucket": source_bucket, "Key": source_key},
        **{condition_type: condition_value},
    )
    print(
        f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."
    )
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    if error_code == "PreconditionFailed":
        print("\tConditional copy failed: Precondition failed")
    elif error_code == "304": # Not modified error code.
        print("\tConditional copy failed: Object not modified")
    else:
        logger.error(f"Unexpected error: {error_code}")
        raise
```

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

## Ruby

### SDK for Ruby

#### Note

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

オブジェクトをコピーします。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for
  #                                     copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
  # target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  # nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
    why: #{e.message}"
  end
end

# Example usage:
```

```
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
  #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

オブジェクトをコピーし、宛先オブジェクトにサーバー側の暗号化を追加します。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for
  #                                     copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
  # target key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  nil.
  def copy_object(target_bucket, target_object_key, encryption)
```

```
@source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
server_side_encryption: encryption)
  target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key,
target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
#{target_object.bucket_name}:#{target_object.key} and "\
    "encrypted the target with #{target_object.server_side_encryption}
encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

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

## Rust

### SDK for Rust

#### Note

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

```
/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object}
with etag {}",
        response
            .copy_object_result
            .unwrap_or_else(||
aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.  
    lo_s3->copyobject(  
        iv_bucket = iv_dest_bucket  
        iv_key = iv_dest_object  
        iv_copysource = |{ iv_src_bucket }/{ iv_src_object }| ).  
    MESSAGE 'Object copied to another bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
    MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

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

## Swift

### SDK for Swift

#### Note

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

```
import AWSS3

public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
    let srcUrl = ("\"(sourceBucket)/
\"(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

    let input = CopyObjectInput(
        bucket: destBucket,
        copySource: srcUrl,
        key: name
    )
    do {
        _ = try await client.copyObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Copying an object."))
        throw error
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CreateBucket** を使用する

次のサンプルコードは、CreateBucket を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [設定管理の使用開始](#)
- [機械学習特徴量ストアの使用開始](#)
- [オブジェクトストレージの使用開始](#)



- [S3 を使用して大量のメッセージを管理する](#)
- [バージョン管理されたオブジェクトを操作する](#)

## .NET

### SDK for .NET (v4)

#### Note

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

```
/// <summary>
/// Shows how to create a new Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>A boolean value representing the success or failure of
/// the bucket creation process.</returns>
public async Task<bool> CreateBucketAsync(string bucketName)
{
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
        };

        var response = await _amazonS3.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}
```

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

## SDK for .NET

### Note

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

オブジェクトロックを有効にしてバケットを作成します。

```
/// <summary>
/// Create a new Amazon S3 bucket with object lock actions.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
{
    Console.WriteLine($"\\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
            ObjectLockEnabledForBucket = enableObjectLock,
        };

        var response = await _amazonS3.PutBucketAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}
```

```
}
```

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

## Bash

### AWS CLIBash スクリプトを使用した

#### Note

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

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
```

```

# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
#                        create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally
unique."
        echo "  [-r region_code]   The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
}

```

```
if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
fi

local bucket_config_arg
# A location constraint for "us-east-1" returns an error.
if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
fi

iecho "Parameters:\n"
iecho "    Bucket name:  $bucket_name"
iecho "    Region code:  $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
fi
}
```

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

## C++

## SDK for C++

 Note

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

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                               const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(
            Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
            std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
            " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

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

## CLI

### AWS CLI

例 1: バケットを作成するには

次の create-bucket の例は、amzn-s3-demo-bucket という名前のバケットを作成します。

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region us-east-1
```

出力:

```
{  
  "Location": "/amzn-s3-demo-bucket"  
}
```

詳細については、「Amazon S3 ユーザーガイド」の「[バケットの作成](#)」を参照してください。

例 2: 所有者の強制を使用してバケットを作成するには

次の create-bucket の例は、S3 オブジェクトの所有権のバケット所有者の強制設定を使用して、amzn-s3-demo-bucket という名前のバケットを作成します。

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region us-east-1 \  
  --object-ownership BucketOwnerEnforced
```

出力:

```
{  
  "Location": "/amzn-s3-demo-bucket"  
}
```

詳細については、Amazon S3 ユーザーガイドの[オブジェクトの所有権のコントロールと ACL の無効化](#)を参照してください。

例 3: ``us-east-1`` リージョンの外にバケットを作成するには

次の create-bucket の例は、eu-west-1 リージョンに amzn-s3-demo-bucket という名前のバケットを作成します。us-east-1 の外にある目的のリージョンにバケットを作成するには、適切な LocationConstraint を指定する必要があります。

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region eu-west-1 \  
  --create-bucket-configuration LocationConstraint=eu-west-1
```

出力:

```
{  
  "Location": "http://amzn-s3-demo-bucket.s3.amazonaws.com/"  
}
```

詳細については、「Amazon S3 ユーザーガイド」の「[バケットの作成](#)」を参照してください。

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

Go

SDK for Go V2

 Note

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

デフォルト設定を使用してバケットを作成します。

```
import (  
  "bytes"
```



```
"context"
"errors"
"fmt"
"io"
"log"
"os"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
    string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    })
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", name)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", name)
        }
    }
}
```

```
    err = exists
}
} else {
    err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
    }
}
return err
}
```

オブジェクトロックを使用してバケットを作成し、作成が完了するまで待ちます。

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking
// enabled
// and waits for the bucket to exist before returning.
```

```
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    }

    if enableObjectLock {
        input.ObjectLockEnabledForBucket = aws.Bool(true)
    }

    _, err := actor.S3Client.CreateBucket(ctx, input)
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", bucket)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", bucket)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
        }
    }

    return bucket, err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

バケットを作成します。

```
/**
 * Creates an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes when the bucket is
 *         created and ready
 * @throws RuntimeException if there is a failure while creating the bucket
 */
public CompletableFuture<Void> createBucketAsync(String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    CompletableFuture<CreateBucketResponse> response =
        getAsyncClient().createBucket(bucketRequest);
    return response.thenCompose(resp -> {
        S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<WaiterResponse<HeadBucketResponse>>
        waiterResponseFuture =
            s3Waiter.waitUntilBucketExists(bucketRequestWait);
        return waiterResponseFuture.thenAccept(waiterResponse -> {
            waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                logger.info(bucketName + " is ready");
            });
        });
    });
}
```

```
    }).whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to create bucket", ex);
        }
    });
}
```

オブジェクトロックを有効にしてバケットを作成します。

```
// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}
```

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

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

バケットを作成します。

```
import {
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  CreateBucketCommand,
  S3Client,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * Create an Amazon S3 bucket.
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Location } = await client.send(
      new CreateBucketCommand({
        // The name of the bucket. Bucket names are unique and have several other
        // constraints.
        // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucketnamingrules.html
        Bucket: bucketName,
      }),
    );
    await waitUntilBucketExists({ client }, { Bucket: bucketName });
    console.log(`Bucket created with location ${Location}`);
  } catch (caught) {
    if (caught instanceof BucketAlreadyExists) {
      console.error(
        `The bucket "${bucketName}" already exists in another AWS account. Bucket names must be globally unique.`,
      );
    }
    // WARNING: If you try to create a bucket in the North Virginia region,
    // and you already own a bucket in that region with the same name, this
    // error will not be thrown. Instead, the call will return successfully
    // and the ACL on that bucket will be reset.
    else if (caught instanceof BucketAlreadyOwnedByYou) {
      console.error(
        `The bucket "${bucketName}" already exists in this AWS account.`,
      );
    }
  }
}
```

```
    } else {  
        throw caught;  
    }  
}  
};
```

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

## Kotlin

### SDK for Kotlin

#### Note

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

```
suspend fun createNewBucket(bucketName: String) {  
    val request =  
        CreateBucketRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.createBucket(request)  
        println("$bucketName is ready")  
    }  
}
```

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

## PHP

### SDK for PHP

#### Note

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

バケットを作成します。

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $this->bucketName \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

デフォルトの設定でバケットを作成します。



```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def create(self, region_override=None):
        """
        Create an Amazon S3 bucket in the default Region for the account or in
        the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this
        is
                                not specified, the Region configured in your
        shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
            region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
            region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
            raise error
```

ライフサイクル設定を使用してバージョン対応バケットを作成します。

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
    lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                    configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
```

```
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
  # This is a client-side object until
  # create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
    else

```

```

        @bucket.client.get_bucket_location(bucket:
@bucket.name).location_constraint
      end
      rescue Aws::Errors::ServiceError => e
        "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
      end
    end
  end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{@Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{@wrapper.bucket.name}."
  puts "Your bucket's region is: #{@wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__

```

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

## Rust

### SDK for Rust

#### Note

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

```

pub async fn create_bucket(
  client: &aws_sdk_s3::Client,
  bucket_name: &str,
  region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
S3ExampleError> {

```

```
let constraint =
aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
    .location_constraint(constraint)
    .build();
let create = client
    .create_bucket()
    .create_bucket_configuration(cfg)
    .bucket(bucket_name)
    .send()
    .await;

// BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
task.
create.map(Some).or_else(|err| {
    if err
        .as_service_error()
        .map(|se| se.is_bucket_already_exists() ||
se.is_bucket_already_owned_by_you())
        == Some(true)
    {
        Ok(None)
    } else {
        Err(S3ExampleError::from(err))
    }
}))
}
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```

TRY.
    " determine our region from our session
    DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
    DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
    " When in the us-east-1 region, you must not specify a constraint
    " In all other regions, specify the region as the constraint
    IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
    ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
    ENDIF.

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
    CATCH /aws1/cx_s3_bucketalrddyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
    CATCH /aws1/cx_s3_bktalrddyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.

```

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

## Swift

### SDK for Swift

#### Note

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

```

import AWSS3

public func createBucket(name: String) async throws {
    var input = CreateBucketInput(

```

```
        bucket: name
    )

    // For regions other than "us-east-1", you must set the
    locationConstraint in the createBucketConfiguration.
    // For more information, see LocationConstraint in the S3 API guide.
    // https://docs.aws.amazon.com/AmazonS3/latest/API/
    API_CreateBucket.html#API_CreateBucket_RequestBody
    if let region = configuration.region {
        if region != "us-east-1" {
            input.createBucketConfiguration =
            S3ClientTypes.CreateBucketConfiguration(locationConstraint:
            S3ClientTypes.BucketLocationConstraint(rawValue: region))
        }
    }

    do {
        _ = try await client.createBucket(input: input)
    }
    catch let error as BucketAlreadyOwnedByYou {
        print("The bucket '\(name)' already exists and is owned by you. You
        may wish to ignore this exception.")
        throw error
    }
    catch {
        print("ERROR: ", dump(error, name: "Creating a bucket"))
        throw error
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK で **CreateMultiRegionAccessPoint** を使用する

次の例は、CreateMultiRegionAccessPoint を使用する方法を説明しています。



## Kotlin

### SDK for Kotlin

#### Note

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

us-west-2 リージョンにリクエストを送信するように S3 コントロールクライアントを設定します。

```
suspend fun createS3ControlClient(): S3ControlClient {  
    // Configure your S3ControlClient to send requests to US West  
    (Oregon).  
    val s3Control = S3ControlClient.fromEnvironment {  
        region = "us-west-2"  
    }  
    return s3Control  
}
```

マルチリージョンアクセスポイントを作成します。

```
suspend fun createMrap(  
    s3Control: S3ControlClient,  
    accountIdParam: String,  
    bucketName1: String,  
    bucketName2: String,  
    mrapName: String,  
): String {  
    println("Creating MRAP ...")  
    val createMrapResponse: CreateMultiRegionAccessPointResponse =  
        s3Control.createMultiRegionAccessPoint {  
            accountId = accountIdParam  
            clientToken = UUID.randomUUID().toString()  
            details {  
                name = mrapName  
                regions = listOf(  
                    Region {  
                        bucket = bucketName1
```

```

        },
        Region {
            bucket = bucketName2
        },
    )
}
}

val requestToken: String? = createMrapResponse.requestTokenArn

// Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
if (requestToken != null) {
    waitForSucceededStatus(s3Control, requestToken, accountIdParam)
    println("MRAP created")
}

val getMrapResponse =
    s3Control.getMultiRegionAccessPoint(
        input = GetMultiRegionAccessPointRequest {
            accountId = accountIdParam
            name = mrapName
        },
    )
val mrapAlias = getMrapResponse.accessPoint?.alias
return "arn:aws:s3:::$accountIdParam:accesspoint/$mrpAlias"
}

```

マルチリージョンアクセスポイントが使用可能になるまで待ちます。

```

suspend fun waitForSucceededStatus(
    s3Control: S3ControlClient,
    requestToken: String,
    accountIdParam: String,
    timeBetweenChecks: Duration = 1.minutes,
) {
    var describeResponse: DescribeMultiRegionAccessPointOperationResponse
    describeResponse = s3Control.describeMultiRegionAccessPointOperation(
        input = DescribeMultiRegionAccessPointOperationRequest {
            accountId = accountIdParam
            requestTokenArn = requestToken
        },
    )
}

```

```
var status: String? = describeResponse.asyncOperation?.requestStatus
while (status != "SUCCEEDED") {
    delay(timeBetweenChecks)
    describeResponse =
s3Control.describeMultiRegionAccessPointOperation(
    input = DescribeMultiRegionAccessPointOperationRequest {
        accountId = accountIdParam
        requestTokenArn = requestToken
    },
)
    status = describeResponse.asyncOperation?.requestStatus
    println(status)
}
}
```

- 詳細については、「[AWS SDK for Kotlin デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[CreateMultiRegionAccessPoint](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **CreateMultipartUpload** を使用する

次のサンプルコードは、CreateMultipartUpload を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [マルチパートコピーを実行する](#)
- [チェックサムの使用](#)
- [Amazon S3 オブジェクトの整合性の操作](#)

## C++

## SDK for C++

 Note

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

```
//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
    &key,
                                Aws::S3::Model::ChecksumAlgorithm
checksumAlgorithm,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
        client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
}
```

```
    return uploadID;
}
```

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

## CLI

### AWS CLI

次のコマンドは、キー multipart/01 を使用して、バケット amzn-s3-demo-bucket にマルチパートアップロードを作成します。

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket --key
'multipart/01'
```

出力:

```
{
  "Bucket": "amzn-s3-demo-bucket",
  "UploadId":
  "dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3
  "Key": "multipart/01"
}
```

完成したファイルは、amzn-s3-demo-bucket バケット内の multipart というフォルダで 01 という名前が付けられます。upload-part コマンドでできるように、アップロード ID、キー、バケット名を保存します。

- API の詳細については、AWS CLI コマンドリファレンスの「[CreateMultipartUpload](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

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

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
```

```
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }
}
```

```
// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
    CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **CreatePresignedPost** を使用する

次の例は、CreatePresignedPost を使用する方法を説明しています。

.NET

SDK for .NET (v4)

### Note

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

署名付き POST URL を作成します。

```
/// <summary>
/// Create a presigned POST URL with conditions.
/// </summary>
/// <param name="s3Client">The Amazon S3 client.</param>
/// <param name="bucketName">The name of the bucket.</param>
/// <param name="objectKey">The object key (path) where the uploaded file
will be stored.</param>
/// <param name="expires">When the presigned URL expires.</param>
/// <param name="fields">Dictionary of fields to add to the form.</param>
/// <param name="conditions">List of conditions to apply.</param>
/// <returns>A CreatePresignedPostResponse object with URL and form fields.</
returns>
public async Task<CreatePresignedPostResponse> CreatePresignedPostAsync(
    IAmazonS3 s3Client,
    string bucketName,
    string objectKey,
    DateTime expires,
    Dictionary<string, string>? fields = null,
    List<S3PostCondition>? conditions = null)
{
    var request = new CreatePresignedPostRequest
    {
        BucketName = bucketName,
        Key = objectKey,
        Expires = expires
    };
};
```



```
// Add custom fields if provided
if (fields != null)
{
    foreach (var field in fields)
    {
        request.Fields.Add(field.Key, field.Value);
    }
}

// Add conditions if provided
if (conditions != null)
{
    foreach (var condition in conditions)
    {
        request.Conditions.Add(condition);
    }
}

return await s3Client.CreatePresignedPostAsync(request);
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucket** を使用する

次のサンプルコードは、DeleteBucket を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [設定管理の使用開始](#)
- [機械学習特徴量ストアの使用開始](#)
- [オブジェクトストレージの使用開始](#)

## .NET

### SDK for .NET (v4)

#### Note

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

```
/// <summary>
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest { BucketName = bucketName, };

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting bucket: {ex.Message}");
        return false;
    }
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[DeleteBucket](#)」を参照してください。

## Bash

### AWS CLIBash スクリプトを使用した

#### Note

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

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api delete-bucket \
        --bucket "$bucket_name")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
        return 1
    fi
}
```

```
fi
}
```

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

## C++

### SDK for C++

#### Note

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

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                               const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[DeleteBucket](#)」を参照してください。

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットを削除します。

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket --region us-east-1
```

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

## Go

### SDK for Go V2

#### Note

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

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"
```

```
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is
// returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
    return err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
/**
 * Deletes an S3 bucket asynchronously.
 *
 * @param bucket the name of the bucket to be deleted
 * @return a {@link CompletableFuture} that completes when the bucket
 * deletion is successful, or throws a {@link RuntimeException}
 * if an error occurs during the deletion process
 */
public CompletableFuture<Void> deleteBucketAsync(String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(bucket + " was deleted.");
        } else {
            throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
        }
    });
    return response.thenApply(r -> null);
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[DeleteBucket](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

バケットを削除します。

```
import {
  DeleteBucketCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Delete an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new DeleteBucketCommand({
    Bucket: bucketName,
  });

  try {
    await client.send(command);
    console.log("Bucket was deleted.");
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting bucket. The bucket doesn't exist.`
      );
    }
  }
}
```



```
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while deleting the bucket. ${caught.name}:
            ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};
```

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

## PHP

### SDK for PHP

#### Note

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

空のバケットを削除します。

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $this->s3client->deleteBucket([
        'Bucket' => $this->bucketName,
    ]);
    echo "Deleted bucket $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $this->bucketName with error: " . $exception-
    >getMessage();
    exit("Please fix error with bucket deletion before continuing.");
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、バケット「test-files」からすべてのオブジェクトとオブジェクトバージョンを削除してから、バケットを削除します。このコマンドは、続行する前に確認を求めるプロンプトを表示します。-Force スイッチを追加すると、確認メッセージが表示されなくなります。空でないバケットは削除できないことに注意が必要です。

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteBucket](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、バケット「test-files」からすべてのオブジェクトとオブジェクトバージョンを削除してから、バケットを削除します。このコマンドは、続行する前に確認を求めるプロンプトを表示します。-Force スイッチを追加すると、確認メッセージが表示されなくなります。空でないバケットは削除できないことに注意が必要です。

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteBucket](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
```

```
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

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

## Rust

### SDK for Rust

#### Note

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

```
pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
```

```

        .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
        == Some("NoSuchBucket")
    {
        Ok(())
    } else {
        Err(S3ExampleError::from(err))
    }
}
}
}
}
}

```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

TRY.

```

lo_s3->deletebucket(
    iv_bucket = iv_bucket_name ).
MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

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

## Swift

### SDK for Swift

#### Note

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

```
import AWSS3

public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    do {
        _ = try await client.deleteBucket(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a bucket"))
        throw error
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### CLI で **DeleteBucketAnalyticsConfiguration** を使用する

次のサンプルコードは、DeleteBucketAnalyticsConfiguration を使用する方法を説明しています。

## CLI

### AWS CLI

バケットの分析設定を削除するには

次の `delete-bucket-analytics-configuration` の例では、指定されたバケットと ID の分析設定を削除します。

```
aws s3api delete-bucket-analytics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 1
```

このコマンドでは何も出力されません。

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケット内の「testfilter」という名前の分析フィルターを削除します。

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteBucketAnalyticsConfiguration](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケット内の「testfilter」という名前の分析フィルターを削除します。

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[DeleteBucketAnalyticsConfiguration](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketCors** を使用する

次のサンプルコードは、DeleteBucketCors を使用する方法を説明しています。

.NET

SDK for .NET

 Note

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

```
/// <summary>
/// Deletes a CORS configuration from an Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used
/// to delete the CORS configuration from the bucket.</param>
private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
{
    DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
    {
        BucketName = BucketName,
    };
    await client.DeleteCORSConfigurationAsync(request);
}
```

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



## CLI

### AWS CLI

次のコマンドは、`amzn-s3-demo-bucket` という名前のバケットから Cross-Origin Resource Sharing 設定を削除します。

```
aws s3api delete-bucket-cors --bucket amzn-s3-demo-bucket
```

- API の詳細については、AWS CLI コマンドリファレンスの「[DeleteBucketCors](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
```

```
try:
    self.bucket.Cors().delete()
    logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
except ClientError:
    logger.exception("Couldn't delete CORS from bucket '%s'.",
self.bucket.name)
    raise
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  end
end
```

```
rescue Aws::Errors::ServiceError => e
  puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
#{e.message}"
  false
end

end
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.
  lo_s3->deletebucketcors(
    iv_bucket = iv_bucket_name ).
  MESSAGE 'Bucket CORS configuration deleted.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で DeleteBucketEncryption を使用する

次のサンプルコードは、DeleteBucketEncryption を使用方法を説明しています。

### CLI

#### AWS CLI

バケットのサーバー側の暗号化設定を削除するには

次の delete-bucket-encryption の例では、指定したバケットのサーバー側の暗号化設定を削除します。

```
aws s3api delete-bucket-encryption \  
  --bucket amzn-s3-demo-bucket
```

このコマンドでは何も出力されません。

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

### PowerShell

#### Tools for PowerShell V4

例 1: これは、指定した S3 バケットで有効になっている暗号化を無効にします。

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on  
target "s3casetestbucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteBucketEncryption](#)」を参照してください。

## Tools for PowerShell V5

例 1: これは、指定した S3 バケットで有効になっている暗号化を無効にします。

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[DeleteBucketEncryption](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **DeleteBucketInventoryConfiguration** を使用する

次のサンプルコードは、DeleteBucketInventoryConfiguration を使用方法を説明しています。

CLI

### AWS CLI

バケットのインベントリ設定を削除するには

次の delete-bucket-inventory-configuration の例では、指定したバケットで ID 1 を持つインベントリ設定を削除します。

```
aws s3api delete-bucket-inventory-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 1
```

このコマンドでは何も出力されません。

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケットに対応する「testInventoryName」という名前のインベントリを削除します。

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -  
InventoryId 'testInventoryName'
```

出力:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketInventoryConfiguration  
(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteBucketInventoryConfiguration](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケットに対応する「testInventoryName」という名前のインベントリを削除します。

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -  
InventoryId 'testInventoryName'
```

出力:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketInventoryConfiguration  
(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
```

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):

- APIの詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[DeleteBucketInventoryConfiguration](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketLifecycle** を使用する

次のサンプルコードは、DeleteBucketLifecycle を使用する方法を説明しています。

.NET

SDK for .NET

 Note

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

```
/// <summary>
/// This method removes the Lifecycle configuration from the named
/// S3 bucket.
/// </summary>
/// <param name="client">The S3 client object used to call
/// the RemoveLifecycleConfigAsync method.</param>
/// <param name="bucketName">A string representing the name of the
/// S3 bucket from which the configuration will be removed.</param>
public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client,
string bucketName)
{
    var request = new DeleteLifecycleConfigurationRequest()
    {
        BucketName = bucketName,
    };
    await client.DeleteLifecycleConfigurationAsync(request);
}
```

```
}
```

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

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットからライフサイクル設定を削除します。

```
aws s3api delete-bucket-lifecycle --bucket amzn-s3-demo-bucket
```

- API の詳細については、AWS CLI コマンドリファレンスの「[DeleteBucketLifecycle](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```



```
def delete_lifecycle_configuration(self):
    """
    Remove the lifecycle configuration from the specified bucket.
    """
    try:
        self.bucket.LifecycleConfiguration().delete()
        logger.info(
            "Deleted lifecycle configuration for bucket '%s'.",
            self.bucket.name
        )
    except ClientError:
        logger.exception(
            "Couldn't delete lifecycle configuration for bucket '%s'.",
            self.bucket.name,
        )
        raise
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.
    lo_s3->deletebucketlifecycle(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket lifecycle configuration deleted.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **DeleteBucketMetricsConfiguration** を使用する

次のサンプルコードは、DeleteBucketMetricsConfiguration を使用する方法を説明しています。

### CLI

#### AWS CLI

バケットのメトリクス設定を削除するには

次の delete-bucket-metrics-configuration の例では、指定したバケットと ID のメトリクス設定を削除します。

```
aws s3api delete-bucket-metrics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 123
```

このコマンドでは何も出力されません。

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

### PowerShell

#### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケット内の「testmetrics」という名前のメトリクスフィルターを削除します。

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteBucketMetricsConfiguration](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケット内の「testmetrics」という名前のメトリクスフィルターを削除します。

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId 'testmetrics'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[DeleteBucketMetricsConfiguration](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **DeleteBucketPolicy** を使用する

次のサンプルコードは、DeleteBucketPolicy を使用する方法を説明しています。

### C++

#### SDK for C++

#### Note

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

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
                                     &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);
```

```
Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
client.DeleteBucketPolicy(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: deleteBucketPolicy: " <<
        err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
} else {
    std::cout << "Policy was deleted from the bucket." << std::endl;
}

return outcome.IsSuccess();
}
```

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

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットからバケットポリシーを削除します。

```
aws s3api delete-bucket-policy --bucket amzn-s3-demo-bucket
```

- API の詳細については、AWS CLI コマンドリファレンスの「[DeleteBucketPolicy](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * 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 DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the policy from
(for example, bucket1).""";

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

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }
}
```

```
/**
 * Deletes the S3 bucket policy for the specified bucket.
 *
 * @param s3 the {@link S3Client} instance to use for the operation
 * @param bucketName the name of the S3 bucket for which the policy should be
deleted
 *
 * @throws S3Exception if there is an error deleting the bucket policy
 */
public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketPolicy(delReq);
        System.out.println("Done!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

バケットポリシーを削除します。

```
import {
```

```
DeleteBucketPolicyCommand,
S3Client,
S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the policy from an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Bucket policy deleted from "${bucketName}".`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}. The bucket
        doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}. ${caught.name}:
        ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

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

## Kotlin

### SDK for Kotlin

#### Note

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

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {  
    val request =  
        DeleteBucketPolicyRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteBucketPolicy(request)  
        println("Done!")  
    }  
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケットに関連付けられたバケットポリシーを削除します。

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteBucketPolicy](#)」を参照してください。



## Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケットに関連付けられたバケットポリシーを削除します。

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteBucketPolicy](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
```

```
logger.exception(  
    "Couldn't delete policy for bucket '%s'." % self.bucket.name  
)  
raise
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
# Wraps an Amazon S3 bucket policy.  
class BucketPolicyWrapper  
  attr_reader :bucket_policy  
  
  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object  
  # configured with an existing bucket.  
  def initialize(bucket_policy)  
    @bucket_policy = bucket_policy  
  end  
  
  def delete_policy  
    @bucket_policy.delete  
    true  
  rescue Aws::Errors::ServiceError => e  
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's  
    why: #{e.message}"  
    false  
  end  
  
end
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.  
    lo_s3->deletebucketpolicy(  
        iv_bucket = iv_bucket_name ).  
    MESSAGE 'Bucket policy deleted.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### CLI で **DeleteBucketReplication** を使用する

次のサンプルコードは、DeleteBucketReplication を使用する方法を説明しています。

#### CLI

##### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットからレプリケーションの設定を削除します。

```
aws s3api delete-bucket-replication --bucket amzn-s3-demo-bucket
```

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

## PowerShell

### Tools for PowerShell V4

例 1: 「amzn-s3-demo-bucket」という名前のバケットに関連付けられているレプリケーションの設定を削除します。なお、このオペレーションには s3:DeleteReplicationConfiguration アクションに対するアクセス許可が必要です。オペレーションを続行する前に確認を求めるプロンプトが表示されます。プロンプトを表示しないようにするには、-Force スイッチを使用します。

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteBucketReplication](#)」を参照してください。

### Tools for PowerShell V5

例 1: 「amzn-s3-demo-bucket」という名前のバケットに関連付けられているレプリケーションの設定を削除します。なお、このオペレーションには s3:DeleteReplicationConfiguration アクションに対するアクセス許可が必要です。オペレーションを続行する前に確認を求めるプロンプトが表示されます。プロンプトを表示しないようにするには、-Force スイッチを使用します。

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[DeleteBucketReplication](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **DeleteBucketTagging** を使用する

次のサンプルコードは、DeleteBucketTagging を使用方法を説明しています。

### CLI

#### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットからタグの設定を削除します。

```
aws s3api delete-bucket-tagging --bucket amzn-s3-demo-bucket
```

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

### PowerShell

#### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケットに関連付けられたすべてのタグを削除します。

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"amzn-s3-demo-bucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteBucketTagging](#)」を参照してください。

#### Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケットに関連付けられたすべてのタグを削除します。

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"amzn-s3-demo-bucket".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[DeleteBucketTagging](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteBucketWebsite** を使用する

次のサンプルコードは、DeleteBucketWebsite を使用方法を説明しています。

C++

SDK for C++

 Note

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

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                       const Aws::S3::S3ClientConfiguration
                                       &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
        client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
```

```
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[DeleteBucketWebsite](#)」を参照してください。

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットからウェブサイト設定を削除します。

```
aws s3api delete-bucket-website --bucket amzn-s3-demo-bucket
```

- API の詳細については、AWS CLI コマンドリファレンスの「[DeleteBucketWebsite](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code 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 DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:      <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the website
configuration from.
            "";

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

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket:
%s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Deletes the website configuration for an Amazon S3 bucket.

```



```
*
* @param s3 The {@link S3Client} instance used to interact with Amazon S3.
* @param bucketName The name of the S3 bucket for which the website
configuration should be deleted.
* @throws S3Exception If an error occurs while deleting the website
configuration.
*/
public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName)
{
    DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketWebsite(delReq);
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.out.println("Failed to delete website configuration!");
        System.exit(1);
    }
}
```

- APIの詳細については、AWS SDK for Java 2.x API リファレンスの「[DeleteBucketWebsite](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

バケットからウェブサイト設定を削除します。

```
import {
    DeleteBucketWebsiteCommand,
```

```
S3Client,  
S3ServiceException,  
} from "@aws-sdk/client-s3";  
  
/**  
 * Remove the website configuration for a bucket.  
 * @param {{ bucketName: string }}  
 */  
export const main = async ({ bucketName }) => {  
  const client = new S3Client({});  
  
  try {  
    await client.send(  
      new DeleteBucketWebsiteCommand({  
        Bucket: bucketName,  
      })),  
    );  
    // The response code will be successful for both removed configurations and  
    // configurations that did not exist in the first place.  
    console.log(  
      `The bucket "${bucketName}" is not longer configured as a website, or it  
never was.`,  
    );  
  } catch (caught) {  
    if (  
      caught instanceof S3ServiceException &&  
      caught.name === "NoSuchBucket"  
    ) {  
      console.error(  
        `Error from S3 while removing website configuration from ${bucketName}.  
The bucket doesn't exist.`,  
      );  
    } else if (caught instanceof S3ServiceException) {  
      console.error(  
        `Error from S3 while removing website configuration from ${bucketName}.  
${caught.name}: ${caught.message}`,  
      );  
    } else {  
      throw caught;  
    }  
  }  
};
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケットの静的ウェブサイトホスティングのプロパティを無効にします。

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteBucketWebsite](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケットの静的ウェブサイトホスティングのプロパティを無効にします。

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[DeleteBucketWebsite](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteObject** を使用する

次のサンプルコードは、DeleteObject を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon S3 オブジェクトの整合性の操作](#)
- [バージョン管理されたオブジェクトを操作する](#)

.NET

SDK for .NET

 Note

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

バージョニング非対応の S3 バケットからオブジェクトを削除します。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete an object from a non-versioned Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteObject
{
```

```
/// <summary>
/// The Main method initializes the necessary variables and then calls
/// the DeleteObjectNonVersionedBucketAsync method to delete the object
/// named by the keyName parameter.
/// </summary>
public static async Task Main()
{
    const string bucketName = "amzn-s3-demo-bucket";
    const string keyName = "testfile.txt";

    // If the Amazon S3 bucket is located in an AWS Region other than the
    // Region of the default account, define the AWS Region for the
    // Amazon S3 bucket in your call to the AmazonS3Client constructor.
    // For example RegionEndpoint.USWest2.
    IAmazonS3 client = new AmazonS3Client();
    await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
}

/// <summary>
/// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
/// desired object from the named bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client used to delete
/// an object from an Amazon S3 bucket.</param>
/// <param name="bucketName">The name of the bucket from which the
/// object will be deleted.</param>
/// <param name="keyName">The name of the object to delete.</param>
public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
{
    try
    {
        var deleteObjectRequest = new DeleteObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
        };

        Console.WriteLine($"Deleting object: {keyName}");
        await client.DeleteObjectAsync(deleteObjectRequest);
        Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
    }
}
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' when deleting an object.");
        }
    }
}
```

バージョニングされた S3 バケットからオブジェクトを削除します。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example creates an object in an Amazon Simple Storage Service
/// (Amazon S3) bucket and then deletes the object version that was
/// created.
/// </summary>
public class DeleteObjectVersion
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "verstioned-object.txt";

        // If the AWS Region of the default user is different from the AWS
        // Region of the Amazon S3 bucket, pass the AWS Region of the
        // bucket region to the Amazon S3 client object's constructor.
        // Define it like this:
        //      RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
        IAmazonS3 client = new AmazonS3Client();

        await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
    }

    /// <summary>
    /// This method creates and then deletes a versioned object.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
```

```

    /// create and delete the object.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object will be created and deleted.</param>
    /// <param name="keyName">The key name of the object to create.</param>
    public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
client, string bucketName, string keyName)
    {
        try
        {
            // Add a sample object.
            string versionID = await PutAnObject(client, bucketName,
keyName);

            // Delete the object by specifying an object key and a version
ID.

            DeleteObjectRequest request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = keyName,
                VersionId = versionID,
            };

            Console.WriteLine("Deleting an object");
            await client.DeleteObjectAsync(request);
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }

    /// <summary>
    /// This method is used to create the temporary Amazon S3 object.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 object which will be
used
    /// to create the temporary Amazon S3 object.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
object
    /// will be created.</param>
    /// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
    /// <returns>The Version ID of the created object.</returns>

```

```

    public static async Task<string> PutAnObject(IAmazonS3 client, string
bucketName, string objectKey)
    {
        PutObjectRequest request = new PutObjectRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            ContentBody = "This is the content body!",
        };

        PutObjectResponse response = await client.PutObjectAsync(request);
        return response.VersionId;
    }
}

```

- API の詳細については、AWS SDK for .NET API リファレンスの「[DeleteObject](#)」を参照してください。

## Bash

### AWS CLIBash スクリプトを使用した

#### Note

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

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_item_in_bucket

```



```
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - The key (file name) in the bucket to delete.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_item_in_bucket() {
    local bucket_name=$1
    local key=$2
    local response

    response=$(aws s3api delete-object \
        --bucket "$bucket_name" \
        --key "$key")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
        return 1
    fi
}
```

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

## C++

### SDK for C++

#### Note

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

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
        .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[DeleteObject](#)」を参照してください。

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットから test.txt という名前のオブジェクトを削除します。

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

バケットバージョニングが有効になっている場合、出力には削除マーカのバージョン ID が含まれます。

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tNDlWlGq",
  "DeleteMarker": true
}
```

オブジェクトの削除の詳細については、「Amazon S3 ユーザーガイド」の「オブジェクトの削除」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[DeleteObject](#)」を参照してください。

Go

SDK for Go V2

 Note

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

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}
```

```
}

// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "AccessDenied":
                log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
                err = nil
            case "InvalidArgument":
                if bypassGovernance {
                    log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
                    err = nil
                }
            }
        }
    } else {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
            time.Minute)
        if err != nil {
```

```
    log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.\n", key, bucket)\n    } else {\n        deleted = true\n    }\n}\nreturn deleted, err\n}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
/**\n * Deletes an object from an S3 bucket asynchronously.\n *\n * @param bucketName the name of the S3 bucket\n * @param key         the key (file name) of the object to be deleted\n * @return a {@link CompletableFuture} that completes when the object has\n *         been deleted\n */\npublic CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,\nString key) {\n    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()\n        .bucket(bucketName)\n        .key(key)\n        .build();
```

```
CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
response.whenComplete((deleteRes, ex) -> {
    if (deleteRes != null) {
        logger.info(key + " was deleted");
    } else {
        throw new RuntimeException("An S3 exception occurred during
delete", ex);
    }
});

return response.thenApply(r -> null);
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[DeleteObject](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

オブジェクトを削除します。

```
import {
    DeleteObjectCommand,
    S3Client,
    S3ServiceException,
    waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete one object from an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
```

```
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    await waitUntilObjectNotExists(
      { client },
      { Bucket: bucketName, Key: key },
    );
    // A successful delete, or a delete for a non-existent object, both return
    // a 204 response code.
    console.log(
      `The object "${key}" from bucket "${bucketName}" was deleted, or it didn't
      exist.` ,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}. The bucket
        doesn't exist.` ,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}.  ${caught.name}:
        ${caught.message}` ,
      );
    } else {
      throw caught;
    }
  }
};
```

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

## PHP

## SDK for PHP

 Note

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

```
public function deleteObject(string $bucketName, string $fileName, array
$args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
    $args);
    try {
        $this->client->deleteObject($parameters);
        if ($this->verbose) {
            echo "Deleted the object named: $fileName from $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}
```

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



## Python

### SDK for Python (Boto3)

#### Note

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

オブジェクトを削除します。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def delete(self):
        """
        Deletes the object.
        """
        try:
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't delete object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
```

```
raise
```

オブジェクトの新しいバージョンを削除して、オブジェクトを以前のバージョンにロールバックします。

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
```

```
        else:
            break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
            f"{object_key}."
        )
```

オブジェクトのアクティブな削除マーカーを削除して、削除されたオブジェクトを復活させます。

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest
    version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
```

```

        latest_version["LastModified"],
    )
    obj = bucket.Object(object_key)
    obj.Version(latest_version["VersionId"]).delete()
    logger.info(
        "Revived %s, active version is now %s with body '%s'",
        object_key,
        obj.version_id,
        obj.get()["Body"].read(),
    )
else:
    logger.warning(
        "Delete marker is not the latest version for %s!", object_key
    )
elif "Versions" in response:
    logger.warning("Got an active version for %s, nothing to do.",
object_key)
else:
    logger.error("Couldn't get any version info for %s.", object_key)

```

S3 オブジェクトから削除マーカを削除する Lambda ハンドラを作成します。このハンドラを使用して、バージョン管理されたバケット内の余分な削除マーカを効率的にクリーンアップできます。

```

import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")

def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker

```

```

        to remove.
:param context: Context about the event.
:return: A result structure that Amazon S3 uses to interpret the result of
the
        operation. When the result code is TemporaryFailure, S3 retries the
        operation.
"""
# Parse job parameters from Amazon S3 batch operations
invocation_id = event["invocationId"]
invocation_schema_version = event["invocationSchemaVersion"]

results = []
result_code = None
result_string = None

task = event["tasks"][0]
task_id = task["taskId"]

try:
    obj_key = parse.unquote_plus(task["s3Key"], encoding="utf-8")
    obj_version_id = task["s3VersionId"]
    bucket_name = task["s3BucketArn"].split(":")[-1]

    logger.info(
        "Got task: remove delete marker %s from object %s.", obj_version_id,
obj_key
    )

    try:
        # If this call does not raise an error, the object version is not a
delete
        # marker and should not be deleted.
        response = s3.head_object(
            Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
        )
        result_code = "PermanentFailure"
        result_string = (
            f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
marker."
        )

        logger.debug(response)
        logger.warning(result_string)
    except ClientError as error:

```

```

        delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
    "x-amz-delete-marker", "false"
)
    if delete_marker == "true":
        logger.info(
            "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
        )
        try:
            s3.delete_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "Succeeded"
            result_string = (
                f"Successfully removed delete marker "
                f"{obj_version_id} from object {obj_key}."
            )
            logger.info(result_string)
        except ClientError as error:
            # Mark request timeout as a temporary failure so it will be
retried.

            if error.response["Error"]["Code"] == "RequestTimeout":
                result_code = "TemporaryFailure"
                result_string = (
                    f"Attempt to remove delete marker from "
                    f"object {obj_key} timed out."
                )
                logger.info(result_string)
            else:
                raise
        else:
            raise ValueError(
                f"The x-amz-delete-marker header is either not "
                f"present or is not 'true'."
            )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {

```

```
        "taskId": task_id,  
        "resultCode": result_code,  
        "resultString": result_string,  
    }  
)  
return {  
    "invocationSchemaVersion": invocation_schema_version,  
    "treatMissingKeysAs": "PermanentFailure",  
    "invocationId": invocation_id,  
    "results": results,  
}
```

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

## Rust

### SDK for Rust

#### Note

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

```
/// Delete an object from a bucket.  
pub async fn remove_object(  
    client: &aws_sdk_s3::Client,  
    bucket: &str,  
    key: &str,  
) -> Result<(), S3ExampleError> {  
    client  
        .delete_object()  
        .bucket(bucket)  
        .key(key)  
        .send()  
        .await?;
```

```
// There are no modeled errors to handle when deleting an object.  
  
Ok::<())  
}
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.  
    lo_s3->deleteobject(  
        iv_bucket = iv_bucket_name  
        iv_key = iv_object_key ).  
    MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

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



## Swift

### SDK for Swift

#### Note

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

```
import AWSS3

public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a file. "))
        throw error
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### CLI で **DeleteObjectTagging** を使用する

次のサンプルコードは、DeleteObjectTagging を使用する方法を説明しています。

## CLI

### AWS CLI

オブジェクトのタグセットを削除するには

次の `delete-object-tagging` の例では、指定したキーを持つタグをオブジェクト `doc1.rtf` から削除します。

```
aws s3api delete-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc1.rtf
```

このコマンドでは何も出力されません。

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

## PowerShell

### Tools for PowerShell V4

このコマンドは、指定した S3 バケット内のキー「`testfile.txt`」を持つオブジェクトに関連付けられたすべてのタグを削除します。

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

出力:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target  
"testfile.txt".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y  
testfile.txt
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteObjectTagging](#)」を参照してください。

## Tools for PowerShell V5

このコマンドは、指定した S3 バケット内のキー「testfile.txt」を持つオブジェクトに関連付けられたすべてのタグを削除します。

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

出力:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target  
"testfile.txt".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y  
testfile.txt
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[DeleteObjectTagging](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DeleteObjects** を使用する

次のサンプルコードは、DeleteObjects を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [バケット内のすべてのオブジェクトを削除する](#)
- [オブジェクトストレージの使用開始](#)

## .NET

### SDK for .NET (v4)

#### Note

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

```
/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>
public async Task<bool> DeleteBucketContentsAsync(string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
    try
    {
        // Delete all objects in the bucket.
        var deleteList = await ListBucketContentsAsync(bucketName, false);
        if (deleteList != null && deleteList.Any())
        {
            await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
            });
        }

        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting objects: {ex.Message}");
        return false;
    }
}
```

```
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[DeleteObjects](#)」を参照してください。

SDK for .NET

 Note

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

バージョニング非対応の S3 バケットから複数のオブジェクトを削除します。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete multiple objects from an Amazon Simple
/// Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    /// <summary>
    /// The Main method initializes the Amazon S3 client and the name of
    /// the bucket and then passes those values to MultiObjectDeleteAsync.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";

        // If the Amazon S3 bucket from which you wish to delete objects is
        not
        // located in the same AWS Region as the default user, define the
        // AWS Region for the Amazon S3 bucket as a parameter to the client
        // constructor.
        IAmazonS3 s3Client = new AmazonS3Client();
```

```

        await MultiObjectDeleteAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method uses the passed Amazon S3 client to first create and then
    /// delete three files from the named bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// Amazon S3 methods.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where
objects
    /// will be created and then deleted.</param>
    public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
bucketName)
    {
        // Create three sample objects which we will then delete.
        var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

        // Now perform the multi-object delete, passing the key names and
        // version IDs. Since we are working with a non-versioned bucket,
        // the object keys collection includes null version IDs.
        DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest
        {
            BucketName = bucketName,
            Objects = keysAndVersions,
        };

        // You can add a specific object key to the delete request using the
        // AddKey method of the multiObjectDeleteRequest.
        try
        {
            DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
        }
        catch (DeleteObjectsException e)
        {
            PrintDeletionErrorStatus(e);
        }
    }
}

```

```

    /// <summary>
    /// Prints the list of errors raised by the call to DeleteObjectsAsync.
    /// </summary>
    /// <param name="ex">A collection of exceptions returned by the call to
    /// DeleteObjectsAsync.</param>
    public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
    {
        DeleteObjectsResponse errorResponse = ex.Response;
        Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

        Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
        Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

        Console.WriteLine("Printing error data...");
        foreach (DeleteError deleteError in errorResponse.DeleteErrors)
        {
            Console.WriteLine($"Object Key:
{deleteError.Key}\\t{deleteError.Code}\\t{deleteError.Message}");
        }
    }

    /// <summary>
    /// This method creates simple text file objects that can be used in
    /// the delete method.
    /// </summary>
    /// <param name="client">The Amazon S3 client used to call
    PutObjectAsync.</param>
    /// <param name="number">The number of objects to create.</param>
    /// <param name="bucketName">The name of the bucket where the objects
    /// will be created.</param>
    /// <returns>A list of keys (object keys) and versions that the calling
    /// method will use to delete the newly created files.</returns>
    public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, int number, string bucketName)
    {
        List<KeyVersion> keys = new List<KeyVersion>();
        for (int i = 0; i < number; i++)
        {
            string key = "ExampleObject-" + new System.Random().Next();
            PutObjectRequest request = new PutObjectRequest
            {

```

```
        BucketName = bucketName,
        Key = key,
        ContentBody = "This is the content body!",
    };

    PutObjectResponse response = await
client.PutObjectAsync(request);

    // For non-versioned bucket operations, we only need the
    // object key.
    KeyVersion keyVersion = new KeyVersion
    {
        Key = key,
    };
    keys.Add(keyVersion);
}

return keys;
}
}
```

バージョニングされた S3 バケットから複数のオブジェクトを削除します。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete objects in a version-enabled Amazon
/// Simple StorageService (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region for your Amazon S3 bucket is different from
        // the AWS Region of the default user, define the AWS Region for
```



```

        // the Amazon S3 bucket and pass it to the client constructor
        // like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
        IAmazonS3 s3Client;

        s3Client = new AmazonS3Client();
        await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
    }

    /// <summary>
    /// This method removes multiple versions and objects from a
    /// version-enabled Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
    {
        // Delete objects (specifying object version in the request).
        await DeleteObjectVersionsAsync(client, bucketName);

        // Delete objects (without specifying object version in the request).
        var deletedObjects = await DeleteObjectsAsync(client, bucketName);

        // Additional exercise - remove the delete markers Amazon S3 returned
from
        // the preceding response. This results in the objects reappearing
        // in the bucket (you can verify the appearance/disappearance of
        // objects in the console).
        await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
    }

    /// <summary>
    /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
    /// them again. The method returns a list of the Amazon S3 objects
deleted.
    /// </summary>

```

```

call
    /// <param name="client">The initialized Amazon S3 client object used to
    /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
    /// <param name="bucketName">The name of the bucket where the objects
    /// will be created and then deleted.</param>
    /// <returns>A list of DeletedObjects.</returns>
    public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
    {
        // Upload the sample objects.
        var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

        // Delete objects using only keys. Amazon S3 creates a delete marker
and
        // returns its version ID in the response.
        List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
        return deletedObjects;
    }

    /// <summary>
    /// This method creates several temporary objects and then deletes them.
    /// </summary>
    /// <param name="client">The S3 client.</param>
    /// <param name="bucketName">Name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
    {
        // Upload the sample objects.
        var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

        // Delete the specific object versions.
        await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
    }

    /// <summary>
    /// Displays the list of information about deleted files to the console.
    /// </summary>
    /// <param name="e">Error information from the delete process.</param>
    private static void DisplayDeletionErrors(DeleteObjectsException e)
    {
        var errorResponse = e.Response;

```

```

        Console.WriteLine($"No. of objects successfully deleted =
{errorResponse.DeletedObjects.Count}");
        Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");
        Console.WriteLine("Printing error data...");
        foreach (var deleteError in errorResponse.DeleteErrors)
        {
            Console.WriteLine($"Object Key:
{deleteError.Key}\\t{deleteError.Code}\\t{deleteError.Message}");
        }
    }

    /// <summary>
    /// Delete multiple objects from a version-enabled bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    /// <param name="keys">A list of key names for the objects to delete.</
param>
    private static async Task VersionedDeleteAsync(IAmazonS3 client, string
bucketName, List<KeyVersion> keys)
    {
        var multiObjectDeleteRequest = new DeleteObjectsRequest
        {
            BucketName = bucketName,
            Objects = keys, // This includes the object keys and specific
version IDs.
        };

        try
        {
            Console.WriteLine("Executing VersionedDelete...");
            DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted all the
{response.DeletedObjects.Count} items");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

```

```

    }
}

/// <summary>
/// Deletes multiple objects from a non-versioned Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
/// RemoveDeleteMarkersAsync.</param>
/// <param name="bucketName">The name of the bucket from which to delete
/// objects.</param>
/// <param name="keys">A list of key names for the objects to delete.</
param>
/// <returns>A list of the deleted objects.</returns>
private static async Task<List<DeletedObject>>
NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
keys)
{
    // Create a request that includes only the object key names.
    DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest();
    multiObjectDeleteRequest.BucketName = bucketName;

    foreach (var key in keys)
    {
        multiObjectDeleteRequest.AddKey(key.Key);
    }

    // Execute DeleteObjectsAsync.
    // The DeleteObjectsAsync method adds a delete marker for each
    // object deleted. You can verify that the objects were removed
    // using the Amazon S3 console.
    DeleteObjectsResponse response;
    try
    {
        Console.WriteLine("Executing NonVersionedDelete...");
        response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
        Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
    }
    catch (DeleteObjectsException ex)
    {

```

```
        DisplayDeletionErrors(ex);
        throw; // Some deletions failed. Investigate before continuing.
    }

    // This response contains the DeletedObjects list which we use to
delete the delete markers.
    return response.DeletedObjects;
}

/// <summary>
/// Deletes the markers left after deleting the temporary objects.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
/// RemoveDeleteMarkersAsync.</param>
/// <param name="bucketName">The name of the bucket from which to delete
/// objects.</param>
/// <param name="deletedObjects">A list of the objects that were
deleted.</param>
private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
{
    var keyVersionList = new List<KeyVersion>();

    foreach (var deletedObject in deletedObjects)
    {
        KeyVersion keyVersion = new KeyVersion
        {
            Key = deletedObject.Key,
            VersionId = deletedObject.DeleteMarkerVersionId,
        };
        keyVersionList.Add(keyVersion);
    }

    // Create another request to delete the delete markers.
    var multiObjectDeleteRequest = new DeleteObjectsRequest
    {
        BucketName = bucketName,
        Objects = keyVersionList,
    };

    // Now, delete the delete marker to bring your objects back to the
bucket.
```

```

        try
        {
            Console.WriteLine("Removing the delete markers .....");
            var deleteObjectResponse = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted the
{deleteObjectResponse.DeletedObjects.Count} delete markers");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

    /// <summary>
    /// Create temporary Amazon S3 objects to show how object deletion works
in an
    /// Amazon S3 bucket with versioning enabled.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// PutObjectAsync to create temporary objects for the example.</param>
    /// <param name="bucketName">A string representing the name of the S3
    /// bucket where we will create the temporary objects.</param>
    /// <param name="number">The number of temporary objects to create.</
param>
    /// <returns>A list of the KeyVersion objects.</returns>
    private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, string bucketName, int number)
    {
        var keys = new List<KeyVersion>();

        for (var i = 0; i < number; i++)
        {
            string key = "ObjectToDelete-" + new System.Random().Next();
            PutObjectRequest request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = key,
                ContentBody = "This is the content body!",
            };

            var response = await client.PutObjectAsync(request);
            KeyVersion keyVersion = new KeyVersion

```

```
        {
            Key = key,
            VersionId = response.VersionId,
        };

        keys.Add(keyVersion);
    }

    return keys;
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[DeleteObjects](#)」を参照してください。

## Bash

AWS CLIBash スクリプトを使用した

### Note

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

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
```

```

# Parameters:
#     $1 - The name of the bucket.
#     $2 - A list of keys in the bucket to delete.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":["
    for key in $keys; do
        delete_items="$delete_items{\"Key\": \"$key\"},"
    done
    delete_items=${delete_items%?} # Remove the final comma.
    delete_items="$delete_items]"

    response=$(aws s3api delete-objects \
        --bucket "$bucket_name" \
        --delete "$delete_items")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
        return 1
    fi
}

```

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



## C++

## SDK for C++

 Note

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

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                                const Aws::String &fromBucket,
                                const Aws::S3::S3ClientConfiguration
                                &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {

deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
        client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }
    }
}
```

```
        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[DeleteObjects](#)」を参照してください。

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットからオブジェクトを削除します。

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

delete.json は、現在のディレクトリにある JSON ドキュメントで、削除するオブジェクトを指定します。

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

出力:

```
{
  "Deleted": [
    {
      "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUP01koHwzU",
      "Key": "test1.txt",
      "DeleteMarker": true
    }
  ]
}
```

```
]
}
```

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

Go

SDK for Go V2

 Note

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

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// DeleteObjects deletes a list of objects from a bucket.
```

```
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &types.Delete{
            Objects: objects,
            Quiet:   aws.Bool(true),
        },
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
    if err != nil || len(delOut.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucket)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
        } else if len(delOut.Errors) > 0 {
            for _, outErr := range delOut.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *delOut.Errors[0].Message)
        }
    } else {
        for _, delObj := range delOut.Deleted {
            err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObj.Key},
                time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                    *delObj.Key)
            } else {
                log.Printf("Deleted %s.\n", *delObj.Key)
            }
        }
    }
}
```

```
    return err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.ArrayList;

/**
 * Before running this Java V2 code 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 DeleteMultiObjects {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:    <bucketName>

        Where:
            bucketName - the Amazon S3 bucket name.
        "";

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

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    deleteBucketObjects(s3, bucketName);
    s3.close();
}

/**
 * Deletes multiple objects from an Amazon S3 bucket.
 *
 * @param s3 An Amazon S3 client object.
 * @param bucketName The name of the Amazon S3 bucket to delete objects from.
 */
public static void deleteBucketObjects(S3Client s3, String bucketName) {
    // Upload three sample objects to the specified Amazon S3 bucket.
    ArrayList<ObjectIdentifier> keys = new ArrayList<>();
    PutObjectRequest putOb;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putOb = PutObjectRequest.builder()
            .bucket(bucketName)
```

```
        .key(keyName)
        .build();

        s3.putObject(putOb, RequestBody.fromString(keyName));
        keys.add(objectId);
    }

    System.out.println(keys.size() + " objects successfully created.");

    // Delete multiple objects in one request.
    Delete del = Delete.builder()
        .objects(keys)
        .build();

    try {
        DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
        .bucket(bucketName)
        .delete(del)
        .build();

        s3.deleteObjects(multiObjectDeleteRequest);
        System.out.println("Multiple objects are deleted!");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- APIの詳細については、AWS SDK for Java 2.x API リファレンスの「[DeleteObjects](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

複数のオブジェクトを削除します。

```
import {
  DeleteObjectsCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete multiple objects from an S3 bucket.
 * @param {{ bucketName: string, keys: string[] }}
 */
export const main = async ({ bucketName, keys }) => {
  const client = new S3Client({});

  try {
    const { Deleted } = await client.send(
      new DeleteObjectsCommand({
        Bucket: bucketName,
        Delete: {
          Objects: keys.map((k) => ({ Key: k })),
        },
      }),
    );
    for (const key in keys) {
      await waitUntilObjectNotExists(
        { client },
        { Bucket: bucketName, Key: key },
      );
    }
    console.log(
```



```
        `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while deleting objects from ${bucketName}. The bucket
doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while deleting objects from ${bucketName}.
${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};
```

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

## Kotlin

### SDK for Kotlin

#### Note

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

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
```

```
) {  
    val objectId =  
        ObjectIdentifier {  
            key = objectName  
        }  
  
    val delObj =  
        Delete {  
            objects = listOf(objectId)  
        }  
  
    val request =  
        DeleteObjectsRequest {  
            bucket = bucketName  
            delete = delObj  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteObjects(request)  
        println("$objectName was deleted from $bucketName")  
    }  
}
```

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

## PHP

### SDK for PHP

#### Note

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

キーのリストからオブジェクトのセットを削除します。

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```

```
try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $this->s3client->deleteObjects([
        'Bucket' => $this->bucketName,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    if (isset($check['Contents']) && count($check['Contents']) > 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、バケット「test-files」からオブジェクト「sample.txt」を削除します。コマンドを実行する前に、確認を求めるプロンプトが表示されます。-Force スイッチを追加すると、確認メッセージが表示されなくなります。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

例 2: このコマンドは、指定されたバージョンのオブジェクト「sample.txt」をバケット「test-files」から削除します。これは、バケットがオブジェクトバージョンを有効にするように設定されていることを前提としています。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

例 3: このコマンドは、バケット「test-files」からオブジェクト「sample1.txt」、「sample2.txt」、「sample3.txt」を 1 回のバッチ操作で削除します。このサービスの応答では、削除の成功またはエラーのステータスを問わず、処理されたすべてのキーがリスト表示されます。このサービスで処理できなかったキーのエラーのみを取得するには、-ReportErrorsOnly パラメータを追加します (このパラメータは -Quiet というエイリアスを使用して指定することもできます)。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
"sample2.txt", "sample3.txt" )
```

例 4: この例では、-KeyCollection パラメータを指定したインライン式を使用して、削除するオブジェクトのキーを取得します。Get-S3Object は Amazon.S3.Model.S3Object インスタンスのコレクションを返します。各インスタンスには、オブジェクトを識別する文字列型の Key メンバーがあります。

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

例 5: この例では、バケット内のキープレフィックス「prefix/subprefix」を持つすべてのオブジェクトを取得して削除します。受け取るオブジェクトは一度に 1 つずつ処理されることに注意が必要です。大規模なコレクションの場合は、コレクションをコマンドレットの -InputObject (エイリアスは -S3ObjectCollection) パラメータに渡して、サービスを 1 回呼び出すだけでバッチで削除できるようにすることを検討してください。

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
Remove-S3Object -Force
```

例 6: この例では、削除マーキーを表す Amazon.S3.Model.S3ObjectVersion インスタンスのコレクションを、削除コマンドレットにパイプします。受け取るオブジェクトは一度に 1 つずつ処理されることに注意が必要です。大規模なコレクションの場合は、コレクションをコマン

ドレットの `-InputObject` (エイリアスは `-S3ObjectCollection`) パラメータに渡して、サービスを 1 回呼び出すだけでバッチで削除できるようにすることを検討してください。

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

例 7: このスクリプトは、`-KeyAndVersionCollection` パラメータで使用するオブジェクトの配列を作成することで、オブジェクトセット (この場合は削除マーカー) をバッチで削除する方法を示しています。

```
$keyVersions = @()  
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where  
  {$_ .IsDeleteMarker -eq "True"}  
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =  
  $marker.VersionId } }  
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -  
Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeleteObjects](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、バケット「test-files」からオブジェクト「sample.txt」を削除します。コマンドを実行する前に、確認を求めるプロンプトが表示されます。`-Force` スイッチを追加すると、確認メッセージが表示されなくなります。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

例 2: このコマンドは、指定されたバージョンのオブジェクト「sample.txt」をバケット「test-files」から削除します。これは、バケットがオブジェクトバージョンを有効にするように設定されていることを前提としています。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId  
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

例 3: このコマンドは、バケット「test-files」からオブジェクト「sample1.txt」、「sample2.txt」、「sample3.txt」を 1 回のバッチ操作で削除します。このサービスの応答では、削除の成功またはエラーのステータスを問わず、処理されたすべてのキーがリスト表示されます。このサービスで処理できなかったキーのエラーのみを取得するには、-

ReportErrorsOnly パラメータを追加します (このパラメータは -Quiet というエイリアスを使用して指定することもできます)。

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",  
    "sample2.txt", "sample3.txt" )
```

例 4: この例では、-KeyCollection パラメータを指定したインライン式を使用して、削除するオブジェクトのキーを取得します。Get-S3Object は Amazon.S3.Model.S3Object インスタンスのコレクションを返します。各インスタンスには、オブジェクトを識別する文字列型の Key メンバーがあります。

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
    "test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

例 5: この例では、バケット内のキープレフィックス「prefix/subprefix」を持つすべてのオブジェクトを取得して削除します。受け取るオブジェクトは一度に 1 つずつ処理されることに注意が必要です。大規模なコレクションの場合は、コレクションをコマンドレットの -InputObject (エイリアスは -S3ObjectCollection) パラメータに渡して、サービスを 1 回呼び出すだけでバッチで削除できるようにすることを検討してください。

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

例 6: この例では、削除マーカを表す Amazon.S3.Model.S3ObjectVersion インスタンスのコレクションを、削除コマンドレットにパイプします。受け取るオブジェクトは一度に 1 つずつ処理されることに注意が必要です。大規模なコレクションの場合は、コレクションをコマンドレットの -InputObject (エイリアスは -S3ObjectCollection) パラメータに渡して、サービスを 1 回呼び出すだけでバッチで削除できるようにすることを検討してください。

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

例 7: このスクリプトは、-KeyAndVersionCollection パラメータで使用するオブジェクトの配列を作成することで、オブジェクトセット (この場合は削除マーカ) をバッチで削除する方法を示しています。

```
$keyVersions = @()  
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where  
{$_ .IsDeleteMarker -eq "True"}
```

```
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =  
    $marker.VersionId } }  
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -  
Force
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[DeleteObjects](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

オブジェクト キーのリストからオブジェクトのセットを削除します。

```
class ObjectWrapper:  
    """Encapsulates S3 object actions."""  
  
    def __init__(self, s3_object):  
        """  
        :param s3_object: A Boto3 Object resource. This is a high-level resource  
        in Boto3  
                           that wraps object actions in a class-like structure.  
        """  
        self.object = s3_object  
        self.key = self.object.key  
  
    @staticmethod  
    def delete_objects(bucket, object_keys):  
        """  
        Removes a list of objects from a bucket.  
        This operation is done as a batch in a single request.  
  
        :param bucket: The bucket that contains the objects. This is a Boto3  
        Bucket  
                       resource.
```

```

:param object_keys: The list of keys that identify the objects to remove.
:return: The response that contains data about which objects were deleted
        and any that could not be deleted.
"""
try:
    response = bucket.delete_objects(
        Delete={"Objects": [{"Key": key} for key in object_keys]}
    )
    if "Deleted" in response:
        logger.info(
            "Deleted objects '%s' from bucket '%s'.",
            [del_obj["Key"] for del_obj in response["Deleted"]],
            bucket.name,
        )
    if "Errors" in response:
        logger.warning(
            "Could not delete objects '%s' from bucket '%s'.",
            [
                f"{del_obj['Key']}: {del_obj['Code']}"
                for del_obj in response["Errors"]
            ],
            bucket.name,
        )
except ClientError:
    logger.exception("Couldn't delete any objects from bucket %s.",
bucket.name)
    raise
else:
    return response

```

バケット内のすべてのオブジェクトを削除します。

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """

```



```
self.object = s3_object
self.key = self.object.key

@staticmethod
def empty_bucket(bucket):
    """
    Remove all objects from a bucket.

    :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
    """
    try:
        bucket.objects.delete()
        logger.info("Emptied bucket '%s'.", bucket.name)
    except ClientError:
        logger.exception("Couldn't empty bucket '%s'.", bucket.name)
        raise
```

すべてのバージョンを削除することによって、バージョン管理されているオブジェクトを完全に削除します。

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

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

## Rust

### SDK for Rust

#### Note

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

```
/// Delete the objects in a bucket.
pub async fn delete_objects(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    objects_to_delete: Vec<String>,
) -> Result<(), S3ExampleError> {
    // Push into a mut vector to use `?` early return errors while building
    // object keys.
    let mut delete_object_ids: Vec<aws_sdk_s3::types::ObjectIdentifier> = vec![];
    for obj in objects_to_delete {
        let obj_id = aws_sdk_s3::types::ObjectIdentifier::builder()
            .key(obj)
            .build()
            .map_err(|err| {
                S3ExampleError::new(format!("Failed to build key for
delete_object: {err:?}"))
            })?;
        delete_object_ids.push(obj_id);
    }

    client
        .delete_objects()
        .bucket(bucket_name)
        .delete(
            aws_sdk_s3::types::Delete::builder()
                .set_objects(Some(delete_object_ids))
                .build()
                .map_err(|err| {
                    S3ExampleError::new(format!("Failed to build delete_object
input {err:?}"))
                })?,
        )
}
```

```
        .send()
        .await?;
    Ok(())
}
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.
    oo_result = lo_s3->deleteobjects(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name
    io_delete = NEW /aws1/cl_s3_delete( it_objects = it_object_keys ).
    MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

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

## Swift

### SDK for Swift

#### Note

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

```
import AWSS3

public func deleteObjects(bucket: String, keys: [String]) async throws {
    let input = DeleteObjectsInput(
        bucket: bucket,
        delete: S3ClientTypes.Delete(
            objects: keys.map { S3ClientTypes.ObjectIdentifier(key: $0) },
            quiet: true
        )
    )

    do {
        _ = try await client.deleteObjects(input: input)
    } catch {
        print("ERROR: deleteObjects:", dump(error))
        throw error
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### CLI で **DeletePublicAccessBlock** を使用する

次のサンプルコードは、DeletePublicAccessBlock を使用する方法を説明しています。

## CLI

### AWS CLI

バケットのブロックパブリックアクセス設定を削除するには

次の `delete-public-access-block` の例では、指定したバケットのブロックパブリックアクセス設定を削除します。

```
aws s3api delete-public-access-block \  
  --bucket amzn-s3-demo-bucket
```

このコマンドでは何も出力されません。

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定したバケットのブロックパブリックアクセス設定をオフにします。

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

出力:

```
amzn-s3-demo-bucket
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[DeletePublicAccessBlock](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定したバケットのブロックパブリックアクセス設定をオフにします。

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

出力:

```
amzn-s3-demo-bucket
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[DeletePublicAccessBlock](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **GetBucketAccelerateConfiguration** を使用する

次のサンプルコードは、GetBucketAccelerateConfiguration を使用する方法を説明しています。

CLI

AWS CLI

バケットの加速設定を取得するには

次の get-bucket-accelerate-configuration の例では、指定したバケットの加速設定を取得します。

```
aws s3api get-bucket-accelerate-configuration \  
  --bucket amzn-s3-demo-bucket
```

出力:

```
{  
  "Status": "Enabled"  
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: 指定したバケットで転送加速設定が有効になっている場合、このコマンドは Enabled の値を返します。

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Value
-----
Enabled
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketAccelerateConfiguration](#)」を参照してください。

### Tools for PowerShell V5

例 1: 指定したバケットで転送加速設定が有効になっている場合、このコマンドは Enabled の値を返します。

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Value
-----
Enabled
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketAccelerateConfiguration](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketAc1** を使用する

次のサンプルコードは、GetBucketAc1 を使用する方法を説明しています。



アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [バケットが存在するかどうかを確認する](#)
- [アクセスコントロールリスト \(ACL\) を管理する](#)

.NET

SDK for .NET

 Note

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

```
/// <summary>
/// Get the access control list (ACL) for the new bucket.
/// </summary>
/// <param name="client">The initialized client object used to get the
/// access control list (ACL) of the bucket.</param>
/// <param name="newBucketName">The name of the newly created bucket.</
param>
/// <returns>An S3AccessControlList.</returns>
public static async Task<S3AccessControlList>
GetACLForBucketAsync(IAmazonS3 client, string newBucketName)
{
    // Retrieve bucket ACL to show that the ACL was properly applied to
    // the new bucket.
    GetACLResponse getACLResponse = await client.GetACLAsync(new
    GetACLRequest
    {
        BucketName = newBucketName,
    });

    return getACLResponse.AccessControlList;
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[GetBucketAcl](#)」を参照してください。

## C++

### SDK for C++

#### Note

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

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
        s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                   << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
```

```

        std::cout << "Type:          "
                    << getGranteeTypeString(grantee.GetType()) <<
std::endl;
    }

    if (grantee.DisplayNameHasBeenSet()) {
        std::cout << "Display name:  "
                    << grantee.GetDisplayName() << std::endl;
    }

    if (grantee.EmailAddressHasBeenSet()) {
        std::cout << "Email address: "
                    << grantee.GetEmailAddress() << std::endl;
    }

    if (grantee.IDHasBeenSet()) {
        std::cout << "ID:          "
                    << grantee.GetID() << std::endl;
    }

    if (grantee.URIHasBeenSet()) {
        std::cout << "URI:          "
                    << grantee.GetURI() << std::endl;
    }

    std::cout << "Permission:    " <<
        getPermissionString(grant.GetPermission()) <<
        std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
 */

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:

```

```

        return "Email address of an AWS account";
    case Aws::S3::Model::Type::CanonicalUser:
        return "Canonical user ID of an AWS account";
    case Aws::S3::Model::Type::Group:
        return "Predefined Amazon S3 group";
    case Aws::S3::Model::Type::NOT_SET:
        return "Not set";
    default:
        return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
    string.
/*!
    \param permission: Permission enumeration.
    \return String: Human-readable string.
*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                "objects in this bucket, and read/write this "
                "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }

    return "Permission unknown";
}

```

- API の詳細については、AWS SDK for C++ API リファレンスの「[GetBucketAcl](#)」を参照してください。

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットのアクセス制御リストを取得します。

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    }
  ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[GetBucketAcl](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;

import java.util.List;

/**
 * Before running this Java V2 code 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 GetAcl {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <bucketName> <objectKey>

        Where:
            bucketName - The Amazon S3 bucket to get the access control list
            (ACL) for.
            objectKey - The object to get the ACL for.\s
        """;
```

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

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3
     * bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be
     * retrieved.
     * @param bucketName The name of the bucket containing the object.
     * @return The ID of the grantee who has permission on the object, or an
     * empty string if an error occurs.
     */
    public static String getBucketACL(S3Client s3, String objectKey, String
    bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
```

```
        System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
        grantee = grant.grantee().id();
    }

    return grantee;
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[GetBucketAcl](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

ACL 許可を取得します。

```
import {
  GetBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Retrieves the Access Control List (ACL) for an S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
```



```
const client = new S3Client({});

try {
  const response = await client.send(
    new GetBucketAclCommand({
      Bucket: bucketName,
    }),
  );
  console.log(`ACL for bucket "${bucketName}":`);
  console.log(JSON.stringify(response, null, 2));
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while getting ACL for ${bucketName}. The bucket doesn't exist.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting ACL for ${bucketName}. ${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

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

## PowerShell

### Tools for PowerShell V5

例 1: コマンドは、S3 オブジェクトのオブジェクト所有者の詳細を取得します。

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

出力:

```
DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[GetBucketAcl](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
```

```

        logger.info(
            "Got ACL for bucket %s. Owner is %s.", self.bucket.name,
            acl.owner
        )
    except ClientError:
        logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
        raise
    else:
        return acl

```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```

TRY.
    oo_result = lo_s3->getbucketacl(          " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved bucket ACL.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **GetBucketAnalyticsConfiguration** を使用する

次のサンプルコードは、GetBucketAnalyticsConfiguration を使用する方法を説明しています。

### CLI

#### AWS CLI

特定の ID を持つバケットの分析設定を取得するには

次の get-bucket-analytics-configuration の例では、指定されたバケットと ID の分析設定を表示します。

```
aws s3api get-bucket-analytics-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 1
```

出力:

```
{
  "AnalyticsConfiguration": {
    "StorageClassAnalysis": {},
    "Id": "1"
  }
}
```

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

### PowerShell

#### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケット内の「testfilter」という名前の分析フィルターの詳細を返します。

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId 'testfilter'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketAnalyticsConfiguration](#)」を参照してください。

#### Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケット内の「testfilter」という名前の分析フィルターの詳細を返します。

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId 'testfilter'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketAnalyticsConfiguration](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

#### AWS SDK または CLI で **GetBucketCors** を使用する

次のサンプルコードは、GetBucketCors を使用方法を説明しています。

#### .NET

##### SDK for .NET

#### Note

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

```
/// <summary>  
/// Retrieve the CORS configuration applied to the Amazon S3 bucket.  
/// </summary>  
/// <param name="client">The initialized Amazon S3 client object used  
/// to retrieve the CORS configuration.</param>
```

```
/// <returns>The created CORS configuration object.</returns>
private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
{
    GetCORSConfigurationRequest request = new
GetCORSConfigurationRequest()
    {
        BucketName = BucketName,
    };
    var response = await client.GetCORSConfigurationAsync(request);
    var configuration = response.Configuration;
    PrintCORSRules(configuration);
    return configuration;
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[GetBucketCors](#)」を参照してください。

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットの Cross-Origin Resource Sharing 設定を取得します。

```
aws s3api get-bucket-cors --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "CORSRules": [
    {
      "AllowedHeaders": [
        "*"
      ],
      "ExposeHeaders": [
        "x-amz-server-side-encryption"
      ],
      "AllowedMethods": [
        "PUT",
```

```
        "POST",
        "DELETE"
    ],
    "MaxAgeSeconds": 3000,
    "AllowedOrigins": [
        "http://www.example.com"
    ]
},
{
    "AllowedHeaders": [
        "Authorization"
    ],
    "MaxAgeSeconds": 3000,
    "AllowedMethods": [
        "GET"
    ],
    "AllowedOrigins": [
        "*"
    ]
}
]
```

- API の詳細については、AWS CLI コマンドリファレンスの「[GetBucketCors](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

バケットの CORS ポリシーを取得します。

```
import {
    GetBucketCorsCommand,
    S3Client,
    S3ServiceException,
```

```
} from "@aws-sdk/client-s3";

/**
 * Log the Cross-Origin Resource Sharing (CORS) configuration information
 * set for the bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new GetBucketCorsCommand({
    Bucket: bucketName,
  });

  try {
    const { CORSRules } = await client.send(command);
    console.log(JSON.stringify(CORSRules));
    CORSRules.forEach((cr, i) => {
      console.log(
        `\\nCORSRule ${i + 1}`,
        `\\n${"-"}.repeat(10)`,
        `\\nAllowedHeaders: ${cr.AllowedHeaders}`,
        `\\nAllowedMethods: ${cr.AllowedMethods}`,
        `\\nAllowedOrigins: ${cr.AllowedOrigins}`,
        `\\nExposeHeaders: ${cr.ExposeHeaders}`,
        `\\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`,
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
}
```



```
}  
};
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in  
Boto3  
        that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def get_cors(self):  
        """  
        Get the CORS rules for the bucket.  
  
        :return The CORS rules for the specified bucket.  
        """  
        try:  
            cors = self.bucket.Cors()  
            logger.info(  

```

```
        "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
self.bucket.name
    )
    except ClientError:
        logger.exception(("Couldn't get CORS for bucket %s.",
self.bucket.name))
        raise
    else:
        return cors
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
  configuration for the bucket.
```

```

def cors
  @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
why: #{e.message}"
    nil
  end
end

end

```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```

TRY.
    oo_result = lo_s3->getbucketcors(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved bucket CORS configuration.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetBucketEncryption** を使用する

次のサンプルコードは、GetBucketEncryption を使用する方法を説明しています。

### .NET

#### SDK for .NET

##### Note

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

```
/// <summary>
/// Get and print the encryption settings of a bucket.
/// </summary>
/// <param name="bucketName">Name of the bucket.</param>
/// <returns>Async task.</returns>
public static async Task GetEncryptionSettings(string bucketName)
{
    // Check and print the bucket encryption settings.
    Console.WriteLine($"Getting encryption settings for bucket
{bucketName}.");

    try
    {
        var settings =
            await _s3Client.GetBucketEncryptionAsync(
                new GetBucketEncryptionRequest() { BucketName =
bucketName });

        foreach (var encryptionSettings in
settings?.ServerSideEncryptionConfiguration?.ServerSideEncryptionRules!)
        {
            Console.WriteLine(
                $"Algorithm:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm}");
            Console.WriteLine(
                $"Key:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionKeyManagementServi
}
        }
    }
}
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "InvalidBucketName"
                ? $"Bucket {bucketName} was not found."
                : $"Unable to get bucket encryption for bucket {bucketName},
{ex.Message}");
        }
    }
}
```

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

## CLI

### AWS CLI

バケットのサーバー側の暗号化設定を取得するには

次の `get-bucket-encryption` の例では、バケット `amzn-s3-demo-bucket` のサーバー側の暗号化設定を取得します。

```
aws s3api get-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: のコマンドは、指定したバケットに関連付けられたすべてのサーバー側暗号化ルールを返します。

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketEncryption](#)」を参照してください。

### Tools for PowerShell V5

例 1: のコマンドは、指定したバケットに関連付けられたすべてのサーバー側暗号化ルールを返します。

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketEncryption](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### CLI で **GetBucketInventoryConfiguration** を使用する

次のサンプルコードは、GetBucketInventoryConfiguration を使用する方法を説明しています。

#### CLI

##### AWS CLI

バケットのインベントリ設定を取得するには

次の get-bucket-inventory-configuration の例では、ID 1 を持つ指定したバケットのインベントリ設定を取得します。

```
aws s3api get-bucket-inventory-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --inventory-configuration-id 1
```

`--id 1`

出力:

```
{
  "InventoryConfiguration": {
    "IsEnabled": true,
    "Destination": {
      "S3BucketDestination": {
        "Format": "ORC",
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
        "AccountId": "123456789012"
      }
    },
    "IncludedObjectVersions": "Current",
    "Id": "1",
    "Schedule": {
      "Frequency": "Weekly"
    }
  }
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケット内の「testinventory」という名前のインベントリの詳細を返します。

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId 'testinventory'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketInventoryConfiguration](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケット内の「testinventory」という名前のインベントリの詳細を返します。

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId 'testinventory'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketInventoryConfiguration](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketLifecycleConfiguration** を使用する

次のサンプルコードは、GetBucketLifecycleConfiguration を使用する方法を説明しています。

.NET

SDK for .NET

 Note

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

```
/// <summary>
/// Returns a configuration object for the supplied bucket name.
/// </summary>
/// <param name="client">The S3 client object used to call
/// the GetLifecycleConfigurationAsync method.</param>
/// <param name="bucketName">The name of the S3 bucket for which a
/// configuration will be created.</param>
/// <returns>Returns a new LifecycleConfiguration object.</returns>
public static async Task<LifecycleConfiguration>
RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)
{
    var request = new GetLifecycleConfigurationRequest()
    {
        BucketName = bucketName,
```



```
};  
var response = await client.GetLifecycleConfigurationAsync(request);  
var configuration = response.Configuration;  
return configuration;  
}
```

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

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットのライフサイクル設定を取得します。

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket
```

出力:

```
{  
  "Rules": [  
    {  
      "ID": "Move rotated logs to Glacier",  
      "Prefix": "rotated/",  
      "Status": "Enabled",  
      "Transitions": [  
        {  
          "Date": "2015-11-10T00:00:00.000Z",  
          "StorageClass": "GLACIER"  
        }  
      ]  
    },  
    {  
      "Status": "Enabled",  
      "Prefix": "",  
      "NoncurrentVersionTransitions": [  
        {  
          "NoncurrentDays": 0,  
          "StorageClass": "GLACIER"  
        }  
      ]  
    }  
  ]  
}
```

```

        }
    ],
    "ID": "Move old versions to Glacier"
}
]
}

```

- API の詳細については、AWS CLI コマンドリファレンスの「[GetBucketLifecycleConfiguration](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()

```

```

        logger.info(
            "Got lifecycle rules %s for bucket '%s'.",
            config.rules,
            self.bucket.name,
        )
    except:
        logger.exception(
            "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
        )
        raise
    else:
        return config.rules

```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```

TRY.
    oo_result = lo_s3->getbucketlifecycleconf(          " oo_result is
returned for testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved bucket lifecycle configuration.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketLocation** を使用する

次のサンプルコードは、GetBucketLocation を使用する方法を説明しています。

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットの場所の制約を取得します (制約が存在する場合)。

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "LocationConstraint": "us-west-2"
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[GetBucketLocation](#)」を参照してください。

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、制約が存在する場合、バケット「amzn-s3-demo-bucket」の場所の制約を返します。

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Value
-----
ap-south-1
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketLocation](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、制約が存在する場合、バケット「amzn-s3-demo-bucket」の場所の制約を返します。

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Value
-----
ap-south-1
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[GetBucketLocation](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

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

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
```

```
        num_buckets += 1;
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint() == Some(&region) {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **GetBucketLogging** を使用する

次のサンプルコードは、GetBucketLogging を使用する方法を説明しています。

## CLI

### AWS CLI

バケットのログ記録ステータスを取得するには

次の `get-bucket-logging` の例では、指定したバケットのログ記録ステータスを取得します。

```
aws s3api get-bucket-logging \
  --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "LoggingEnabled": {
    "TargetPrefix": "",
    "TargetBucket": "amzn-s3-demo-bucket-logs"
  }
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定したバケットのログ記録ステータスを返します。

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

出力:

| TargetBucketName | Grants | TargetPrefix |
|------------------|--------|--------------|
| testbucket1      | {}     | testprefix   |

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketLogging](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、指定したバケットのログ記録ステータスを返します。

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

出力:

| TargetBucketName | Grants | TargetPrefix |
|------------------|--------|--------------|
| testbucket1      | {}     | testprefix   |

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketLogging](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **GetBucketMetricsConfiguration** を使用する

次のサンプルコードは、GetBucketMetricsConfiguration を使用方法を説明しています。

CLI

### AWS CLI

特定の ID を持つバケットのメトリクス設定を取得するには

次の get-bucket-metrics-configuration の例では、指定したバケットと ID のメトリクス設定を表示します。

```
aws s3api get-bucket-metrics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 123
```

出力:

```
{  
  "MetricsConfiguration": {  
    "Filter": {
```



```
        "Prefix": "logs"
      },
      "Id": "123"
    }
  }
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケットの「testfilter」という名前のメトリクスフィルターに関する詳細を返します。

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId 'testfilter'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketMetricsConfiguration](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケットの「testfilter」という名前のメトリクスフィルターに関する詳細を返します。

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId 'testfilter'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketMetricsConfiguration](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **GetBucketNotification** を使用する

次のサンプルコードは、GetBucketNotification を使用する方法を説明しています。

## CLI

### AWS CLI

次のコマンドは、`amzn-s3-demo-bucket` という名前のバケットの通知設定を取得します。

```
aws s3api get-bucket-notification --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "TopicConfiguration": {
    "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
    "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNL",
    "Event": "s3:ObjectCreated:*",
    "Events": [
      "s3:ObjectCreated:*"
    ]
  }
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: この例では、指定したバケットの通知設定を取得します

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
TopicConfigurations
```

出力:

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketNotification](#)」を参照してください。

## Tools for PowerShell V5

例 1: この例では、指定したバケットの通知設定を取得します

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty  
TopicConfigurations
```

出力:

```
Id      Topic  
--      -  
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketNotification](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketPolicy** を使用する

次のサンプルコードは、GetBucketPolicy を使用する方法を説明しています。

C++

SDK for C++

### Note

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

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,  
                                const Aws::S3::S3ClientConfiguration  
    &clientConfig) {  
    Aws::S3::S3Client s3Client(clientConfig);  
  
    Aws::S3::Model::GetBucketPolicyRequest request;  
    request.SetBucket(bucketName);
```

```

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
        s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
                    << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n"
<<
                policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[GetBucketPolicy](#)」を参照してください。

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットのバケットポリシーを取得します。

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "Policy": "{\"Version\":\"2008-10-17\",\"Statement\": [{\"Sid\":\"\",\"Effect\":"
  \"Allow\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":
```

```
\\"arn:aws:s3:::amzn-s3-demo-bucket/*\\"}, {\\"Sid\\":\\"\\",\\"Effect\\":\\"Deny\\",
\\"Principal\\":\\"*\\",\\"Action\\":\\"s3:GetObject\\",\\"Resource\\":\\"arn:aws:s3:::amzn-
s3-demo-bucket/secret/*\\"}]]}"
}
```

次の例は、Amazon S3 バケットポリシーをダウンロードしてファイルを変更し、put-bucket-policy を使用して変更したバケットポリシーを適用する方法を示しています。バケットポリシーをファイルにダウンロードするには、以下を実行します。

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output
text > policy.json
```

その後、必要に応じて policy.json ファイルを変更できます。最後に、次のコマンドを実行して、変更したポリシーを S3 バケットに適用することができます。

必要に応じて policy.json ファイル。最後に、次のコマンドを実行して、変更したポリシーを S3 バケットに適用することができます。

必要に応じてファイル。最後に、次のコマンドを実行して、変更したポリシーを S3 バケットに適用することができます。

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json
```

- API の詳細については、AWS CLI コマンドリファレンスの「[GetBucketPolicy](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code 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 GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            "";

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

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    /**
     * Retrieves the policy for the specified Amazon S3 bucket.
     */
}
```

```
*
* @param s3 the {@link S3Client} instance to use for making the request
* @param bucketName the name of the S3 bucket for which to retrieve the
policy
* @return the policy text for the specified bucket, or an empty string if an
error occurs
*/
public static String getPolicy(S3Client s3, String bucketName) {
    String policyText;
    System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
    GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
        policyText = policyRes.policy();
        return policyText;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[GetBucketPolicy](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

バケットポリシーを取得します。

```
import {
  GetBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Logs the policy for a specified bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Policy } = await client.send(
      new GetBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Policy for "${bucketName}":\n${Policy}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}. The bucket
        doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}.  ${caught.name}:
        ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```



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

## Kotlin

### SDK for Kotlin

#### Note

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

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケットに関連付けられたバケットポリシーを出力します。

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketPolicy](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケットに関連付けられたバケットポリシーを出力します。

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[GetBucketPolicy](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_policy(self):
        """
```

```
Get the security policy of the bucket.

:return: The security policy of the specified bucket, in JSON format.
"""
try:
    policy = self.bucket.Policy()
    logger.info(
        "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
    )
except ClientError:
    logger.exception("Couldn't get policy for bucket '%s'.",
self.bucket.name)
    raise
else:
    return json.loads(policy.policy)
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end
end
```

```
# Gets the policy of a bucket.
#
# @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
def policy
  policy = @bucket_policy.data.policy
  policy.respond_to?(:read) ? policy.read : policy
rescue Aws::Errors::ServiceError => e
  puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
#{e.message}"
  nil
end

end
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.
    oo_result = lo_s3->getbucketpolicy(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    DATA(lv_policy) = oo_result->get_policy( ).
    MESSAGE 'Retrieved bucket policy.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **GetBucketPolicyStatus** を使用する

次のサンプルコードは、GetBucketPolicyStatus を使用する方法を説明しています。

### CLI

#### AWS CLI

バケットがパブリックかどうかを示すバケットのポリシーステータスを取得するには

次の get-bucket-policy-status の例では、バケット amzn-s3-demo-bucket のポリシーステータスを取得します。

```
aws s3api get-bucket-policy-status \  
  --bucket amzn-s3-demo-bucket
```

出力:

```
{  
  "PolicyStatus": {  
    "IsPublic": false  
  }  
}
```

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

### PowerShell

#### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケットのポリシーステータスを返し、バケットがパブリックかどうかを示します。

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketPolicyStatus](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケットのポリシーステータスを返し、バケットがパブリックかどうかを示します。

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketPolicyStatus](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetBucketReplication** を使用する

次のサンプルコードは、GetBucketReplication を使用する方法を説明しています。

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットのレプリケーション設定を取得します。

```
aws s3api get-bucket-replication --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "ReplicationConfiguration": {
    "Rules": [
      {
        "Status": "Enabled",
        "Prefix": "",
        "Destination": {
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket-backup",
          "StorageClass": "STANDARD"
        }
      }
    ]
  }
}
```

```
        "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
    },
    ],
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
}
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
/**
 * Retrieves the replication details for the specified S3 bucket.
 *
 * @param s3Client      the S3 client used to interact with the S3
service
 * @param sourceBucketName the name of the S3 bucket to retrieve the
replication details for
 *
 * @throws S3Exception if there is an error retrieving the replication
details
 */
public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
    GetBucketReplicationRequest getRequest =
GetBucketReplicationRequest.builder()
        .bucket(sourceBucketName)
        .build();

    try {
        ReplicationConfiguration replicationConfig =
s3Client.getBucketReplication(getRequest).replicationConfiguration();
    }
```

```
ReplicationRule rule = replicationConfig.rules().get(0);
System.out.println("Retrieved destination bucket: " +
rule.destination().bucket());
System.out.println("Retrieved priority: " + rule.priority());
System.out.println("Retrieved source-bucket replication rule status:
" + rule.status());

    } catch (S3Exception e) {
        System.err.println("Failed to retrieve replication details: " +
e.awsErrorDetails().errorMessage());
    }
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: 「amzn-s3-demo-bucket」という名前のバケットに設定されているレプリケーション設定の情報を返します。

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketReplication](#)」を参照してください。

### Tools for PowerShell V5

例 1: 「amzn-s3-demo-bucket」という名前のバケットに設定されているレプリケーション設定の情報を返します。

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketReplication](#)」を参照してください。



AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で `GetBucketRequestPayment` を使用する

次のサンプルコードは、`GetBucketRequestPayment` を使用する方法を説明しています。

### CLI

#### AWS CLI

バケットのリクエストの支払い設定を取得するには

次の `get-bucket-request-payment` の例では、指定したバケットのリクエスト支払いの設定を取得します。

```
aws s3api get-bucket-request-payment \
  --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "Payer": "BucketOwner"
}
```

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

### PowerShell

#### Tools for PowerShell V4

例 1: 「`amzn-s3-demo-bucket`」という名前のバケットのリクエストの支払い設定を返します。デフォルトでは、バケット所有者はバケットからのダウンロード料金を支払います。

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketRequestPayment](#)」を参照してください。

## Tools for PowerShell V5

例 1: 「amzn-s3-demo-bucket」という名前のバケットのリクエストの支払い設定を返します。デフォルトでは、バケット所有者はバケットからのダウンロード料金を支払います。

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketRequestPayment](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **GetBucketTagging** を使用する

次のサンプルコードは、GetBucketTagging を使用する方法を説明しています。

CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットのタグ付け設定を取得します。

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "TagSet": [
    {
      "Value": "marketing",
      "Key": "organization"
    }
  ]
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定バケットに関連付けられたすべてのタグを返します。

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketTagging](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定バケットに関連付けられたすべてのタグを返します。

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketTagging](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **GetBucketVersioning** を使用する

次のサンプルコードは、GetBucketVersioning を使用方法を説明しています。

### CLI

#### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットのバージョニング設定を取得します。

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

出力:

```
{
```

```
"Status": "Enabled"
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定バケットに関するバージョニングステータスを返します。

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketVersioning](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定バケットに関するバージョニングステータスを返します。

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketVersioning](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetBucketWebsite** を使用する

次のサンプルコードは、GetBucketWebsite を使用する方法を説明しています。

## .NET

### SDK for .NET

#### Note

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

```
// Get the website configuration.
GetBucketWebsiteRequest getRequest = new
GetBucketWebsiteRequest()
{
    BucketName = bucketName,
};
GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[GetBucketWebsite](#)」を参照してください。

## C++

### SDK for C++

#### Note

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

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
```

```
const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
        s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                    << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult =
outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                    << std::endl << std::endl
                    << "For bucket '" << bucketName << "':"
                    << std::endl
                    << "Index page : "
                    << websiteResult.GetIndexDocument().GetSuffix()
                    << std::endl
                    << "Error page: "
                    << websiteResult.GetErrorDocument().GetKey()
                    << std::endl;
    }

    return outcome.IsSuccess();
}
```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[GetBucketWebsite](#)」を参照してください。

## CLI

### AWS CLI

次のコマンドは、`amzn-s3-demo-bucket` という名前のバケットの静的ウェブサイト設定を取得します。

```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[GetBucketWebsite](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

ウェブサイト設定を取得します。

```
import {
  GetBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
```

```
} from "@aws-sdk/client-s3";

/**
 * Log the website configuration for a bucket.
 * @param {{ bucketName }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    console.log(
      `Your bucket is set up to host a website with the following configuration:
\n${JSON.stringify(response, null, 2)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchWebsiteConfiguration"
    ) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}. The
bucket isn't configured as a website.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}.
${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

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



## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定 S3 バケットの静的ウェブサイトの設定の詳細を返します。

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetBucketWebsite](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定 S3 バケットの静的ウェブサイトの設定の詳細を返します。

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetBucketWebsite](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **GetObject** を使用する

次のサンプルコードは、GetObject を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [バケットが変更されている場合、バケットからオブジェクトを取得する](#)
- [マルチリージョンアクセスポイント からオブジェクトを取得する](#)
- [暗号化の開始方法](#)
- [オブジェクトストレージの使用開始](#)
- [条件付きリクエストの実行](#)
- [アップロードとダウンロードを追跡する](#)

## .NET

### SDK for .NET (v4)

#### Note

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

```
/// <summary>
/// Shows how to download an object from an Amazon S3 bucket to the
/// local computer.
/// </summary>
/// <param name="bucketName">The name of the bucket where the object is
/// currently stored.</param>
/// <param name="objectName">The name of the object to download.</param>
/// <param name="filePath">The path, including filename, where the
/// downloaded object will be stored.</param>
/// <returns>A boolean value indicating the success or failure of the
/// download process.</returns>
public async Task<bool> DownloadObjectFromBucketAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    var request = new GetObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
    };

    using GetObjectResponse response = await
        _amazonS3.GetObjectAsync(request);

    try
    {
        // Save object to local file
        await response.WriteResponseStreamToFileAsync($"{filePath}\
        \{objectName}", true, CancellationToken.None);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

```
    }  
    catch (AmazonS3Exception ex)  
    {  
        Console.WriteLine($"Error saving {objectName}: {ex.Message}");  
        return false;  
    }  
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[GetObject](#)」を参照してください。

SDK for .NET

 Note

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

条件付きリクエストを使用してオブジェクトを取得します。

```
/// <summary>  
/// Retrieves an object from Amazon S3 with a conditional request.  
/// </summary>  
/// <param name="objectKey">The key of the object to retrieve.</param>  
/// <param name="sourceBucket">The source bucket of the object.</param>  
/// <param name="conditionType">The type of condition: 'IfMatch',  
'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>  
/// <param name="conditionDateValue">The value to use for the condition for  
dates.</param>  
/// <param name="etagConditionalValue">The value to use for the condition for  
etags.</param>  
/// <returns>True if the conditional read is successful, False otherwise.</  
returns>  
public async Task<bool> GetObjectConditional(string objectKey, string  
sourceBucket,  
    S3ConditionType conditionType, DateTime? conditionDateValue = null,  
string? etagConditionalValue = null)  
{  
    try  
    {
```

```
var getObjectRequest = new GetObjectRequest
{
    BucketName = sourceBucket,
    Key = objectKey
};

switch (conditionType)
{
    case S3ConditionType.IfMatch:
        getObjectRequest.EtagToMatch = etagConditionalValue;
        break;
    case S3ConditionType.IfNoneMatch:
        getObjectRequest.EtagToNotMatch = etagConditionalValue;
        break;
    case S3ConditionType.IfModifiedSince:
        getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
        break;
    case S3ConditionType.IfUnmodifiedSince:
        getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
        break;
    default:
        throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
}

var response = await _amazonS3.GetObjectAsync(getObjectRequest);
var sampleBytes = new byte[20];
await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
_logger.LogInformation($"Conditional read
successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
return true;
}
catch (AmazonS3Exception e)
{
    if (e.ErrorCode == "PreconditionFailed")
    {
        _logger.LogError("Conditional read failed: Precondition failed");
    }
    else if (e.ErrorCode == "NotModified")
    {
        _logger.LogError("Conditional read failed: Object not modified");
    }
}
```

```

    }
    else
    {
        _logger.LogError($"Unexpected error: {e.ErrorCode}");
        throw;
    }
    return false;
}
}

```

- API の詳細については、AWS SDK for .NET API リファレンスの「[GetObject](#)」を参照してください。

## Bash

### AWS CLIBash スクリプトを使用した

#### Note

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

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#     $1 - The name of the bucket to download the object from.
#     $2 - The path and file name to store the downloaded bucket.

```

```
#      $3 - The key (name) of the object in the bucket.
#
# Returns:
#      0 - If successful.
#      1 - If it fails.
#####
function download_object_from_bucket() {
    local bucket_name=$1
    local destination_file_name=$2
    local object_name=$3
    local response

    response=$(aws s3api get-object \
        --bucket "$bucket_name" \
        --key "$object_name" \
        "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}
```

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

## C++

### SDK for C++

#### Note

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

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
```

```
Aws::S3::S3Client client(clientConfig);

Aws::S3::Model::GetObjectRequest request;
request.SetBucket(fromBucket);
request.SetKey(objectKey);

Aws::S3::Model::GetObjectOutcome outcome =
    client.GetObject(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: getObject: " <<
        err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
} else {
    std::cout << "Successfully retrieved '" << objectKey << "' from '"
        << fromBucket << "'." << std::endl;
}

return outcome.IsSuccess();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[GetObject](#)」を参照してください。

## CLI

### AWS CLI

次の例は `get-object` コマンドを使用して、Amazon S3 からオブジェクトをダウンロードします。

```
aws s3api get-object --bucket text-content --key dir/  
my_images.tar.bz2 my_images.tar.bz2
```

outfile パラメータは、「--outfile」などのオプション名なしで指定されていることに注意してください。出力ファイルの名前はコマンドの最後のパラメータでなければなりません。

次の例は `--range` を使用して、オブジェクトから特定のバイト範囲をダウンロードする方法を示しています。バイト範囲には「bytes=」というプレフィックスを付ける必要があることに注意してください。

```
aws s3api get-object --bucket text-content --key dir/my_data --  
range bytes=8888-9999 my_data_range
```

オブジェクトの取得の詳細については、「Amazon S3 ユーザーガイド」の「オブジェクトを取得する」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[GetObject](#)」を参照してください。

Go

SDK for Go V2

 Note

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

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.
```



```
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
                objectKey, bucketName)
            err = noKey
        } else {
            log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
                objectKey, err)
        }
        return err
    }
    defer result.Body.Close()
    file, err := os.Create(fileName)
    if err != nil {
        log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
        return err
    }
    defer file.Close()
    body, err := io.ReadAll(result.Body)
    if err != nil {
        log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
            err)
    }
    _, err = file.Write(body)
    return err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

[S3Client](#) を使用して、データをバイト配列として読み取ります。

```
/**
 * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
 * and writes them to a local file.
 *
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName     the key (or name) of the S3 object to retrieve
 * @param path        the local file path where the object's bytes will be
 * written
 * @return a {@link CompletableFuture} that completes when the object bytes
 * have been written to the local file
 */
public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

    CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
    return response.thenAccept(objectBytes -> {
        try {
            byte[] data = objectBytes.asByteArray();
            Path filePath = Paths.get(path);
```

```
        Files.write(filePath, data);
        logger.info("Successfully obtained bytes from an S3 object");
    } catch (IOException ex) {
        throw new RuntimeException("Failed to write data to file", ex);
    }
}).whenComplete((resp, ex) -> {
    if (ex != null) {
        throw new RuntimeException("Failed to get object bytes from S3",
ex);
    }
});
}
```

[S3TransferManager](#) を使用して、ローカルファイルに S3 バケットの [オブジェクトをダウンロード](#) します。 [完全なファイル](#) と [テスト](#) を表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String
bucketName,

                                String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFilePath))
```

```

        .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentType());
        return downloadResult.response().contentType();
    }

```

[S3Client](#) を使用して、オブジェクトに属するタグを読み取ります。

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;

import java.util.List;

/**
 * Before running this Java V2 code 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 GetObjectTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:

```

```
        bucketName - The Amazon S3 bucket name.\s
        keyName - A key name that represents the object.\s
        """;

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

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listTags(s3, bucketName, keyName);
    s3.close();
}

/**
 * Lists the tags associated with an Amazon S3 object.
 *
 * @param s3 the S3Client object used to interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket that contains the object
 * @param keyName the key (name) of the S3 object
 */
public static void listTags(S3Client s3, String bucketName, String keyName) {
    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        GetObjectTaggingResponse tags =
s3.getObjectTagging(getTaggingRequest);
        List<Tag> tagSet = tags.tagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.key());
            System.out.println(tag.value());
        }

    } catch (S3Exception e) {
```

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

[S3Client](#) を使用してオブジェクトの URL を取得します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URL;

/**
 * Before running this Java V2 code 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 GetObjectUrl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
            """;

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

```
String bucketName = args[0];
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

getUrl(s3, bucketName, keyName);
s3.close();
}

/**
 * Retrieves the URL for a specific object in an Amazon S3 bucket.
 *
 * @param s3 the S3Client object used to interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket where the object is stored
 * @param keyName the name of the object for which the URL should be
retrieved
 * @throws S3Exception if there is an error retrieving the URL for the
specified object
 */
public static void getUrl(S3Client s3, String bucketName, String keyName) {
    try {
        GetUrlRequest request = GetUrlRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        URL url = s3.utilities().getUrl(request);
        System.out.println("The URL for " + keyName + " is " + url);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

[S3Client](#) を使用する S3Presigner クライアントオブジェクトを使用してオブジェクトを取得します。

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code 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 GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

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



```
String bucketName = args[0];
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Presigner presigner = S3Presigner.builder()
    .region(region)
    .build();

getPresignedUrl(presigner, bucketName, keyName);
presigner.close();
}

/**
 * Generates a pre-signed URL for an Amazon S3 object.
 *
 * @param presigner The {@link S3Presigner} instance to use for generating
the pre-signed URL.
 * @param bucketName The name of the Amazon S3 bucket where the object is
stored.
 * @param keyName The key name (file name) of the object in the Amazon S3
bucket.
 *
 * @throws S3Exception If there is an error interacting with the Amazon S3
service.
 * @throws IOException If there is an error opening the HTTP connection or
reading/writing the request/response.
 */
public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
    try {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(60))
            .getObjectRequest(getObjectRequest)
            .build();

        PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
        String theUrl = presignedGetObjectRequest.url().toString();
        System.out.println("Presigned URL: " + theUrl);
    }
}
```

```

        HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
        presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
            values.forEach(value -> {
                connection.addRequestProperty(header, value);
            });
        });

        // Send any request payload that the service needs (not needed when
        // isBrowserExecutable is true).
        if (presignedGetObjectRequest.signedPayload().isPresent()) {
            connection.setDoOutput(true);

            try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
                OutputStream httpOutputStream =
connection.getOutputStream()) {
                IoUtils.copy(signedPayload, httpOutputStream);
            }
        }

        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            System.out.println("Service returned response: ");
            IoUtils.copy(content, System.out);
        }
    } catch (S3Exception | IOException e) {
        e.printStackTrace();
    }
}
}

```

ResponseTransformer オブジェクトと [S3Client](#) を使用してオブジェクトを取得します。

```

import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;

```

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code 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 GetObjectData {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            "";

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
```

```
        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    /**
     * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves
     * them to a local file.
     *
     * @param s3 The S3Client instance used to interact with the Amazon S3
     * service.
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param keyName The key (or name) of the S3 object.
     * @param path The local file path where the object's bytes will be saved.
     * @throws IOException If an I/O error occurs while writing the bytes to the
     * local file.
     * @throws S3Exception If an error occurs while retrieving the object from
     * the S3 bucket.
     */
    public static void getObjectBytes(S3Client s3, String bucketName, String
    keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}  
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[GetObject](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

オブジェクトをダウンロードします。

```
import {  
  GetObjectCommand,  
  NoSuchKey,  
  S3Client,  
  S3ServiceException,  
} from "@aws-sdk/client-s3";  
  
/**  
 * Get a single object from a specified S3 bucket.  
 * @param {{ bucketName: string, key: string }}  
 */  
export const main = async ({ bucketName, key }) => {  
  const client = new S3Client({});  
  
  try {  
    const response = await client.send(  
      new GetObjectCommand({  
        Bucket: bucketName,  
        Key: key,  
      })),  
    );  
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'  
    methods.
```

```
const str = await response.Body.transformToString();
console.log(str);
} catch (caught) {
  if (caught instanceof NoSuchKey) {
    console.error(
      `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

ETag が指定されたものと一致することを条件に、オブジェクトをダウンロードします。

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
```

```
    }),
  );
  // The Body object also has 'transformToByteArray' and 'transformToWebStream'
  methods.
  const str = await response.Body.transformToString();
  console.log("Success. Here is text of the file:", str);
} catch (caught) {
  if (caught instanceof NoSuchKey) {
    console.error(
      `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utills/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  },
```

```
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

ETag が指定されたものと一致しないことを条件に、オブジェクトをダウンロードします。

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfNoneMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
  }
```



```
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utills/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
```

```
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

特定の期間に作成または変更されたことを条件に、オブジェクトをダウンロードします。

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfModifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
```

```
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
```

```
    console.error(errors.join("\n"));
  }
}
```

特定の期間に作成または変更されていないことを条件に、オブジェクトをダウンロードします。

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfUnmodifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
        such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {

```

```
        console.error(
            `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
            ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
        key: {
            type: "string",
            required: true,
        },
    };
    const results = parseArgs({ options });
    const { errors } = validateArgs({ options }, results);
    return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。

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

## Kotlin

### SDK for Kotlin

#### Note

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

```
suspend fun getObjectBytes(  
    bucketName: String,  
    keyName: String,  
    path: String,  
) {  
    val request =  
        GetObjectRequest {  
            key = keyName  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.getObject(request) { resp ->  
            val myFile = File(path)  
            resp.body?.writeToFile(myFile)  
            println("Successfully read $keyName from $bucketName")  
        }  
    }  
}
```

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

## PHP

### SDK for PHP

#### Note

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

オブジェクトを取得します。

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $file = $this->s3client->getObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、バケット「amzn-s3-demo-bucket」から項目「sample.txt」を取得して、それを現在の場所にある「local-sample.txt」という名前のファイルに保存します。このコマンドを呼び出す前にファイル「local-sample.txt」が存在していなくても構いません。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

例 2: このコマンドは、バケット「amzn-s3-demo-bucket」から仮想ディレクトリ「DIR」を取得し、それを現在の場所の「Local-DIR」という名前のフォルダに保存します。このコマンドを呼び出す前に、フォルダー「Local-DIR」が存在していなくても構いません。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

例 3: バケット名に「config」を含むバケットから、キーが「.json」で終わるすべてのオブジェクトを、指定したフォルダ内のファイルにダウンロードします。オブジェクトキーはファイル名の設定に使用されます。

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetObject](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、バケット「amzn-s3-demo-bucket」から項目「sample.txt」を取得して、それを現在の場所にある「local-sample.txt」という名前のファイルに保存します。このコマンドを呼び出す前にファイル「local-sample.txt」が存在していなくても構いません。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

例 2: このコマンドは、バケット「amzn-s3-demo-bucket」から仮想ディレクトリ「DIR」を取得し、それを現在の場所の「Local-DIR」という名前のフォルダに保存します。このコマンドを呼び出す前に、フォルダー「Local-DIR」が存在していなくても構いません。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

例 3: バケット名に「config」を含むバケットから、キーが「.json」で終わるすべてのオブジェクトを、指定したフォルダ内のファイルにダウンロードします。オブジェクトキーはファイル名の設定に使用されます。



```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

例 4: マルチパート並列ダウンロードを使用して 1 つのオブジェクトをダウンロードし、大きなファイルのパフォーマンスを向上させます。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt -UseMultipartDownload
```

例 5: カスタム 64 MB のパートサイズで RANGE モードを使用して大きなオブジェクトをダウンロードします。RANGE モードは HTTP バイト範囲リクエストを使用し、アップロード方法に関係なくすべてのオブジェクトで機能します。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

例 6: マルチパートダウンロードと同時ファイルダウンロードを使用して、キープレフィックス「data/」を持つすべてのオブジェクトをローカルフォルダにダウンロードします。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

例 7: マルチパートダウンロードと 20 の同時接続を使用して単一のオブジェクトをダウンロードし、高帯域幅ネットワークの最大スループットを実現します。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

例 8: マルチパートダウンロードを使用して、キープレフィックス「data/」を持つすべてのオブジェクトをローカルフォルダにダウンロードし、中止するのではなく、個々のファイルの失敗をスキップして続行します。

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[GetObject](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get(self):
        """
        Gets the object.

        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
```

```
except ClientError:
    logger.exception(
        "Couldn't get object '%s' from bucket '%s'.",
        self.object.key,
        self.object.bucket_name,
    )
    raise
else:
    return body
```

条件付きリクエストを使用してオブジェクトを取得します。

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
```

```
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
        'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
    try:
        response = self.s3.get_object(
            Bucket=source_bucket,
            Key=object_key,
            **{condition_type: condition_value},
        )
        sample_bytes = response["Body"].read(20)
        print(
            f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
        )
        print(f"\t{sample_bytes}")
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional read failed: Precondition failed")
        elif error_code == "304": # Not modified error code.
            print("\tConditional read failed: Object not modified")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise
```

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

## Ruby

### SDK for Ruby

#### Note

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

オブジェクトを取得します。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is
  # downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
  # successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes) downloaded to
  #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

オブジェクトを取得し、サーバー側の暗号化状態をレポートします。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
  # successful; otherwise nil.
  def object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
    object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
    obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

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

## Rust

### SDK for Rust

#### Note

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

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, S3ExampleError> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone()).map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to initialize file for saving S3 download: {err:?}"
        ))
    })?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await.map_err(|err| {
        S3ExampleError::new(format!("Failed to read from S3 download stream:
{err:?}"))
    })? {
        let bytes_len = bytes.len();
        file.write_all(&bytes).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to write from S3 download stream to local file: {err:?}"
            ))
        })?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }
}
```

```
Ok(byte_count)
}
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.
    oo_result = lo_s3->getobject(          " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key ).
    DATA(lv_object_data) = oo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

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



## Swift

### SDK for Swift

#### Note

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

```
import AWSS3

public func downloadFile(bucket: String, key: String, to: String) async
throws {
    let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        try data.write(to: fileUrl)
    }
    catch {
        print("ERROR: ", dump(error, name: "Downloading a file."))
        throw error
    }
}
```

```
import AWSS3
```

```
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        return data
    }
    catch {
        print("ERROR: ", dump(error, name: "Reading a file. "))
        throw error
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetObjectAcl** を使用する

次のサンプルコードは、GetObjectAcl を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アクセスコントロールリスト \(ACL\) を管理する](#)

C++

SDK for C++

 Note

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

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
        s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
                    << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            std::cout << "For object " << objectKey << ": "
                        << std::endl << std::endl;

            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            if (grantee.TypeHasBeenSet()) {
```

```

        std::cout << "Type:          "
                    << getGranteeTypeString(grantee.GetType()) <<
std::endl;
    }

    if (grantee.DisplayNameHasBeenSet()) {
        std::cout << "Display name:  "
                    << grantee.GetDisplayName() << std::endl;
    }

    if (grantee.EmailAddressHasBeenSet()) {
        std::cout << "Email address: "
                    << grantee.GetEmailAddress() << std::endl;
    }

    if (grantee.IDHasBeenSet()) {
        std::cout << "ID:          "
                    << grantee.GetID() << std::endl;
    }

    if (grantee.URIHasBeenSet()) {
        std::cout << "URI:          "
                    << grantee.GetURI() << std::endl;
    }

    std::cout << "Permission:    " <<
        getPermissionString(grant.GetPermission()) <<
        std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
 */
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
    }
}

```

```

        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
    string.
    /*!
    \param permission: Permission enumeration.
    \return String: Human-readable string
    */
    Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
        switch (permission) {
            case Aws::S3::Model::Permission::FULL_CONTROL:
                return "Can read this object's data and its metadata, "
                    "and read/write this object's permissions";
            case Aws::S3::Model::Permission::NOT_SET:
                return "Permission not set";
            case Aws::S3::Model::Permission::READ:
                return "Can read this object's data and its metadata";
            case Aws::S3::Model::Permission::READ_ACP:
                return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
            case Aws::S3::Model::Permission::WRITE_ACP:
                return "Can write this object's permissions";
            default:
                return "Permission unknown";
        }
    }
}

```

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

## CLI

## AWS CLI

次のコマンドは、`amzn-s3-demo-bucket` という名前のバケット内のオブジェクトのアクセス制御リストを取得します。

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key index.html
```

出力:

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    },
    {
      "Grantee": {
        "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
      },
      "Permission": "READ"
    }
  ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[GetObjectAcl](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

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

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

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

## PowerShell

### Tools for PowerShell V5

例 1: コマンドは、S3 オブジェクトのオブジェクト所有者の詳細を取得します。

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
*).Owner
```

出力:

```
DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[GetObjectAcl](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
```



```

        logger.info(
            "Got ACL for object %s owned by %s.",
            self.object.key,
            acl.owner["DisplayName"],
        )
    except ClientError:
        logger.exception("Couldn't get ACL for object %s.", self.object.key)
        raise
    else:
        return acl

```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```

TRY.
    oo_result = lo_s3->getobjectacl(          " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key ).
    MESSAGE 'Retrieved object ACL.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetObjectAttributes** を使用する

次のサンプルコードは、GetObjectAttributes を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon S3 オブジェクトの整合性の操作](#)

C++

SDK for C++

 Note

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

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of
the object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
&key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
```

```

        Aws::String &hashData,
        std::vector<Aws::String> *partHashes,
        const Aws::S3::S3Client &client) {

    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break; // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:

```

```

        hashData = result.GetChecksum().GetChecksumSHA1();
        break;
    case AwsDoc::S3::SHA256:
        hashData = result.GetChecksum().GetChecksumSHA256();
        break;
    case AwsDoc::S3::CRC32:
        hashData = result.GetChecksum().GetChecksumCRC32();
        break;
    case AwsDoc::S3::CRC32C:
        hashData = result.GetChecksum().GetChecksumCRC32C();
        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        return false;
    }
} else {
    std::cerr << "Error retrieving object checksum attributes." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

if (nullptr != partHashes) {
    attributes.clear();
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
    request.SetObjectAttributes(attributes);
    outcome = client.GetObjectAttributes(request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
        for (const Aws::S3::Model::ObjectPart &part: parts) {
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                    break;
                case AwsDoc::S3::MD5: // MD5 is not supported.
                    break;
                case AwsDoc::S3::SHA1:
                    partHashes->push_back(part.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:
                    partHashes->push_back(part.GetChecksumSHA256());
                    break;
            }
        }
    }
}

```

```

        case AwsDoc::S3::CRC32:
            partHashes->push_back(part.GetChecksumCRC32());
            break;
        case AwsDoc::S3::CRC32C:
            partHashes->push_back(part.GetChecksumCRC32C());
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            return false;
    }
}
} else {
    std::cerr << "Error retrieving object attributes for object
parts." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
}
}
return true;
}

```

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

## CLI

### AWS CLI

オブジェクト自体を返さずにオブジェクトからメタデータを取得するには

次の get-object-attributes の例は、オブジェクト doc1.rtf からメタデータを取得します。

```

aws s3api get-object-attributes \
    --bucket amzn-s3-demo-bucket \
    --key doc1.rtf \
    --object-attributes "StorageClass" "ETag" "ObjectSize"

```

出力:

```
{
  "LastModified": "2022-03-15T19:37:31+00:00",
  "VersionId": "IuCPjXTDzHNf1dAuitVBIKJpF2p1fg4P",
  "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",
  "StorageClass": "STANDARD",
  "ObjectSize": 405
}
```

詳細については、「Amazon S3 API リファレンス」の [GetObjectAttributes](#) を参照してください。

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetObjectLegalHold** を使用する

次のサンプルコードは、GetObjectLegalHold を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon S3 オブジェクトをロックする](#)

.NET

SDK for .NET

#### Note

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

```
/// <summary>
/// Get the legal hold details for an S3 object.
```

```
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object legal hold details.</returns>
public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectLegalHoldAsync(request);
        Console.WriteLine($"\\tObject legal hold for {objectKey} in
{bucketName}: " +
            $"\\n\\tStatus: {response.LegalHold.Status}");
        return response.LegalHold;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch legal hold: '{ex.Message}'");
        return new ObjectLockLegalHold();
    }
}
```

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

## CLI

### AWS CLI

オブジェクトのリーガルホールドステータスを取得する

次の get-object-legal-hold の例では、指定したオブジェクトのリーガルホールドステータスを取得します。

```
aws s3api get-object-legal-hold \
    --bucket amzn-s3-demo-bucket-with-object-lock \
```

```
--key doc1.rtf
```

出力:

```
{
  "LegalHold": {
    "Status": "ON"
  }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[GetObjectLegalHold](#)」を参照してください。

Go

SDK for Go V2

 Note

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

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
```



```
S3Client  *s3.Client
S3Manager *manager.Uploader
}

// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
    var status *types.ObjectLockLegalHoldStatus
    input := &s3.GetObjectLegalHoldInput{
        Bucket:    aws.String(bucket),
        Key:        aws.String(key),
        VersionId: aws.String(versionId),
    }

    output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
    if err != nil {
        var noSuchKeyErr *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noSuchKeyErr) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noSuchKeyErr
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "NoSuchObjectLockConfiguration":
                log.Printf("Object %s does not have an object lock configuration.\n", key)
                err = nil
            case "InvalidRequest":
                log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
                err = nil
            }
        }
    } else {
        status = &output.LegalHold.Status
    }

    return status, err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
    }

    return null;
}
```

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

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

```
import {
  GetObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get an object's current legal hold status.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectLegalHoldCommand({
        Bucket: bucketName,
        Key: key,
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the
        bucket>",
        // VersionId: "<the specific version id of the object to check>",
      }),
    );
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
```

```
        `Error from S3 while getting legal hold status for ${key} in
        ${bucketName}. The bucket doesn't exist.` ,
    );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting legal hold status for ${key} in
            ${bucketName} from ${bucketName}.  ${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
        key: {
            type: "string",
            required: true,
        },
    };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
```

```
}
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

オブジェクトにリーガルホールドを適用します。

```
def get_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Get the legal hold status of a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to get the legal hold status of.
    """
    print()
    logger.info("Getting legal hold status of file [%s] in bucket [%s]", key,
                bucket)
    try:
        response = s3_client.get_object_legal_hold(Bucket=bucket, Key=key)
        legal_hold_status = response["LegalHold"]["Status"]
        logger.debug(
            "Legal hold status of file [%s] in bucket [%s] is [%s]",
            key,
            bucket,
            legal_hold_status,
        )
    except Exception as e:
        logger.error(
            "Failed to get legal hold status of file [%s] in bucket [%s]: %s",
```

```

        key,
        bucket,
        e,
    )

```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```

TRY.
    oo_result = lo_s3->getobjectlegalhold(          " oo_result is returned
for testing purposes. "
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key ).
    MESSAGE 'Retrieved object legal hold status.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **GetObjectLockConfiguration** を使用する

次のサンプルコードは、GetObjectLockConfiguration を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon S3 オブジェクトをロックする](#)

### .NET

#### SDK for .NET

##### Note

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

```
/// <summary>
/// Get the object lock configuration details for an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to get details.</param>
/// <returns>The bucket's object lock configuration details.</returns>
public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
{
    try
    {
        var request = new GetObjectLockConfigurationRequest()
        {
            BucketName = bucketName
        };

        var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
        Console.WriteLine($"Bucket object lock config for {bucketName} in
{bucketName}: " +
            $"Enabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
            $"Rule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");
    }
}
```

```
        return response.ObjectLockConfiguration;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to fetch object lock config:
'{ex.Message}'");
        return new ObjectLockConfiguration();
    }
}
```

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

## CLI

### AWS CLI

バケットのオブジェクトロック設定を取得するには

次の `get-object-lock-configuration` 例では、指定されたバケットのオブジェクトロック設定を取得します。

```
aws s3api get-object-lock-configuration \
    --bucket amzn-s3-demo-bucket-with-object-lock
```

出力:

```
{
  "ObjectLockConfiguration": {
    "ObjectLockEnabled": "Enabled",
    "Rule": {
      "DefaultRetention": {
        "Mode": "COMPLIANCE",
        "Days": 50
      }
    }
  }
}
```



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

Go

SDK for Go V2

 Note

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

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectLockConfiguration retrieves the object lock configuration for an S3  
// bucket.  
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket  
    string) (*types.ObjectLockConfiguration, error) {
```

```
var lockConfig *types.ObjectLockConfiguration
input := &s3.GetObjectLockConfigurationInput{
    Bucket: aws.String(bucket),
}

output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
"ObjectLockConfigurationNotFoundError" {
        log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
        err = nil
    }
} else {
    lockConfig = output.ObjectLockConfiguration
}

return lockConfig, err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
```

```
GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .build();

GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
System.out.println("Bucket object lock config for "+bucketName+": ");
System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().getObjectLockEnabled());
System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[GetObjectLockConfiguration](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

```
import {
  GetObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Gets the Object Lock configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
```

```
try {
  const { ObjectLockConfiguration } = await client.send(
    new GetObjectLockConfigurationCommand({
      Bucket: bucketName,
      // Optionally, you can provide additional parameters
      // ExpectedBucketOwner: "<account ID that is expected to own the
bucket>",
    }),
  );
  console.log(
    `Object Lock Configuration:\n${JSON.stringify(ObjectLockConfiguration)}`,
  );
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while getting object lock configuration for ${bucketName}.
The bucket doesn't exist.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting object lock configuration for ${bucketName}.
${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
```

```
    },  
    };  
    const results = parseArgs({ options });  
    const { errors } = validateArgs({ options }, results);  
    return { errors, results };  
};  
  
if (isMain(import.meta.url)) {  
    const { errors, results } = loadArgs();  
    if (!errors) {  
        main(results.values);  
    } else {  
        console.error(errors.join("\n"));  
    }  
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: 指定した S3 バケットのオブジェクトロック設定が有効になっている場合、このコマンドは値「Enabled」を返します。

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

出力:

```
Value  
-----  
Enabled
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetObjectLockConfiguration](#)」を参照してください。

## Tools for PowerShell V5

例 1: 指定した S3 バケットのオブジェクトロック設定が有効になっている場合、このコマンドは値「Enabled」を返します。

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

出力:

```
Value  
-----  
Enabled
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[GetObjectLockConfiguration](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

オブジェクトロック設定を取得します。

```
def is_object_lock_enabled(s3_client, bucket: str) -> bool:  
    """  
    Check if object lock is enabled for a bucket.  
  
    Args:  
        s3_client: Boto3 S3 client.  
        bucket: The name of the bucket to check.  
  
    Returns:  
        True if object lock is enabled, False otherwise.  
    """
```

```

try:
    response = s3_client.get_object_lock_configuration(Bucket=bucket)
    return (
        "ObjectLockConfiguration" in response
        and response["ObjectLockConfiguration"]["ObjectLockEnabled"] ==
"Enabled"
    )
except s3_client.exceptions.ClientError as e:
    if e.response["Error"]["Code"] == "ObjectLockConfigurationNotFoundError":
        return False
    else:
        raise

```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```

TRY.
    oo_result = lo_s3->getobjectlockconfiguration(          " oo_result is
returned for testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved object lock configuration.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **GetObjectRetention** を使用する

次のサンプルコードは、GetObjectRetention を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon S3 オブジェクトをロックする](#)

.NET

SDK for .NET

 Note

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

```
/// <summary>
/// Get the retention period for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object retention details.</returns>
public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectRetentionAsync(request);
```



```
        Console.WriteLine($"{\tObject retention for {objectKey} in
{bucketName}: " +
                                $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}."");
        return response.Retention;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"{\tUnable to fetch object lock retention:
'{ex.Message}'");
        return new ObjectLockRetention();
    }
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[GetObjectRetention](#)」を参照してください。

## CLI

### AWS CLI

オブジェクトのオブジェクト保持設定を取得するには

次の `get-object-retention` の例では、指定されたオブジェクトの保持設定を取得します。

```
aws s3api get-object-retention \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf
```

出力:

```
{
  "Retention": {
    "Mode": "GOVERNANCE",
    "RetainUntilDate": "2025-01-01T00:00:00.000Z"
  }
}
```

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

## Go

## SDK for Go V2

 Note

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

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectRetention retrieves the object retention configuration for an S3  
// object.  
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key  
string) (*types.ObjectLockRetention, error) {  
    var retention *types.ObjectLockRetention  
    input := &s3.GetObjectRetentionInput{  
        Bucket: aws.String(bucket),  
        Key:    aws.String(key),  
    }
```

```
}

output, err := actor.S3Client.GetObjectRetention(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have locking enabled.", bucket)
            err = nil
        }
    }
} else {
    retention = output.Retention
}

return retention, err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
// Get the retention period for an S3 object.
```

```
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
        GetObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectRetentionResponse response =
        getClient().getObjectRetention(retentionRequest);
        System.out.println("Object retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[GetObjectRetention](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

```
import {
    GetObjectRetentionCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
/**
 * Log the "RetainUntilDate" for an object in an S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const { Retention } = await client.send(
      new GetObjectRetentionCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    console.log(
      `${key} in ${bucketName} will be retained until
    ${Retention.RetainUntilDate}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchObjectLockConfiguration"
    ) {
      console.warn(
        `The object "${key}" in the bucket "${bucketName}" does not have an
    ObjectLock configuration.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object retention settings for
    "${bucketName}".  ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
```

```
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、オブジェクトが保持されるまでのモードと日付を返します。

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetObjectRetention](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、オブジェクトが保持されるまでのモードと日付を返します。

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetObjectRetention](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **GetObjectTagging** を使用する

次のサンプルコードは、GetObjectTagging を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [タグの使用開始](#)

## CLI

### AWS CLI

オブジェクトにアタッチされたタグを取得するには

次の get-object-tagging 指定したオブジェクトから指定したキーの値を取得します。

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc1.rtf
```

出力:

```
{  
  "TagSet": [  
    {  
      "Value": "confidential",
```

```
        "Key": "designation"
      }
    ]
  }
}
```

次の `get-object-tagging` の例では、タグのないオブジェクト `doc2.rtf` のタグセットの取得を試行します。

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc2.rtf
```

出力:

```
{
  "TagSet": []
}
```

次の `get-object-tagging` の例では、複数のタグがあるオブジェクト `doc3.rtf` のタグセットの取得を試行します。

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc3.rtf
```

出力:

```
{
  "TagSet": [
    {
      "Value": "confidential",
      "Key": "designation"
    },
    {
      "Value": "finance",
      "Key": "department"
    },
    {
      "Value": "payroll",
      "Key": "team"
    }
  ]
}
```



```
]
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: この例では、指定した S3 バケット上に存在するオブジェクトに関連付けられたタグを返します。

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Key  Value
---  -
test value
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetObjectTagging](#)」を参照してください。

### Tools for PowerShell V5

例 1: この例では、指定した S3 バケット上に存在するオブジェクトに関連付けられたタグを返します。

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

出力:

```
Key  Value
---  -
test value
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetObjectTagging](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **GetPublicAccessBlock** を使用する

次のサンプルコードは、GetPublicAccessBlock を使用する方法を説明しています。

### CLI

#### AWS CLI

バケットのブロックパブリックアクセス設定を指定または削除するには

次の get-public-access-block の例では、指定したバケットのブロックパブリックアクセス設定を表示します。

```
aws s3api get-public-access-block \  
  --bucket amzn-s3-demo-bucket
```

出力:

```
{  
  "PublicAccessBlockConfiguration": {  
    "IgnorePublicAcls": true,  
    "BlockPublicPolicy": true,  
    "BlockPublicAcls": true,  
    "RestrictPublicBuckets": true  
  }  
}
```

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

### PowerShell

#### Tools for PowerShell V4

例 1: このコマンドは、指定 S3 バケットのパブリックアクセスブロック設定を返します。

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[GetPublicAccessBlock](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、指定 S3 バケットのパブリックアクセスブロック設定を返します。

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[GetPublicAccessBlock](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **HeadBucket** を使用する

次のサンプルコードは、HeadBucket を使用する方法を説明しています。

### Bash

#### AWS CLIBash スクリプトを使用した

#### Note

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

```
#####  
# function bucket_exists  
#  
# This function checks to see if the specified bucket already exists.  
#  
# Parameters:  
#     $1 - The name of the bucket to check.  
#  
# Returns:  
#     0 - If the bucket already exists.
```

```
#      1 - If the bucket doesn't exist.
#####
function bucket_exists() {
    local bucket_name
    bucket_name=$1

    # Check whether the bucket already exists.
    # We suppress all output - we're interested only in the return code.

    if aws s3api head-bucket \
        --bucket "$bucket_name" \
        >/dev/null 2>&1; then
        return 0 # 0 in Bash script means true.
    else
        return 1 # 1 in Bash script means false.
    fi
}
```

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

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットへのアクセスを確認します。

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

バケットが存在し、そのバケットにアクセスできる場合、出力は返されません。それ以外の場合は、エラーメッセージが表示されます。例えば、次のようになります。

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

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

## Go

## SDK for Go V2

 Note

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

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}  
  
// BucketExists checks whether a bucket exists in the current account.
```

```
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
(bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
    exists := true
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
                err = nil
            default:
                log.Printf("Either you don't have access to bucket %v or another error
occurred. "+
                    "Here's what happened: %v\n", bucketName, err)
            }
        }
    } else {
        log.Printf("Bucket %v exists and you already own it.", bucketName)
    }

    return exists, err
}
```

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

## PowerShell

### Tools for PowerShell V5

例 1: このコマンドは、ユーザーがバケットにアクセスするアクセス許可を持っている場合に、既存のバケットに対して HTTP ステータスコード 200 OK の出力を返します。BucketArn パラメータは S3 ディレクトリバケットでのみサポートされています。

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

出力:

```
AccessPointAlias    : False
BucketArn           :
BucketLocationName  :
BucketLocationType  :
BucketRegion        : us-east-2
ResponseMetadata    : Amazon.Runtime.ResponseMetadata
ContentLength        : 0
HttpStatusCode       : OK
```

例 2: このコマンドは、存在しないバケットに対して HTTP ステータスコード NotFound でエラーをスローします。

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

出力:

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status
Code NotFound. No further error information was returned by the service.
```

例 3: このコマンドは、ユーザーがアクセス許可を持たない既存のバケットに対して HTTP ステータスコード Forbidden でエラーをスローします。

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

出力:

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status
Code Forbidden. No further error information was returned by the service.
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[HeadBucket](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
                self.bucket.name,
            )
            exists = False
        return exists
```



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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.  
    oo_result = lo_s3->headbucket(          " oo_result is returned for  
testing purposes. "  
    iv_bucket = iv_bucket_name ).  
    MESSAGE 'Bucket exists and you have access to it.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **HeadObject** を使用する

次のサンプルコードは、HeadObject を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [オブジェクトが存在するかどうかを確認します。](#)
- [オブジェクトストレージの使用開始](#)

## CLI

### AWS CLI

次のコマンドは、`amzn-s3-demo-bucket` という名前のバケット内のオブジェクトのメタデータを取得します。

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

出力:

```
{
  "AcceptRanges": "bytes",
  "ContentType": "text/html",
  "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
  "ContentLength": 77,
  "VersionId": "null",
  "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
  "Metadata": {}
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[HeadObject](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

### オブジェクトのコンテンツタイプを調べる

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
```

```
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code 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 GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
            "";

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

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the content type of an object stored in an Amazon S3 bucket.
     *
     * @param s3 an instance of the {@link S3Client} class, which is used to
     * interact with the Amazon S3 service
     */
}
```

```
* @param bucketName the name of the S3 bucket where the object is stored
* @param keyName the key (file name) of the object in the S3 bucket
*/
public static void getContentType(S3Client s3, String bucketName, String
keyName) {
    try {
        HeadObjectRequest objectRequest = HeadObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        HeadObjectResponse objectHead = s3.headObject(objectRequest);
        String type = objectHead.contentType();
        System.out.println("The object content type is " + type);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

## オブジェクトの復元ステータスを取得する

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;
```

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

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Checks the restoration status of an Amazon S3 object.
     *
     * @param s3          an instance of the {@link S3Client} class used to
     *                    interact with the Amazon S3 service
     * @param bucketName  the name of the Amazon S3 bucket where the object is
     *                    stored
     * @param keyName     the name of the Amazon S3 object to be checked
     * @throws S3Exception if an error occurs while interacting with the Amazon
     *                    S3 service
     */
    public static void checkStatus(S3Client s3, String bucketName, String
keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
}  
}
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
require 'aws-sdk-s3'  
  
# Wraps Amazon S3 object actions.  
class ObjectExistsWrapper  
  attr_reader :object  
  
  # @param object [Aws::S3::Object] An Amazon S3 object.  
  def initialize(object)  
    @object = object  
  end  
  
  # Checks whether the object exists.  
  #  
  # @return [Boolean] True if the object exists; otherwise false.  
  def exists?  
    @object.exists?  
    rescue Aws::Errors::ServiceError => e  
      puts "Couldn't check existence of object  
#{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"  
      false  
    end  
  end  
end  
  
# Example usage:
```

```
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **ListBucketAnalyticsConfigurations** を使用する

次のサンプルコードは、ListBucketAnalyticsConfigurations を使用する方法を説明しています。

CLI

AWS CLI

バケットの分析設定のリストを取得するには

次の list-bucket-analytics-configurations の例では、指定したバケットの分析設定のリストを取得します。

```
aws s3api list-bucket-analytics-configurations \
  --bucket amzn-s3-demo-bucket
```

出力:

```
{
```

```
"AnalyticsConfigurationList": [  
  {  
    "StorageClassAnalysis": {},  
    "Id": "1"  
  }  
],  
"IsTruncated": false  
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケットの最初の 100 つの分析設定を返します。

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ListBucketAnalyticsConfigurations](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケットの最初の 100 つの分析設定を返します。

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[ListBucketAnalyticsConfigurations](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **ListBucketInventoryConfigurations** を使用する

次のサンプルコードは、ListBucketInventoryConfigurations を使用する方法を説明しています。



## CLI

## AWS CLI

バケットのインベントリ設定のリストを取得するには

次の `list-bucket-inventory-configurations` の例では、指定したバケットのインベントリ設定をリスト表示します。

```
aws s3api list-bucket-inventory-configurations \
  --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "InventoryConfigurationList": [
    {
      "IsEnabled": true,
      "Destination": {
        "S3BucketDestination": {
          "Format": "ORC",
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
          "AccountId": "123456789012"
        }
      },
      "IncludedObjectVersions": "Current",
      "Id": "1",
      "Schedule": {
        "Frequency": "Weekly"
      }
    },
    {
      "IsEnabled": true,
      "Destination": {
        "S3BucketDestination": {
          "Format": "CSV",
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
          "AccountId": "123456789012"
        }
      },
      "IncludedObjectVersions": "Current",
      "Id": "2",
      "Schedule": {
```

```
        "Frequency": "Daily"
    }
}
],
"IsTruncated": false
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケットの最初の 100 つのインベントリ設定を返します。

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ListBucketInventoryConfigurations](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケットの最初の 100 つのインベントリ設定を返します。

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[ListBucketInventoryConfigurations](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListBuckets** を使用する

次のサンプルコードは、ListBuckets を使用する方法を説明しています。

## .NET

### SDK for .NET

#### Note

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

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
    /// Service (Amazon S3) buckets belonging to the default account.
    /// </summary>
    public class ListBuckets
    {
        private static IAmazonS3 _s3Client;

        /// <summary>
        /// Get a list of the buckets owned by the default user.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <returns>The response from the ListingBuckets call that contains a
        /// list of the buckets owned by the default user.</returns>
        public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3
client)
        {
            return await client.ListBucketsAsync();
        }

        /// <summary>
        /// This method lists the name and creation date for the buckets in
        /// the passed List of S3 buckets.
        /// </summary>
    }
}
```

```
/// <param name="bucketList">A List of S3 bucket objects.</param>
public static void DisplayBucketList(List<S3Bucket> bucketList)
{
    bucketList
        .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
created on: {b.CreationDate}"));
}

public static async Task Main()
{
    // The client uses the AWS Region of the default user.
    // If the Region where the buckets were created is different,
    // pass the Region to the client constructor. For example:
    // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
    _s3Client = new AmazonS3Client();
    var response = await GetBuckets(_s3Client);
    DisplayBucketList(response.Buckets);
}
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[ListBuckets](#)」を参照してください。

## C++

### SDK for C++

#### Note

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

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();
```

```
bool result = true;
if (!outcome.IsSuccess()) {
    std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
    result = false;
} else {
    std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
buckets\n";
    for (auto &&b: outcome.GetResult().GetBuckets()) {
        std::cout << b.GetName() << std::endl;
    }
}

return result;
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[ListBuckets](#)」を参照してください。

## CLI

### AWS CLI

次のコマンドは、list-buckets コマンドを使用してすべての Amazon S3 バケット (すべてのリージョン) の名前を表示します。

```
aws s3api list-buckets --query "Buckets[].Name"
```

クエリオプションで list-buckets の出力をバケット名のみにフィルタリングします。

バケットの詳細については、「Amazon S3 ユーザーガイド」の「バケットの使用」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[ListBuckets](#)」を参照してください。

## Go

## SDK for Go V2

 Note

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

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}  
  
// ListBuckets lists the buckets in the current account.
```

```
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
error) {
    var err error
    var output *s3.ListBucketsOutput
    var buckets []types.Bucket
    bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
&s3.ListBucketsInput{})
    for bucketPaginator.HasMorePages() {
        output, err = bucketPaginator.NextPage(ctx)
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
                fmt.Println("You don't have permission to list buckets for this account.")
                err = apiErr
            } else {
                log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
            }
            break
        } else {
            buckets = append(buckets, output.Buckets...)
        }
    }
    return buckets, err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListBucketsIterable;
/**
 * 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 ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);
    }

    /**
     * Lists all the S3 buckets available in the current AWS account.
     *
     * @param s3 The {@link S3Client} instance to use for interacting with the
     * Amazon S3 service.
     */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
            System.out.println("Bucket Name: " + bucket.name()));
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[ListBuckets](#)」を参照してください。



## JavaScript

### SDK for JavaScript (v3)

#### Note

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

バケットを一覧表示します。

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the Amazon S3 buckets in your account.
 */
export const main = async () => {
  const client = new S3Client({});
  /** @type {?import('@aws-sdk/client-s3').Owner} */
  let Owner = null;

  /** @type {import('@aws-sdk/client-s3').Bucket[]} */
  const Buckets = [];

  try {
    const paginator = paginateListBuckets({ client }, {});

    for await (const page of paginator) {
      if (!Owner) {
        Owner = page.Owner;
      }

      Buckets.push(...page.Buckets);
    }

    console.log(
      `${Owner.DisplayName} owns ${Buckets.length} bucket${
        Buckets.length === 1 ? "" : "s"
      }`
    );
  }
}
```

```
    }:` ,  
  );  
  console.log(`${Buckets.map((b) => ` • ${b.Name}` ).join("\n")}`);  
} catch (caught) {  
  if (caught instanceof S3ServiceException) {  
    console.error(  
      `Error from S3 while listing buckets.  ${caught.name}:  
${caught.message}` ,  
    );  
  } else {  
    throw caught;  
  }  
}  
};
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドはすべての S3 バケットを返します。

```
Get-S3Bucket
```

例 2: このコマンドは「test-files」という名前のバケットを返します。

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[ListBuckets](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドはすべての S3 バケットを返します。

```
Get-S3Bucket
```

例 2: このコマンドは「test-files」という名前のバケットを返します。

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[ListBuckets](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    @staticmethod
    def list(s3_resource):
        """
        Get the buckets in all Regions for the current account.

        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
        Boto3
                        that contains collections and factory methods to
        create
                        other high-level S3 sub-resources.
        :return: The list of buckets.
        """
        try:
```

```
        buckets = list(s3_resource.buckets.all())
        logger.info("Got buckets: %s.", buckets)
    except ClientError:
        logger.exception("Couldn't get buckets.")
        raise
    else:
        return buckets
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
require 'aws-sdk-s3'

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts 'Found these buckets:'
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
    end
  end
end
```

```
        count -= 1
        break if count.zero?
      end
      true
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't list buckets. Here's why: #{e.message}"
      false
    end
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

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

## Rust

### SDK for Rust

#### Note

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

```
async fn show_buckets(
  strict: bool,
  client: &Client,
  region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
  let mut buckets = client.list_buckets().into_paginator().send();

  let mut num_buckets = 0;
  let mut in_region = 0;
```

```
while let Some(Ok(output)) = buckets.next().await {
    for bucket in output.buckets() {
        num_buckets += 1;
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint() == Some(&region) {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }
}

println!();
if strict {
    println!(
        "Found {} buckets in the {} region out of a total of {} buckets.",
        in_region, region, num_buckets
    );
} else {
    println!("Found {} buckets in all regions.", num_buckets);
}

Ok(())
}
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.  
    oo_result = lo_s3->listbuckets(          " oo_result is returned for  
testing purposes. "  
    ).  
    DATA(lv_bucket_count) = lines( oo_result->get_buckets( ) ).  
    MESSAGE |Retrieved { lv_bucket_count } buckets in all regions.| TYPE 'I'.  
CATCH /aws1/cx_rt_generic.  
    MESSAGE 'Unable to list buckets.' TYPE 'E'.  
ENDTRY.
```

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

## Swift

### SDK for Swift

#### Note

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

```
import AWSS3  
  
/// Return an array containing information about every available bucket.  
///  
/// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing  
/// each bucket.
```

```
public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {  
    return try await client.listBuckets(input: ListBucketsInput())  
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListMultipartUploads** を使用する

次のサンプルコードは、ListMultipartUploads を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [不完全なマルチパートアップロードを特定する](#)

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットのすべてのアクティブなマルチパートアップロードを一覧表示します。

```
aws s3api list-multipart-uploads --bucket amzn-s3-demo-bucket
```

出力:

```
{  
  "Uploads": [  
    {  
      "Initiator": {  
        "DisplayName": "username",  
        "ID": "arn:aws:iam::0123456789012:user/username"  
      },  
      "Initiated": "2015-06-02T18:01:30.000Z",
```



```

        "UploadId":
        "dfRtDYU0WWCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
        "StorageClass": "STANDARD",
        "Key": "multipart/01",
        "Owner": {
            "DisplayName": "aws-account-name",
            "ID":
            "100719349fc3b6dcd7c820a124bf7aec408092c3d7b51b38494939801fc248b"
        }
    },
    "CommonPrefixes": []
}

```

進行中のマルチパートアップロードでは、Amazon S3 のストレージコストが発生します。アクティブなマルチパートアップロードを完了または中止して、その一部をアカウントから削除します。

- API の詳細については、AWS CLI コマンドリファレンスの「[ListMultipartUploads](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
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 ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
            """;

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

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the multipart uploads currently in progress in the specified Amazon
S3 bucket.
     *
     * @param s3 the S3Client object used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket to list the multipart
uploads for
     */
    public static void listUploads(S3Client s3, String bucketName) {
        try {

```

```
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

        ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            System.out.println("Upload in progress: Key = \"" + upload.key()
+ "\", id = " + upload.uploadId());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListObjectVersions** を使用する

次のサンプルコードは、ListObjectVersions を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [オブジェクトストレージの使用開始](#)
- [バージョン管理されたオブジェクトを操作する](#)

## .NET

### SDK for .NET

#### Note

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

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example lists the versions of the objects in a version enabled
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class ListObjectVersions
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region where your bucket is defined is different from
        // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
        // for the AWS Region to the client constructor like this:
        // var client = new AmazonS3Client(RegionEndpoint.USWest2);
        IAmazonS3 client = new AmazonS3Client();
        await GetObjectListWithAllVersionsAsync(client, bucketName);
    }

    /// <summary>
    /// This method lists all versions of the objects within an Amazon S3
    /// version enabled bucket.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// ListVersionsAsync.</param>
    /// <param name="bucketName">The name of the version enabled Amazon S3
bucket
```

```
    /// for which you want to list the versions of the contained objects.</
param>
    public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
client, string bucketName)
    {
        try
        {
            // When you instantiate the ListVersionRequest, you can
            // optionally specify a key name prefix in the request
            // if you want a list of object versions of a specific object.

            // For this example we set a small limit in MaxKeys to return
            // a small list of versions.
            ListVersionsRequest request = new ListVersionsRequest()
            {
                BucketName = bucketName,
                MaxKeys = 2,
            };

            do
            {
                ListVersionsResponse response = await
client.ListVersionsAsync(request);

                // Process response.
                foreach (S3ObjectVersion entry in response.Versions)
                {
                    Console.WriteLine($"key: {entry.Key} size:
{entry.Size}");
                }

                // If response is truncated, set the marker to get the next
                // set of keys.
                if (response.IsTruncated)
                {
                    request.KeyMarker = response.NextKeyMarker;
                    request.VersionIdMarker = response.NextVersionIdMarker;
                }
                else
                {
                    request = null;
                }
            }
            while (request != null);
        }
    }
}
```

```
    }  
    catch (AmazonS3Exception ex)  
    {  
        Console.WriteLine($"Error: '{ex.Message}'");  
    }  
}  
}
```

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

## CLI

### AWS CLI

次のコマンドは、`amzn-s3-demo-bucket` という名前のバケット内のオブジェクトのバージョン情報を取得します。

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix index.html
```

出力:

```
{  
  "DeleteMarkers": [  
    {  
      "Owner": {  
        "DisplayName": "my-username",  
        "ID":  
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"  
      },  
      "IsLatest": true,  
      "VersionId": "B2VsEK5saUNNHKc0AJj7hIE86RozToyq",  
      "Key": "index.html",  
      "LastModified": "2015-11-10T00:57:03.000Z"  
    },  
    {  
      "Owner": {  
        "DisplayName": "my-username",  
        "ID":  
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"  
      }  
    }  
  ]  
}
```

```

    },
    "IsLatest": false,
    "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
    "Key": "index.html",
    "LastModified": "2015-11-09T23:32:20.000Z"
  }
],
"Versions": [
  {
    "LastModified": "2015-11-10T00:20:11.000Z",
    "VersionId": "Rb_l2T8UHDkFEwCgJjhlGPOZC0qJ.vpD",
    "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
    "StorageClass": "STANDARD",
    "Key": "index.html",
    "Owner": {
      "DisplayName": "my-username",
      "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
    },
    "IsLatest": false,
    "Size": 38
  },
  {
    "LastModified": "2015-11-09T23:26:41.000Z",
    "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAff",
    "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
    "StorageClass": "STANDARD",
    "Key": "index.html",
    "Owner": {
      "DisplayName": "my-username",
      "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
    },
    "IsLatest": false,
    "Size": 38
  },
  {
    "LastModified": "2015-11-09T22:50:50.000Z",
    "VersionId": "null",
    "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
    "StorageClass": "STANDARD",
    "Key": "index.html",
    "Owner": {
      "DisplayName": "my-username",

```

```
        "ID":  
        "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"  
    },  
    "IsLatest": false,  
    "Size": 533823  
}  
]  
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[ListObjectVersions](#)」を参照してください。

Go

SDK for Go V2

 Note

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

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client *s3.Client
```



```
S3Manager *manager.Uploader
}

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
    var versions []types.ObjectVersion
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
    for versionPaginator.HasMorePages() {
        output, err = versionPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
            break
        } else {
            versions = append(versions, output.Versions...)
        }
    }
    return versions, err
}
```

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

## Rust

### SDK for Rust

#### Note

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

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
        println!(" version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
}
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.
    oo_result = lo_s3->listobjectversions(          " oo_result is returned
for testing purposes. "
    iv_bucket = iv_bucket_name
```

```
iv_prefix = iv_prefix ).
MESSAGE 'Retrieved object versions.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **ListObjects** を使用する

次のサンプルコードは、ListObjects を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon S3 オブジェクトを一覧表示するウェブページを作成する](#)

## CLI

### AWS CLI

次の例は、list-objects コマンドを使用して、指定されたバケット内のすべてのオブジェクトの名前を表示します。

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

この例では、--query 引数を使用して list-objects の出力を各オブジェクトのキー値とサイズでフィルタリングしています。

オブジェクトの詳細については、「Amazon S3 デベロッパーガイド」の「Working with Amazon S3 Objects」を参照してください。

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、バケット「test-files」内のすべての項目に関する情報を取得します。

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

例 2: このコマンドは、バケット「test-files」内の項目「sample.txt」に関する情報を取得します。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

例 3: このコマンドは、バケット「test-files」からプレフィックス「sample」を持つすべての項目に関する情報を取得します。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- API の詳細については、「AWS Tools for PowerShell Cmdlet Reference(V4)」の「[ListObjects](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、バケット「test-files」内のすべての項目に関する情報を取得します。

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

例 2: このコマンドは、バケット「test-files」内の項目「sample.txt」に関する情報を取得します。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

例 3: このコマンドは、バケット「test-files」からプレフィックス「sample」を持つすべての項目に関する情報を取得します。

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[ListObjects](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **ListObjectsV2** を使用する

次のサンプルコードは、ListObjectsV2 を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [バケット内のすべてのオブジェクトを削除する](#)
- [オブジェクトストレージの使用開始](#)

.NET

SDK for .NET (v4)

 Note

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

```
/// <summary>
/// Shows how to list the objects in an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket for which to list.
/// <param name="printList">True to print out the list.
/// <returns>The collection of objects.</returns>
public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
bool printList = true)
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
```

```
};

if (printList)
{
    Console.WriteLine("-----");
    Console.WriteLine($"Listing the contents of {bucketName}:");
    Console.WriteLine("-----");
}

var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
{
    BucketName = bucketName,
});
var s3Objects = new List<S3Object>();
await foreach (var response in listObjectsV2Paginator.Responses)
{
    if (response.S3Objects != null)
    {
        s3Objects.AddRange(response.S3Objects);
    }
}

if (printList)
{
    Console.WriteLine($"Number of Objects: {s3Objects.Count}");
    foreach (var entry in s3Objects)
    {
        Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
    }
}

return s3Objects;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' getting list of objects.");
    return null;
}
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[ListObjectsV2](#)」を参照してください。

## SDK for .NET

### Note

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

ページネーターを使用してオブジェクトを一覧表示します。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// The following example lists objects in an Amazon Simple Storage
/// Service (Amazon S3) bucket.
/// </summary>
public class ListObjectsPaginator
{
    private const string BucketName = "amzn-s3-demo-bucket";

    public static async Task Main()
    {
        IAmazonS3 s3Client = new AmazonS3Client();

        Console.WriteLine($"Listing the objects contained in {BucketName}:
\n");

        await ListingObjectsAsync(s3Client, BucketName);
    }

    /// <summary>
    /// This method uses a paginator to retrieve the list of objects in an
    /// an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">An Amazon S3 client object.</param>
    /// <param name="bucketName">The name of the S3 bucket whose objects
    /// you want to list.</param>
}
```

```

        public static async Task ListingObjectsAsync(IAmazonS3 client, string
bucketName)
        {
            var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
ListObjectsV2Request
            {
                BucketName = bucketName,
            });

            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
                Console.WriteLine($"Number of Keys: {response.KeyCount}");
                foreach (var entry in response.S3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }
        }
    }
}

```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[ListObjectsV2](#)」を参照してください。

## Bash

### AWS CLIBash スクリプトを使用した

#### Note

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

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####

```



```

function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     The list of files in text format.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function list_items_in_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api list-objects \
        --bucket "$bucket_name" \
        --output text \
        --query 'Contents[].{Key: Key, Size: Size}')

    # shellcheck disable=SC2181
    if [[ ${?} -eq 0 ]]; then
        echo "$response"
    else
        errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
        return 1
    fi
}

```

- APIの詳細については、「AWS CLI API リファレンス」の「[ListObjectsV2](#)」を参照してください。

## C++

## SDK for C++

 Note

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

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());
}
```

```
std::cout << allObjects.size() << " object(s) found:" << std::endl;

for (const auto &object: allObjects) {
    std::cout << " " << object.GetKey() << std::endl;
    keysResult.push_back(object.GetKey());
}

return true;
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[ListObjectsV2](#)」を参照してください。

## CLI

### AWS CLI

バケット内のオブジェクトのリストを取得するには

次の list-objects-v2 の例では、指定したバケットのオブジェクトを一覧表示します。

```
aws s3api list-objects-v2 \
  --bucket amzn-s3-demo-bucket
```

出力:

```
{
  "Contents": [
    {
      "LastModified": "2019-11-05T23:11:50.000Z",
      "ETag": "\"621503c373607d548b37cff8778d992c\"",
      "StorageClass": "STANDARD",
      "Key": "doc1.rtf",
      "Size": 391
    },
    {
      "LastModified": "2019-11-05T23:11:50.000Z",
      "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",
      "StorageClass": "STANDARD",
      "Key": "doc2.rtf",
      "Size": 373
    }
  ]
}
```

```
    },
    {
      "LastModified": "2019-11-05T23:11:50.000Z",
      "ETag": "\"08210852f65a2e9cb999972539a64d68\"",
      "StorageClass": "STANDARD",
      "Key": "doc3.rtf",
      "Size": 399
    },
    {
      "LastModified": "2019-11-05T23:11:50.000Z",
      "ETag": "\"d1852dd683f404306569471af106988e\"",
      "StorageClass": "STANDARD",
      "Key": "doc4.rtf",
      "Size": 6225
    }
  ]
}
```

- APIの詳細については、「AWS CLI API リファレンス」の「[ListObjectsV2](#)」を参照してください。

Go

SDK for Go V2

 Note

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

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "io"
    "log"
    "os"
    "time"
```

```
"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
([]types.Object, error) {
    var err error
    var output *s3.ListObjectsV2Output
    input := &s3.ListObjectsV2Input{
        Bucket: aws.String(bucketName),
    }
    var objects []types.Object
    objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
    for objectPaginator.HasMorePages() {
        output, err = objectPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
            break
        } else {
            objects = append(objects, output.Contents...)
        }
    }
    return objects, err
}
```

```
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の「[ListObjectsV2](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

```
/**
 * Asynchronously lists all objects in the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket to list objects for
 * @return a {@link CompletableFuture} that completes when all objects have
 * been listed
 */
public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
    ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    ListObjectsV2Publisher paginator =
        getAsyncClient().listObjectsV2Paginator(initialRequest);
    return paginator.subscribe(response -> {
        response.contents().forEach(s3object -> {
            logger.info("Object key: " + s3object.key());
        });
    }).thenRun(() -> {
        logger.info("Successfully listed all objects in the bucket: " +
            bucketName);
    }).exceptionally(ex -> {
```

```
        throw new RuntimeException("Failed to list objects", ex);
    });
}
```

ページネーションを使用してオブジェクトを一覧表示します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
            """;

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

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the objects in the specified S3 bucket.
     */
}
```

```
* @param s3 the S3Client instance used to interact with Amazon S3
* @param bucketName the name of the S3 bucket to list the objects from
*/
public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
        listRes.stream()
            .flatMap(r -> r.contents().stream())
            .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[ListObjectsV2](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

バケット内のすべてのオブジェクトを一覧表示します。オブジェクトが複数ある場合は、IsTruncated と NextContinuationToken を使用してリスト全体を繰り返し処理します。

```
import {
```



```

    S3Client,
    S3ServiceException,
    // This command supersedes the ListObjectsCommand and is the recommended way to
    list objects.
    paginateListObjectsV2,
} from "@aws-sdk/client-s3";

/**
 * Log all of the object keys in a bucket.
 * @param {{ bucketName: string, pageSize: string }}
 */
export const main = async ({ bucketName, pageSize }) => {
    const client = new S3Client({});
    /** @type {string[][]} */
    const objects = [];
    try {
        const paginator = paginateListObjectsV2(
            { client, /* Max items per page */ pageSize: Number.parseInt(pageSize) },
            { Bucket: bucketName },
        );

        for await (const page of paginator) {
            objects.push(page.Contents.map((o) => o.Key));
        }
        objects.forEach((objectList, pageNum) => {
            console.log(
                `Page ${pageNum + 1}\n-----\n${objectList.map((o) => `•
${o}`).join("\n")}\n`,
            );
        });
    } catch (caught) {
        if (
            caught instanceof S3ServiceException &&
            caught.name === "NoSuchBucket"
        ) {
            console.error(
                `Error from S3 while listing objects for "${bucketName}". The bucket
doesn't exist.`,
            );
        } else if (caught instanceof S3ServiceException) {
            console.error(
                `Error from S3 while listing objects for "${bucketName}".
${caught.name}: ${caught.message}`,
            );
        }
    }
};

```

```
    } else {  
        throw caught;  
    }  
}  
};
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の「[ListObjectsV2](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

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

```
suspend fun listBucketObjects(bucketName: String) {  
    val request =  
        ListObjectsRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        val response = s3.listObjects(request)  
        response.contents?.forEach { myObject ->  
            println("The name of the key is ${myObject.key}")  
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")  
            println("The owner is ${myObject.owner}")  
        }  
    }  
}  
  
private fun calKb(intValue: Long): Long = intValue / 1024
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[ListObjectsV2](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

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

バケット内のオブジェクトを一覧表示します。

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $contents = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with listing objects before continuing.");
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[ListObjectsV2](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                                that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix
        are listed.
        :return: The list of objects.
        """
        try:
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
                "Got objects %s from bucket '%s'", [o.key for o in objects],
                bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.",
                bucket.name)
            raise
        else:
            return objects
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[ListObjectsV2](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

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

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
    #{e.message}"
    0
  end
end

# Example usage:
```

```
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ListObjectsV2](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

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

```
pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
Result<(), S3ExampleError> {
  let mut response = client
    .list_objects_v2()
    .bucket(bucket.to_owned())
    .max_keys(10) // In this example, go 10 at a time.
    .into_paginator()
    .send();

  while let Some(result) = response.next().await {
    match result {
      Ok(output) => {
        for object in output.contents() {
          println!(" - {}", object.key().unwrap_or("Unknown"));
        }
      }
      Err(err) => {
```

```
        eprintln!("{err:?}")
    }
}
}

Ok(())
}
```

- API の詳細については、「AWS SDK for Rust API リファレンス」の「[ListObjectsV2](#)」を参照してください。

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.
    oo_result = lo_s3->listobjectsv2(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- API の詳細については、「AWS SDK for SAP ABAP API リファレンス」の「[ListObjectsV2](#)」を参照してください。

## Swift

### SDK for Swift

#### Note

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

```
import AWSS3

public func listBucketFiles(bucket: String) async throws -> [String] {
    do {
        let input = ListObjectsV2Input(
            bucket: bucket
        )

        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'continuationToken' in
        "ListObjectsV2Output".
        let output = client.listObjectsV2Paginated(input: input)
        var names: [String] = []

        for try await page in output {
            guard let objList = page.contents else {
                print("ERROR: listObjectsV2Paginated returned nil contents.")
                continue
            }

            for obj in objList {
                if let objName = obj.key {
                    names.append(objName)
                }
            }
        }

        return names
    }
    catch {
        print("ERROR: ", dump(error, name: "Listing objects."))
    }
}
```



```
        throw error;
    }
}
```

- API の詳細については、「AWS SDK for Swift API リファレンス」の「[ListObjectsV2](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketAccelerateConfiguration** を使用する

次のサンプルコードは、PutBucketAccelerateConfiguration を使用する方法を説明しています。

.NET

SDK for .NET

 Note

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

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
```

```
/// The main method initializes the client object and sets the
/// Amazon Simple Storage Service (Amazon S3) bucket name before
/// calling EnableAccelerationAsync.
/// </summary>
public static async Task Main()
{
    var s3Client = new AmazonS3Client();
    const string bucketName = "amzn-s3-demo-bucket";

    await EnableAccelerationAsync(s3Client, bucketName);
}

/// <summary>
/// This method sets the configuration to enable transfer acceleration
/// for the bucket referred to in the bucketName parameter.
/// </summary>
/// <param name="client">An Amazon S3 client used to enable the
/// acceleration on an Amazon S3 bucket.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket for which
the
/// method will be enabling acceleration.</param>
private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
{
    try
    {
        var putRequest = new PutBucketAccelerateConfigurationRequest
        {
            BucketName = bucketName,
            AccelerateConfiguration = new AccelerateConfiguration
            {
                Status = BucketAccelerateStatus.Enabled,
            },
        };
        await client.PutBucketAccelerateConfigurationAsync(putRequest);

        var getRequest = new GetBucketAccelerateConfigurationRequest
        {
            BucketName = bucketName,
        };
        var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

        Console.WriteLine($"Acceleration state = '{response.Status}' ");
    }
}
```

```
    }  
    catch (AmazonS3Exception ex)  
    {  
        Console.WriteLine($"Error occurred. Message: '{ex.Message}' when  
setting transfer acceleration");  
    }  
}  
}
```

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

## CLI

### AWS CLI

バケットの加速設定を指定するには

次の `put-bucket-accelerate-configuration` の例では、指定したバケットの加速設定を有効にします。

```
aws s3api put-bucket-accelerate-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --accelerate-configuration Status=Enabled
```

このコマンドでは何も出力されません。

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定 S3 バケットの転送加速を有効にします。

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PutBucketAccelerateConfiguration](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、指定 S3 バケットの転送加速を有効にします。

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -
AccelerateConfiguration_Status $statusVal
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[PutBucketAccelerateConfiguration](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **PutBucketAc1** を使用する

次のサンプルコードは、PutBucketAc1 を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アクセスコントロールリスト \(ACL\) を管理する](#)

## .NET

### SDK for .NET

#### Note

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

```
///  
/// <summary>
```

```
/// Creates an Amazon S3 bucket with an ACL to control access to the
/// bucket and the objects stored in it.
/// </summary>
/// <param name="client">The initialized client object used to create
/// an Amazon S3 bucket, with an ACL applied to the bucket.
/// </param>
/// <param name="region">The AWS Region where the bucket will be
created.</param>
/// <param name="newBucketName">The name of the bucket to create.</param>
/// <returns>A boolean value indicating success or failure.</returns>
public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
client, S3Region region, string newBucketName)
{
    try
    {
        // Create a new Amazon S3 bucket with Canned ACL.
        var putBucketRequest = new PutBucketRequest()
        {
            BucketName = newBucketName,
            BucketRegion = region,
            CannedACL = S3CannedACL.LogDeliveryWrite,
        };

        PutBucketResponse putBucketResponse = await
client.PutBucketAsync(putBucketRequest);

        return putBucketResponse.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Amazon S3 error: {ex.Message}");
    }

    return false;
}
```

- APIの詳細については、AWS SDK for .NET API リファレンスの「[PutBucketAcl](#)」を参照してください。

## C++

## SDK for C++

 Note

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

```
bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
    &ownerID,
                                const Aws::String &granteePermission,
                                const Aws::String &granteeType, const Aws::String
    &granteeID,
                                const Aws::String &granteeEmailAddress,
                                const Aws::String &granteeURI, const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));
```

```

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
        s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
            << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
    enumeration.
/*!
    \param access: Human readable string.
    \return Permission: A Permission enum.
    */

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")

```

```

        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
 */

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}

```

- API の詳細については、AWS SDK for C++ API リファレンスの「[PutBucketAcl](#)」を参照してください。

## CLI

### AWS CLI

この例は、2 人の AWS ユーザー (user1@example.com と user2@example.com) に full control 権限を付与し、read 権限をすべてのユーザーに付与します。

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --grant-full-control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

カスタム ACL の詳細については、<http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> を参照してください (put-bucket-acl などの s3api ACL コマンドは、同じ略記法を使用します)。

- API の詳細については、AWS CLI コマンドリファレンスの「[PutBucketAcl](#)」を参照してください。



## Java

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code 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 SetAcl {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
        <bucketName> <id>\s

        Where:
        bucketName - The Amazon S3 bucket to grant permissions on.\s
    }
}
```

```

        id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
        """;

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

    String bucketName = args[0];
    String id = args[1];
    System.out.format("Setting access \n");
    System.out.println(" in bucket: " + bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setBucketAcl(s3, bucketName, id);
    System.out.println("Done!");
    s3.close();
}

/**
 * Sets the Access Control List (ACL) for an Amazon S3 bucket.
 *
 * @param s3 the S3Client instance to be used for the operation
 * @param bucketName the name of the S3 bucket to set the ACL for
 * @param id the ID of the AWS user or account that will be granted full
control of the bucket
 * @throws S3Exception if an error occurs while setting the bucket ACL
 */
public static void setBucketAcl(S3Client s3, String bucketName, String id) {
    try {
        Grant ownerGrant = Grant.builder()
            .grantee(builder -> builder.id(id)
                .type(Type.CANONICAL_USER))
            .permission(Permission.FULL_CONTROL)
            .build();

        List<Grant> grantList2 = new ArrayList<>();
        grantList2.add(ownerGrant);

        AccessControlPolicy acl = AccessControlPolicy.builder()

```

```
        .owner(builder -> builder.id(id))
        .grants(grantList2)
        .build();

    PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
        .bucket(bucketName)
        .accessControlPolicy(acl)
        .build();

    s3.putBucketAcl(putAclReq);

    } catch (S3Exception e) {
        e.printStackTrace();
        System.exit(1);
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[PutBucketAcl](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

バケット ACL をプットします。

```
import {
    PutBucketAclCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant read access to a user using their canonical AWS account ID.
```

```

*
* Most Amazon S3 use cases don't require the use of access control lists (ACLs).
* We recommend that you disable ACLs, except in unusual circumstances where
* you need to control access for each object individually. Consider a policy
  instead.
* For more information see https://docs.aws.amazon.com/AmazonS3/latest/
  userguide/bucket-policies.html.
* @param {{ bucketName: string, granteeCanonicalUserId: string,
  ownerCanonicalUserId }}
*/
export const main = async ({
  bucketName,
  granteeCanonicalUserId,
  ownerCanonicalUserId,
}) => {
  const client = new S3Client({});
  const command = new PutBucketAclCommand({
    Bucket: bucketName,
    AccessControlPolicy: {
      Grants: [
        {
          Grantee: {
            // The canonical ID of the user. This ID is an obfuscated form of
            your AWS account number.
            // It's unique to Amazon S3 and can't be found elsewhere.
            // For more information, see https://docs.aws.amazon.com/AmazonS3/
            latest/userguide/finding-canonical-user-id.html.
            ID: granteeCanonicalUserId,
            Type: "CanonicalUser",
          },
          // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
          // https://docs.aws.amazon.com/AmazonS3/latest/API/
          API\_Grant.html#AmazonS3-Type-Grant-Permission
          Permission: "READ",
        },
      ],
      Owner: {
        ID: ownerCanonicalUserId,
      },
    },
  });

  try {
    await client.send(command);
  }

```

```
console.log(`Granted READ access to ${bucketName}`);
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while setting ACL for bucket ${bucketName}. The bucket
doesn't exist.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while setting ACL for bucket ${bucketName}.
${caught.name}: ${caught.message}`
    );
  } else {
    throw caught;
  }
}
};
```

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

## Kotlin

### SDK for Kotlin

#### Note

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

```
suspend fun setBucketAcl(
    bucketName: String,
    idVal: String,
) {
    val myGrant =
```

```
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def grant_log_delivery_access(self):
        """
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only
        recommended
        use of an S3 bucket ACL.
        """
        try:
            acl = self.bucket.Acl()
            # Putting an ACL overwrites the existing ACL. If you want to preserve
            # existing grants, append new grants to the list of existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {
                        "Type": "Group",
                        "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
                    },
                    "Permission": "WRITE",
                }
            )
```

```

        }
    )
    acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
    logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
    except ClientError:
        logger.exception("Couldn't add ACL to bucket '%s'.",
self.bucket.name)
        raise

```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```

TRY.
  " Example: Grant log delivery access to a bucket
  " iv_grantwrite = 'uri=http://acs.amazonaws.com/groups/s3/LogDelivery'
  lo_s3->putbucketacl(
    iv_bucket = iv_bucket_name
    iv_grantwrite = iv_grantwrite ).
  MESSAGE 'Bucket ACL updated.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

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



AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketCors** を使用する

次のサンプルコードは、PutBucketCors を使用する方法を説明しています。

.NET

SDK for .NET

 Note

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

```
/// <summary>
/// Add CORS configuration to the Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used
/// to apply the CORS configuration to an Amazon S3 bucket.</param>
/// <param name="configuration">The CORS configuration to apply.</param>
private static async Task PutCORSConfigurationAsync(AmazonS3Client
client, CORSConfiguration configuration)
{
    PutCORSConfigurationRequest request = new
PutCORSConfigurationRequest()
    {
        BucketName = BucketName,
        Configuration = configuration,
    };

    _ = await client.PutCORSConfigurationAsync(request);
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[PutBucketCors](#)」を参照してください。

## CLI

## AWS CLI

次の例は、www.example.com からの PUT、POST、および DELETE の各リクエストを有効化し、任意のドメインからの GET リクエストを有効化します。

```
aws s3api put-bucket-cors --bucket amzn-s3-demo-bucket --cors-configuration file://cors.json
```

```
cors.json:
{
  "CORSRules": [
    {
      "AllowedOrigins": ["http://www.example.com"],
      "AllowedHeaders": ["*"],
      "AllowedMethods": ["PUT", "POST", "DELETE"],
      "MaxAgeSeconds": 3000,
      "ExposeHeaders": ["x-amz-server-side-encryption"]
    },
    {
      "AllowedOrigins": ["*"],
      "AllowedHeaders": ["Authorization"],
      "AllowedMethods": ["GET"],
      "MaxAgeSeconds": 3000
    }
  ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[PutBucketCors](#)」を参照してください。

## Java

## SDK for Java 2.x

 Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

import java.util.ArrayList;
import java.util.List;

import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code 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 S3Cors {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            "";

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

        String bucketName = args[0];
        String accountId = args[1];
```

```
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    /**
     * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an
     * Amazon S3 bucket.
     *
     * @param s3          the {@link S3Client} instance used to interact with
     *                    the Amazon S3 service
     * @param bucketName  the name of the Amazon S3 bucket for which the CORS
     *                    configuration should be deleted
     * @param accountId   the expected AWS account ID of the bucket owner
     *
     * @throws S3Exception if an error occurs while deleting the CORS
     *                    configuration for the bucket
     */
    public static void deleteBucketCorsInformation(S3Client s3, String
        bucketName, String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
                DeleteBucketCorsRequest.builder()
                    .bucket(bucketName)
                    .expectedBucketOwner(accountId)
                    .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
     * specified S3 bucket.
```

```

    *
    * @param s3 the S3Client instance to use for the operation
    * @param bucketName the name of the S3 bucket to retrieve the CORS
configuration for
    * @param accountId the expected bucket owner's account ID
    *
    * @throws S3Exception if there is an error retrieving the CORS configuration
    */
    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest =
GetBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.corsRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3
bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket to set the CORS rules for.
     * @param accountId The AWS account ID of the bucket owner.
     */
    public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
        List<String> allowMethods = new ArrayList<>();
        allowMethods.add("PUT");
    }

```

```
allowMethods.add("POST");
allowMethods.add("DELETE");

List<String> allowOrigins = new ArrayList<>();
allowOrigins.add("http://example.com");
try {
    // Define CORS rules.
    CORSRule corsRule = CORSRule.builder()
        .allowedMethods(allowMethods)
        .allowedOrigins(allowOrigins)
        .build();

    List<CORSRule> corsRules = new ArrayList<>();
    corsRules.add(corsRule);
    CORSConfiguration configuration = CORSConfiguration.builder()
        .corsRules(corsRules)
        .build();

    PutBucketCorsRequest putBucketCorsRequest =
    PutBucketCorsRequest.builder()
        .bucket(bucketName)
        .corsConfiguration(configuration)
        .expectedBucketOwner(accountId)
        .build();

    s3.putBucketCors(putBucketCorsRequest);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[PutBucketCors](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

CORS ルールを追加します。

```
import {
  PutBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Allows cross-origin requests to an S3 bucket by setting the CORS
 * configuration.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new PutBucketCorsCommand({
        Bucket: bucketName,
        CORSConfiguration: {
          CORSRules: [
            {
              // Allow all headers to be sent to this bucket.
              AllowedHeaders: ["*"],
              // Allow only GET and PUT methods to be sent to this bucket.
              AllowedMethods: ["GET", "PUT"],
              // Allow only requests from the specified origin.
              AllowedOrigins: ["https://www.example.com"],
              // Allow the entity tag (ETag) header to be returned in the
              // response. The ETag header
              // The entity tag represents a specific version of the object. The
              // ETag reflects
            }
          ]
        }
      })
    );
  } catch (err) {
    if (err instanceof S3ServiceException) {
      // Handle S3ServiceException
    }
  }
}
```

```
        // changes only to the contents of an object, not its metadata.
        ExposeHeaders: ["ETag"],
        // How long the requesting browser should cache the preflight
response. After
        // this time, the preflight request will have to be made again.
        MaxAgeSeconds: 3600,
    },
],
},
)),
);
console.log(`Successfully set CORS rules for bucket: ${bucketName}`);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while setting CORS rules for ${bucketName}. The bucket
doesn't exist.`
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while setting CORS rules for ${bucketName}.
${caught.name}: ${caught.message}`
        );
    } else {
        throw caught;
    }
}
};
```

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



## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_cors(self, cors_rules):
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that
        are
        allowed from other domains.

        :param cors_rules: The CORS rules to apply.
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules,
                self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
                self.bucket.name)
            raise
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
  end
end
```

```

    }
  ]
}
)
true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
#{e.message}"
  false
end
end
end

```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```

TRY.
  " Example: Allow PUT, POST, DELETE methods from http://www.example.com
  lo_s3->putbucketcors(
    iv_bucket = iv_bucket_name
    io_corsconfiguration = NEW /aws1/cl_s3_corsconfiguration(
      it_corsrules = it_cors_rules ) ).
  MESSAGE 'Bucket CORS configuration set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketEncryption** を使用する

次のサンプルコードは、PutBucketEncryption を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [オブジェクトストレージの使用開始](#)

.NET

SDK for .NET

 Note

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

```
/// <summary>
/// Set the bucket server side encryption to use AWSKMS with a customer-
managed key id.
/// </summary>
/// <param name="bucketName">Name of the bucket.</param>
/// <param name="kmsKeyId">The Id of the KMS Key.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> SetBucketServerSideEncryption(string
bucketName, string kmsKeyId)
{
    var serverSideEncryptionByDefault = new ServerSideEncryptionConfiguration
    {
        ServerSideEncryptionRules = new List<ServerSideEncryptionRule>
        {
            new ServerSideEncryptionRule
            {
                ServerSideEncryptionByDefault = new
ServerSideEncryptionByDefault
```

```
        {
            ServerSideEncryptionAlgorithm =
ServerSideEncryptionMethod.AWSKMS,
            ServerSideEncryptionKeyManagementServiceKeyId = kmsKeyId
        }
    }
};
try
{
    var encryptionResponse = await _s3Client.PutBucketEncryptionAsync(new
PutBucketEncryptionRequest
    {
        BucketName = bucketName,
        ServerSideEncryptionConfiguration =
serverSideEncryptionByDefault,
    });

    return encryptionResponse.HttpStatusCode == HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine(ex.ErrorCode == "AccessDenied"
        ? $"This account does not have permission to set encryption on
{bucketName}, please try again."
        : $"Unable to set bucket encryption for bucket {bucketName},
{ex.Message}");
}
return false;
}
```

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

## CLI

### AWS CLI

バケットのサーバー側の暗号化を設定するには

次の `put-bucket-encryption` の例では、指定したバケットのデフォルトとして AES256 暗号化を設定します。

```
aws s3api put-bucket-encryption \  
  --bucket amzn-s3-demo-bucket \  
  --server-side-encryption-configuration '{"Rules":  
  [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

このコマンドでは何も出力されません。

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定したバケットで Amazon S3 マネージドキー (SSE-S3) を使用したデフォルトの AES256 サーバー側暗号化を有効にします。

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
  @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PutBucketEncryption](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定したバケットで Amazon S3 マネージドキー (SSE-S3) を使用したデフォルトの AES256 サーバー側暗号化を有効にします。

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
  @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[PutBucketEncryption](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **PutBucketLifecycleConfiguration** を使用する

次のサンプルコードは、PutBucketLifecycleConfiguration を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [不完全なマルチパートアップロードを特定する](#)
- [S3 を使用して大量のメッセージを管理する](#)
- [バージョン管理されたオブジェクトを操作する](#)

### .NET

#### SDK for .NET

##### Note

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

```
/// <summary>
/// Adds lifecycle configuration information to the S3 bucket named in
/// the bucketName parameter.
/// </summary>
/// <param name="client">The S3 client used to call the
/// PutLifecycleConfigurationAsync method.</param>
/// <param name="bucketName">A string representing the S3 bucket to
/// which configuration information will be added.</param>
/// <param name="configuration">A LifecycleConfiguration object that
/// will be applied to the S3 bucket.</param>
public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
string bucketName, LifecycleConfiguration configuration)
{
    var request = new PutLifecycleConfigurationRequest()
    {
        BucketName = bucketName,
        Configuration = configuration,
    };
}
```

```
        var response = await client.PutLifecycleConfigurationAsync(request);  
    }
```

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

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットにライフサイクル設定を適用します。

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --  
lifecycle-configuration file:///lifecycle.json
```

lifecycle.json ファイルは、現在のフォルダ内の JSON ドキュメントで、次の 2 つのルールを指定します。

```
{  
  "Rules": [  
    {  
      "ID": "Move rotated logs to Glacier",  
      "Prefix": "rotated/",  
      "Status": "Enabled",  
      "Transitions": [  
        {  
          "Date": "2015-11-10T00:00:00.000Z",  
          "StorageClass": "GLACIER"  
        }  
      ]  
    },  
    {  
      "Status": "Enabled",  
      "Prefix": "",  
      "NoncurrentVersionTransitions": [  
        {  
          "NoncurrentDays": 2,  
          "StorageClass": "GLACIER"  
        }  
      ]  
    }  
  ]  
}
```



```
    ],  
    "ID": "Move old versions to Glacier"  
  }  
]  
}
```

1 つ目のルールは、指定した日付に、プレフィックス `rotated` の付いたファイルを Glacier に移動します。2 つ目のルールは、最新でない古いオブジェクトバージョンを Glacier に移動します。詳細については、「AWS CLI ユーザーガイド」の「Specifying Parameter Values」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[PutBucketLifecycleConfiguration](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;  
import software.amazon.awssdk.services.s3.model.Transition;  
import  
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;  
import  
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;  
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;  
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;  
import software.amazon.awssdk.services.s3.model.LifecycleRule;  
import software.amazon.awssdk.services.s3.model.ExpirationStatus;  
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;  
import  
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code 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 LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            """;

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

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and deleted a
Lifecycle configuration");
        s3.close();
    }
}
```

```
}

/**
 * Sets the lifecycle configuration for an Amazon S3 bucket.
 *
 * @param s3          The Amazon S3 client to use for the operation.
 * @param bucketName  The name of the Amazon S3 bucket.
 * @param accountId   The expected owner of the Amazon S3 bucket.
 *
 * @throws S3Exception if there is an error setting the lifecycle
configuration.
 */
public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
    try {
        // Create a rule to archive objects with the "glacierobjects/" prefix
to the
        // S3 Glacier Flexible Retrieval storage class immediately.
        LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
            .prefix("glacierobjects/")
            .build();

        Transition transition = Transition.builder()
            .storageClass(TransitionStorageClass.GLACIER)
            .days(0)
            .build();

        LifecycleRule rule1 = LifecycleRule.builder()
            .id("Archive immediately rule")
            .filter(ruleFilter)
            .transitions(transition)
            .status(ExpirationStatus.ENABLED)
            .build();

        // Create a second rule.
        Transition transition2 = Transition.builder()
            .storageClass(TransitionStorageClass.GLACIER)
            .days(0)
            .build();

        List<Transition> transitionList = new ArrayList<>();
        transitionList.add(transition2);

        LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
```

```
        .prefix("glacierobjects/")
        .build();

LifecycleRule rule2 = LifecycleRule.builder()
    .id("Archive and then delete rule")
    .filter(ruleFilter2)
    .transitions(transitionList)
    .status(ExpirationStatus.ENABLED)
    .build();

// Add the LifecycleRule objects to an ArrayList.
ArrayList<LifecycleRule> ruleList = new ArrayList<>();
ruleList.add(rule1);
ruleList.add(rule2);

BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
    .rules(ruleList)
    .build();

PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
    .builder()
    .bucket(bucketName)
    .lifecycleConfiguration(lifecycleConfiguration)
    .expectedBucketOwner(accountId)
    .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a
 * new lifecycle rule to it.
 *
 * @param s3 the S3Client instance used to interact with Amazon S3
 * @param bucketName the name of the Amazon S3 bucket
 * @param accountId the expected owner of the Amazon S3 bucket
```

```
    */
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
            List<LifecycleRule> newList = new ArrayList<>();
            List<LifecycleRule> rules = response.rules();
            for (LifecycleRule rule : rules) {
                newList.add(rule);
            }

            // Add a new rule with both a prefix predicate and a tag predicate.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("YearlyDocuments/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(3650)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("NewRule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the new rule to the list.
            newList.add(rule1);
            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(newList)
                .build();
```

```
        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
        .builder()
        .bucket(bucketName)
        .lifecycleConfiguration(lifecycleConfiguration)
        .expectedBucketOwner(accountId)
        .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Deletes the lifecycle configuration for an Amazon S3 bucket.
 *
 * @param s3 the {@link S3Client} to use for the operation
 * @param bucketName the name of the S3 bucket
 * @param accountId the expected account owner of the S3 bucket
 *
 * @throws S3Exception if an error occurs while deleting the lifecycle
 * configuration
 */
public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest
        .builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();

        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_lifecycle_configuration(self, lifecycle_rules):
        """
        Apply a lifecycle configuration to the bucket. The lifecycle
        configuration can
        be used to archive or delete the objects in the bucket according to
        specified
        parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
```

```

        LifecycleConfiguration={"Rules": lifecycle_rules}
    )
    logger.info(
        "Put lifecycle rules %s for bucket '%s'.",
        lifecycle_rules,
        self.bucket.name,
    )
except ClientError:
    logger.exception(
        "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
    )
    raise

```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```

TRY.
  " Example: Expire objects with prefix 'logs/' after 30 days
  lo_s3->putbucketlifecycleconf(
    iv_bucket = iv_bucket_name
    io_lifecycleconfiguration = NEW /aws1/cl_s3_bucketlcconf(
      it_rules = it_lifecycle_rule ) ).
  MESSAGE 'Bucket lifecycle configuration set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```



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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketLogging** を使用する

次のサンプルコードは、PutBucketLogging を使用する方法を説明しています。

.NET

SDK for .NET

 Note

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

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
/// </summary>
public class ServerAccessLogging
{
    private static IConfiguration _configuration = null!;

    public static async Task Main()
    {
```

```
LoadConfig();

string bucketName = _configuration["BucketName"];
string logBucketName = _configuration["LogBucketName"];
string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
string accountId = _configuration["AccountId"];

// If the AWS Region defined for your default user is different
// from the Region where your Amazon S3 bucket is located,
// pass the Region name to the Amazon S3 client object's constructor.
// For example: RegionEndpoint.USSouth1 or RegionEndpoint.USEast2.
IAmazonS3 client = new AmazonS3Client();

try
{
    // Update bucket policy for target bucket to allow delivery of
logs to it.
    await SetBucketPolicyToAllowLogDelivery(
        client,
        bucketName,
        logBucketName,
        logObjectKeyPrefix,
        accountId);

    // Enable logging on the source bucket.
    await EnableLoggingAsync(
        client,
        bucketName,
        logBucketName,
        logObjectKeyPrefix);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine($"Error: {e.Message}");
}

}

/// <summary>
/// This method grants appropriate permissions for logging to the
/// Amazon S3 bucket where the logs will be stored.
/// </summary>
/// <param name="client">The initialized Amazon S3 client which will be
used
/// to apply the bucket policy.</param>
```

```

    /// <param name="sourceBucketName">The name of the source bucket.</param>
    /// <param name="logBucketName">The name of the bucket where logging
    /// information will be stored.</param>
    /// <param name="logPrefix">The logging prefix where the logs should be
    delivered.</param>
    /// <param name="accountId">The account id of the account where the
    source bucket exists.</param>
    /// <returns>Async task.</returns>
    public static async Task SetBucketPolicyToAllowLogDelivery(
        IAmazonS3 client,
        string sourceBucketName,
        string logBucketName,
        string logPrefix,
        string accountId)
    {
        var resourceArn = @""arn:aws:s3::" + logBucketName + "/" +
logPrefix + @"""";

        var newPolicy = @"{
            ""Statement"": [{
                ""Sid"": ""S3ServerAccessLogsPolicy"",
                ""Effect"": ""Allow"",
                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                ""Action"": [""s3:PutObject""],
                ""Resource"": ["" + resourceArn + @""],
                ""Condition"": {
                    ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3::" + sourceBucketName + @"""" },
                    ""StringEquals"": { ""aws:SourceAccount"": """" +
accountId + @"""" }
                }
            }]
        }";

        Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
        Console.WriteLine(newPolicy);

        PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
        {
            BucketName = logBucketName,
            Policy = newPolicy,
        };
        await client.PutBucketPolicyAsync(putRequest);
    }

```

```
        Console.WriteLine("Policy applied.");
    }

    /// <summary>
    /// This method enables logging for an Amazon S3 bucket. Logs will be
stored
    /// in the bucket you selected for logging. Selected prefix
    /// will be prepended to each log object.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client which will be
used
    /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
you
    /// wish to enable logging.</param>
    /// <param name="logBucketName">The name of the Amazon S3 bucket where
logging
    /// information will be stored.</param>
    /// <param name="logObjectKeyPrefix">The prefix to prepend to each
    /// object key.</param>
    /// <returns>Async task.</returns>
    public static async Task EnableLoggingAsync(
        IAmazonS3 client,
        string bucketName,
        string logBucketName,
        string logObjectKeyPrefix)
    {
        Console.WriteLine($"Enabling logging for bucket {bucketName}.");
        var loggingConfig = new S3BucketLoggingConfig
        {
            TargetBucketName = logBucketName,
            TargetPrefix = logObjectKeyPrefix,
        };

        var putBucketLoggingRequest = new PutBucketLoggingRequest
        {
            BucketName = bucketName,
            LoggingConfig = loggingConfig,
        };
        await client.PutBucketLoggingAsync(putBucketLoggingRequest);
        Console.WriteLine($"Logging enabled.");
    }
}
```

```
/// <summary>
/// Loads configuration from settings files.
/// </summary>
public static void LoadConfig()
{
    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json", true) // Optionally, load
local settings.
        .Build();
}
}
```

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

## CLI

### AWS CLI

例 1: バケットポリシーのログ記録を設定するには

次の put-bucket-logging の例では、amzn-s3-demo-bucket のログ記録ポリシーを設定します。まず、put-bucket-policy コマンドを使用して、バケットポリシーでログ記録サービスプリンシパルに許可を付与します。

```
aws s3api put-bucket-policy \
  --bucket amzn-s3-demo-bucket \
  --policy file://policy.json
```

policy.json の内容:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3ServerAccessLogsPolicy",
      "Effect": "Allow",
```

```
    "Principal": {"Service": "logging.s3.amazonaws.com"},
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/Logs/*",
    "Condition": {
      "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},
      "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}
    }
  }
]
```

ログ記録ポリシーを適用するには、`put-bucket-logging` を使用します。

```
aws s3api put-bucket-logging \
  --bucket amzn-s3-demo-bucket \
  --bucket-logging-status file:///logging.json
```

`logging.json` の内容:

```
{
  "LoggingEnabled": {
    "TargetBucket": "amzn-s3-demo-bucket",
    "TargetPrefix": "Logs/"
  }
}
```

`put-bucket-policy` コマンドは、ログ記録サービスプリンシパルに `s3:PutObject` アクセス許可を付与するために必要です。

詳細については、「Amazon S3 ユーザーガイド」の「[Amazon S3 サーバーアクセスのログ記録](#)」を参照してください。

例 2: 1 人のユーザーのみにアクセスをログ記録するためのバケットポリシーを設定するには

次の `put-bucket-logging` の例では、`amzn-s3-demo-bucket` のログ記録ポリシーを設定します。AWS ユーザー `bob@example.com` はログファイルを完全に制御でき、他のユーザーはアクセスできません。まず、`put-bucket-acl` で S3 アクセス許可を付与します。

```
aws s3api put-bucket-acl \
  --bucket amzn-s3-demo-bucket \
  --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \
```

```
--grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

次に、`put-bucket-logging` を使用してログ記録ポリシーを適用します。

```
aws s3api put-bucket-logging \  
  --bucket amzn-s3-demo-bucket \  
  --bucket-logging-status file:///logging.json
```

`logging.json` の内容:

```
{  
  "LoggingEnabled": {  
    "TargetBucket": "amzn-s3-demo-bucket",  
    "TargetPrefix": "amzn-s3-demo-bucket-logs/",  
    "TargetGrants": [  
      {  
        "Grantee": {  
          "Type": "AmazonCustomerByEmail",  
          "EmailAddress": "bob@example.com"  
        },  
        "Permission": "FULL_CONTROL"  
      }  
    ]  
  }  
}
```

`put-bucket-acl` コマンドは、Amazon S3 のログ配信システムに必要なアクセス許可 (write および read-acp アクセス許可) を付与するために必要です。

詳細については、「Amazon S3 デベロッパーガイド」の「[Amazon S3 サーバーアクセスログ](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[PutBucketLogging](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

CLI で **PutBucketNotification** を使用する

次のサンプルコードは、`PutBucketNotification` を使用する方法を説明しています。

## CLI

## AWS CLI

amzn-s3-demo-bucket という名前のバケットに通知設定を適用します。

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-configuration file://notification.json
```

notification.json ファイルは、現在のフォルダにある JSON ファイルで、モニタリングする SNS トピックとイベントタイプを指定します。

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

SNS トピックには、Amazon S3 に公開を許可する IAM ポリシーがアタッチされている必要があります。

```
{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "example-statement-ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-west-2:123456789012:amzn-s3-demo-bucket",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
        }
      }
    }
  ]
}
```



```
]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[PutBucketNotification](#)」を参照してください。

## PowerShell

### Tools for PowerShell V4

例 1: この例では、S3 イベント ObjectRemovedDelete の SNS トピック設定を設定して、指定した S3 バケットの通知を有効にします

```
$topic = [Amazon.S3.Model.TopicConfiguration] @{
    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
    $topic
```

例 2: この例では、指定したバケットの ObjectCreatedAll の通知を有効にして、Lambda 関数に送信します。

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
    LambdaFunctionConfiguration $lambdaConfig
```

例 3: この例では、異なるキーサフィックスに基づいて 2 つの異なる Lambda 設定を作成し、両方を 1 つのコマンドで設定します。

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".ps1"}
            )
        }
    }
}

#Lambda Config 2

$secondLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = [Amazon.S3.EventType]::ObjectCreatedAll
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
    Id = "ObjectCreated-dada-json"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".json"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondLambdaConfig
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PutBucketNotification](#)」を参照してください。

## Tools for PowerShell V5

例 1: この例では、S3 イベント ObjectRemovedDelete の SNS トピック設定を設定して、指定した S3 バケットの通知を有効にします

```
$topic = [Amazon.S3.Model.TopicConfiguration] @{
    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
    $topic
```

例 2: この例では、指定したバケットの ObjectCreatedAll の通知を有効にして、Lambda 関数に送信します。

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
    LambdaFunctionConfiguration $lambdaConfig
```

例 3: この例では、異なるキーサフィックスに基づいて 2 つの異なる Lambda 設定を作成し、両方を 1 つのコマンドで設定します。

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*
```

```
FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
Id = "ObjectCreated-dada-ps1"
Filter = @{
    S3KeyFilter = @{
        FilterRules = @(
            @{Name="Prefix";Value="dada"}
            @{Name="Suffix";Value=".ps1"}
        )
    }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = [Amazon.S3.EventType]::ObjectCreatedAll
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
    Id = "ObjectCreated-dada-json"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".json"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[PutBucketNotification](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketNotificationConfiguration** を使用する

次のサンプルコードは、PutBucketNotificationConfiguration を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [S3 イベント通知の処理](#)
- [EventBridge へのイベント通知の送信](#)

.NET

SDK for .NET

 Note

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

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to enable notifications for an Amazon Simple
/// Storage Service (Amazon S3) bucket.
/// </summary>
public class EnableNotifications
{
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket1";
        const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
        const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

        IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
        await EnableNotificationAsync(client, bucketName, snsTopic,
sqsQueue);
    }
}
```

```
/// <summary>
/// This method makes the call to the PutBucketNotificationAsync method.
/// </summary>
/// <param name="client">An initialized Amazon S3 client used to call
/// the PutBucketNotificationAsync method.</param>
/// <param name="bucketName">The name of the bucket for which
/// notifications will be turned on.</param>
/// <param name="snsTopic">The ARN for the Amazon Simple Notification
/// Service (Amazon SNS) topic associated with the S3 bucket.</param>
/// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
/// (Amazon SQS) queue to which notifications will be pushed.</param>
public static async Task EnableNotificationAsync(
    IAmazonS3 client,
    string bucketName,
    string snsTopic,
    string sqsQueue)
{
    try
    {
        // The bucket for which we are setting up notifications.
        var request = new PutBucketNotificationRequest()
        {
            BucketName = bucketName,
        };

        // Defines the topic to use when sending a notification.
        var topicConfig = new TopicConfiguration()
        {
            Events = new List<EventType> { EventType.ObjectCreatedCopy },
            Topic = snsTopic,
        };
        request.TopicConfigurations = new List<TopicConfiguration>
        {
            topicConfig,
        };
        request.QueueConfigurations = new List<QueueConfiguration>
        {
            new QueueConfiguration()
            {
                Events = new List<EventType>
                { EventType.ObjectCreatedPut },
                Queue = sqsQueue,
            },
        };
    }
}
```

```
        // Now apply the notification settings to the bucket.
        PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}
}
```

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

## CLI

### AWS CLI

バケットへの指定された通知を有効にするには

次の `put-bucket-notification-configuration` の例では、`amzn-s3-demo-bucket` という名前のバケットに通知設定を適用します。`notification.json` ファイルは、現在のフォルダにある JSON ファイルで、モニタリングする SNS トピックとイベントタイプを指定します。

```
aws s3api put-bucket-notification-configuration \
  --bucket amzn-s3-demo-bucket \
  --notification-configuration file://notification.json
```

`notification.json` の内容:

```
{
  "TopicConfigurations": [
    {
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-
topic",
      "Events": [
        "s3:ObjectCreated:*"
```

```
    ]
  }
]
}
```

SNS トピックには、Amazon S3 に公開を許可する IAM ポリシーがアタッチされている必要があります。

```
{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "example-statement-ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-west-2:123456789012::s3-notification-
topic",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
        }
      }
    }
  ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[PutBucketNotificationConfiguration](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketPolicy** を使用する

次のサンプルコードは、PutBucketPolicy を使用する方法を説明しています。



## C++

## SDK for C++

 Note

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

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                  const Aws::String &policyBody,
                                  const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
        Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
        s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                   << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                   bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}

//! Build a policy JSON string.
/*!
```

```

\param userArn: Aws user Amazon Resource Name (ARN).
    For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
\param bucketName: Name of a bucket.
\return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                           const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\", \n"
        "  \"Statement\": [\n"
        "    {\n"
        "      \"Sid\": \"1\", \n"
        "      \"Effect\": \"Allow\", \n"
        "      \"Principal\": {\n"
        "        \"AWS\": \"\"
        + userArn +
        "\"\"\"      }, \n"
        "      \"Action\": [ \"s3:getObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3::\"
        + bucketName +
        \"/*\" ] \n"
        "    } \n"
        "  ] \n"
        "}";
}

```

- APIの詳細については、AWS SDK for C++ API リファレンスの「[PutBucketPolicy](#)」を参照してください。

## CLI

### AWS CLI

この例では、MySecretFolder 内のものを除いて、すべてのユーザーが amzn-s3-demo-bucket 内の任意のオブジェクトを取得できるようにしています。また、AWS アカウント 1234-5678-9012 のルートユーザーに、put および delete 権限を付与します。

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://policy.json
```

**policy.json:**

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/MySecretFolder/*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": [
        "s3:DeleteObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  ]
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[PutBucketPolicy](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code 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 SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <bucketName> <polFile>

        Where:
```

```

        bucketName - The Amazon S3 bucket to set the policy on.
        polFile - A JSON file containing the policy (see the Amazon S3
Readme for an example).\s
        """;

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

    String bucketName = args[0];
    String polFile = args[1];
    String policyText = getBucketPolicyFromFile(polFile);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setPolicy(s3, bucketName, policyText);
    s3.close();
}

/**
 * Sets the policy for an Amazon S3 bucket.
 *
 * @param s3          the {@link S3Client} object used to interact with the
Amazon S3 service
 * @param bucketName the name of the Amazon S3 bucket
 * @param policyText the text of the policy to be set on the bucket
 * @throws S3Exception if there is an error setting the bucket policy
 */
public static void setPolicy(S3Client s3, String bucketName, String
policyText) {
    System.out.println("Setting policy:");
    System.out.println("----");
    System.out.println(policyText);
    System.out.println("----");
    System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
            .bucket(bucketName)
            .policy(policyText)
            .build();
    }
}

```

```
s3.putBucketPolicy(policyReq);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Done!");
}

/**
 * Retrieves the bucket policy from a specified file.
 *
 * @param policyFile the path to the file containing the bucket policy
 * @return the content of the bucket policy file as a string
 */
public static String getBucketPolicyFromFile(String policyFile) {
    StringBuilder fileText = new StringBuilder();
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
        }

    } catch (IOException jpe) {
        jpe.printStackTrace();
    }
    return fileText.toString();
}
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[PutBucketPolicy](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

ポリシーを追加します。

```
import {
  PutBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant an IAM role GetObject access to all of the objects
 * in the provided bucket.
 * @param {{ bucketName: string, iamRoleArn: string }}
 */
export const main = async ({ bucketName, iamRoleArn }) => {
  const client = new S3Client({});
  const command = new PutBucketPolicyCommand({
    // This is a resource-based policy. For more information on resource-based
    // policies,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/
    // access_policies.html#policies_resource-based.
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
```

```
        AWS: iamRoleArn,
      },
      Action: "s3:GetObject",
      Resource: `arn:aws:s3:::${bucketName}/*`,
    },
  ],
}),
// Apply the preceding policy to this bucket.
Bucket: bucketName,
});

try {
  await client.send(command);
  console.log(
    `GetObject access to the bucket "${bucketName}" was granted to the provided IAM role.`
  );
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "MalformedPolicy"
  ) {
    console.error(
      `Error from S3 while setting the bucket policy for the bucket "${bucketName}". The policy was malformed.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while setting the bucket policy for the bucket "${bucketName}". ${caught.name}: ${caught.message}`
    );
  } else {
    throw caught;
  }
}
};
```

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



## Python

### SDK for Python (Boto3)

#### Note

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

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
            logger.info("Put policy %s for bucket '%s'.", policy,
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
self.bucket.name)
            raise
```

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

## Ruby

### SDK for Ruby

#### Note

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

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  # configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
#{e.message}"
    false
  end
end
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.  
    " Example policy JSON string  
    " iv_policy = '{"Version":"2012-10-17",          "Statement":  
[{"Effect":"Allow","Principal":{"AWS":{"arn:aws:iam::123456789012:user/  
user"}}, "Action":["s3:GetObject"], "Resource":["arn:aws:s3:::bucketname/*"]}]}'  
    lo_s3->putbucketpolicy(  
        iv_bucket = iv_bucket_name  
        iv_policy = iv_policy ).  
    MESSAGE 'Bucket policy set.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **PutBucketReplication** を使用する

次のサンプルコードは、PutBucketReplication を使用する方法を説明しています。

#### CLI

##### AWS CLI

S3 バケットのレプリケーションを設定するには

次の `put-bucket-replication` の例では、指定した S3 バケットにレプリケーション設定を適用します。

```
aws s3api put-bucket-replication \  
  --bucket amzn-s3-demo-bucket1 \  
  --replication-configuration file://replication.json
```

`replication.json` の内容:

```
{  
  "Role": "arn:aws:iam::123456789012:role/s3-replication-role",  
  "Rules": [  
    {  
      "Status": "Enabled",  
      "Priority": 1,  
      "DeleteMarkerReplication": { "Status": "Disabled" },  
      "Filter" : { "Prefix": ""},  
      "Destination": {  
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket2"  
      }  
    }  
  ]  
}
```

レプリケーション先のバケットではバージョニングが有効になっている必要があります。指定したロールは、レプリケーション先のバケットへの書き込みアクセス許可を持ち、Amazon S3 がそのロールを引き受けることを許可する信頼関係が必要です。

ロールのアクセス許可ポリシーの例:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetReplicationConfiguration",  
        "s3:ListBucket"  
      ],  
      "Resource": [  
        "arn:aws:s3:::amzn-s3-demo-bucket1"  
      ]  
    }  
  ]  
}
```

```

    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ReplicateTags"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket2/*"
    }
  ]
}

```

信頼関係ポリシーの例:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

このコマンドでは何も出力されません。

詳細については、「Amazon Simple Storage Service コンソールユーザーガイド」の「[チュートリアル: レプリケーションの設定例](#)」を参照してください。

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

## Java

### SDK for Java 2.x

#### Note

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

```
/**
 * Sets the replication configuration for an Amazon S3 bucket.
 *
 * @param s3Client          the S3Client instance to use for the operation
 * @param sourceBucketName  the name of the source bucket
 * @param destBucketName    the name of the destination bucket
 * @param destinationBucketARN the Amazon Resource Name (ARN) of the
destination bucket
 * @param roleARN           the ARN of the IAM role to use for the
replication configuration
 */
public static void setReplication(S3Client s3Client, String sourceBucketName,
String destBucketName, String destinationBucketARN, String roleARN) {
    try {
        Destination destination = Destination.builder()
            .bucket(destinationBucketARN)
            .storageClass(StorageClass.STANDARD)
            .build();

        // Define a prefix filter for replication.
        ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
            .prefix("documents/")
            .build();

        // Define delete marker replication setting.
        DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
            .status(DeleteMarkerReplicationStatus.DISABLED)
```

```
        .build();

// Create the replication rule.
ReplicationRule replicationRule = ReplicationRule.builder()
    .priority(1)
    .filter(ruleFilter)
    .status(ReplicationRuleStatus.ENABLED)
    .deleteMarkerReplication(deleteMarkerReplication)
    .destination(destination)
    .build();

List<ReplicationRule> replicationRuleList = new ArrayList<>();
replicationRuleList.add(replicationRule);

// Define the replication configuration with IAM role.
ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
    .role(roleARN)
    .rules(replicationRuleList)
    .build();

// Apply the replication configuration to the source bucket.
PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
    .bucket(sourceBucketName)
    .replicationConfiguration(configuration)
    .build();

s3Client.putBucketReplication(replicationRequest);
System.out.println("Replication configuration set successfully.");

} catch (IllegalArgumentException e) {
    System.err.println("Configuration error: " + e.getMessage());
} catch (S3Exception e) {
    System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
    System.err.println("Status Code: " + e.statusCode());
    System.err.println("Error Code: " + e.awsErrorDetails().errorCode());

} catch (SdkException e) {
    System.err.println("SDK Exception: " + e.getMessage());
}
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: この例では、バケット「amzn-s3-demo-bucket」内のキー名プレフィックス「TaxDocs」で作成された新しいオブジェクトの「amzn-s3-demo-bucket」バケットへのレプリケーションを有効にする単一のルールを使用してレプリケーション設定を指定します。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

例 2: この例では、キー名のプレフィックスが「TaxDocs」または「OtherDocs」の新しく作成されたオブジェクトを、バケット「amzn-s3-demo-bucket」にレプリケートするための複数のルールを使用して、レプリケーション設定を指定します。キーのプレフィックスの重複は許可されません。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }
```



```
$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

例 3: この例では、指定されたバケットのレプリケーション設定を更新して、「TaxDocs」というキー名のプレフィックスを持つオブジェクトのバケット「amzn-s3-demo-bucket」へのレプリケーションを制御するルールを無効にします。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PutBucketReplication](#)」を参照してください。

## Tools for PowerShell V5

例 1: この例では、バケット「amzn-s3-demo-bucket」内のキー名プレフィックス「TaxDocs」で作成された新しいオブジェクトの「amzn-s3-demo-bucket」バケットへのレプリケーションを有効にする単一のルールを使用してレプリケーション設定を指定します。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

例 2: この例では、キー名のプレフィックスが「TaxDocs」または「OtherDocs」の新しく作成されたオブジェクトを、バケット「amzn-s3-demo-bucket」にレプリケートするための複数のルールを使用して、レプリケーション設定を指定します。キーのプレフィックスの重複は許可されません。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
```

```
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

例 3: この例では、指定されたバケットのレプリケーション設定を更新して、「TaxDocs」というキー名のプレフィックスを持つオブジェクトのバケット「amzn-s3-demo-bucket」へのレプリケーションを制御するルールを無効にします。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[PutBucketReplication](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で **PutBucketRequestPayment** を使用する

次のサンプルコードは、PutBucketRequestPayment を使用する方法を説明しています。

## CLI

### AWS CLI

例 1: バケットの「リクエスト支払い」設定を有効にするには

次の `put-bucket-request-payment` の例では、指定したバケットの `requester pays` を有効にします。

```
aws s3api put-bucket-request-payment \  
  --bucket amzn-s3-demo-bucket \  
  --request-payment-configuration '{"Payer":"Requester"}'
```

このコマンドでは何も出力されません。

例 2: バケットの「リクエスト支払い」設定を無効にするには

次の `put-bucket-request-payment` の例では、指定したバケットの `requester pays` を無効にします。

```
aws s3api put-bucket-request-payment \  
  --bucket amzn-s3-demo-bucket \  
  --request-payment-configuration '{"Payer":"BucketOwner"}'
```

このコマンドでは何も出力されません。

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

## PowerShell

### Tools for PowerShell V4

例 1: 「amzn-s3-demo-bucket」という名前のバケットのリクエスト支払い設定を更新して、バケットからのダウンロードをリクエストしたユーザーにダウンロード料金が請求されるようにします。デフォルトでは、バケット所有者がダウンロード料金を支払います。リクエストの支払いをデフォルトに戻すには、`RequestPaymentConfiguration_Payer` パラメーターに「`BucketOwner`」を使用します。

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PutBucketRequestPayment](#)」を参照してください。

## Tools for PowerShell V5

例 1: 「amzn-s3-demo-bucket」という名前のバケットのリクエスト支払い設定を更新して、バケットからのダウンロードをリクエストしたユーザーにダウンロード料金が請求されるようにします。デフォルトでは、バケット所有者がダウンロード料金を支払います。リクエストの支払いをデフォルトに戻すには、RequestPaymentConfiguration\_Payer パラメーターに「BucketOwner」を使用します。

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[PutBucketRequestPayment](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## CLI で PutBucketTagging を使用する

次のサンプルコードは、PutBucketTagging を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [オブジェクトストレージの使用開始](#)

## CLI

### AWS CLI

次のコマンドは、タグ付けの設定を amzn-s3-demo-bucket という名前のバケットに適用します。

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://  
tagging.json
```

この `tagging.json` ファイルは、タグを指定する現在のフォルダ内の JSON ドキュメントです。

```
{
  "TagSet": [
    {
      "Key": "organization",
      "Value": "marketing"
    }
  ]
}
```

または、コマンドラインから直接タグ設定を `amzn-s3-demo-bucket` に適用します。

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging
'TagSet=[{Key=organization,Value=marketing}]'
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、**cloudtrail-test-2018** という名前のバケットに 2 つのタグを適用します。1 つは Stage のキーと値が Test のタグで、もう 1 つはキーが Environment で、値は Alpha のタグです。タグがバケットに追加されたことを確認するには、**Get-S3BucketTagging -BucketName bucket\_name** を実行します。結果では、最初のコマンドでバケットに適用したタグが表示されるはずですが、**Write-S3BucketTagging** は、バケットに設定されている既存のタグセット全体を上書きすることに注意が必要です。個別のタグを追加または削除するには、リソースグループとタグ付けの API コマンドレット、**Add-RGResourceTag**、**Remove-RGResourceTag** を実行します。または、AWS マネジメントコンソールのタグエディタを使用して S3 バケットのタグを管理します。

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
  Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

例 2: このコマンドは、バケット **cloudtrail-test-2018** を **Write-S3BucketTagging** コマンドレットにパイプします。これにより、Stage:Production と Department:Finance とい

うタグがバケットに適用されます。**Write-S3BucketTagging** は、バケットに設定されている既存のタグセット全体を上書きすることに注意が必要です。

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
Value="Finance" } )
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PutBucketTagging](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、**cloudtrail-test-2018** という名前のバケットに 2 つのタグを適用します。1 つは Stage のキーと値が Test のタグで、もう 1 つはキーが Environment で、値は Alpha のタグです。タグがバケットに追加されたことを確認するには、**Get-S3BucketTagging -BucketName bucket\_name** を実行します。結果では、最初のコマンドでバケットに適用したタグが表示されるはずですが、**Write-S3BucketTagging** は、バケットに設定されている既存のタグセット全体を上書きすることに注意が必要です。個別のタグを追加または削除するには、リソースグループとタグ付けの API コマンドレット、**Add-RGResourceTag**、**Remove-RGResourceTag** を実行します。または、AWS マネジメントコンソールのタグエディタを使用して S3 バケットのタグを管理します。

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

例 2: このコマンドは、バケット **cloudtrail-test-2018** を **Write-S3BucketTagging** コマンドレットにパイプします。これにより、Stage:Production と Department:Finance というタグがバケットに適用されます。**Write-S3BucketTagging** は、バケットに設定されている既存のタグセット全体を上書きすることに注意が必要です。

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
Value="Finance" } )
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[PutBucketTagging](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketVersioning** を使用する

次のサンプルコードは、PutBucketVersioning を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [オブジェクトストレージの使用開始](#)

## CLI

### AWS CLI

次のコマンドは、amzn-s3-demo-bucket という名前のバケットのバージョニングを有効にします。

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled
```

次のコマンドはバージョニングを有効にします。これは、MFA コードを使用します。

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled --mfa "SERIAL 123456"
```

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

## Java

### SDK for Java 2.x

#### Note

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



```
/**
 * Enables bucket versioning for the specified S3 bucket.
 *
 * @param s3Client the S3 client to use for the operation
 * @param bucketName the name of the S3 bucket to enable versioning for
 */
public static void enableBucketVersioning(S3Client s3Client, String
bucketName){
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    s3Client.putBucketVersioning(versioningRequest);
    System.out.println("Bucket versioning has been enabled for "+bucketName);
}
```

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

## PowerShell

### Tools for PowerShell V4

- 例 1: このコマンドは、指定した S3 バケットのバージョニングを有効にします。

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PutBucketVersioning](#)」を参照してください。

### Tools for PowerShell V5

- 例 1: このコマンドは、指定した S3 バケットのバージョニングを有効にします。

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- API の詳細については、AWS Tools for PowerShell コマンドレットリファレンス (V5) の「[PutBucketVersioning](#)」を参照してください。

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.  
    " Example: Enable versioning on a bucket  
    " iv_status = 'Enabled'  
    lo_s3->putbucketversioning(  
        iv_bucket = iv_bucket_name  
        io_versioningconfiguration = NEW /aws1/cl_s3_versioningconf(  
            iv_status = iv_status ) ).  
    MESSAGE 'Bucket versioning enabled.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutBucketWebsite** を使用する

次のサンプルコードは、PutBucketWebsite を使用する方法を説明しています。

## .NET

### SDK for .NET

#### Note

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

```
// Put the website configuration.
PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
{
    BucketName = bucketName,
    WebsiteConfiguration = new WebsiteConfiguration()
    {
        IndexDocumentSuffix = indexDocumentSuffix,
        ErrorDocument = errorDocument,
    },
};
PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- API の詳細については、AWS SDK for .NET API リファレンスの「PutBucketWebsite」を参照してください。 <https://docs.aws.amazon.com/goto/DotNetSDKV3/s3-2006-03-01/PutBucketWebsite>

## C++

### SDK for C++

#### Note

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

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::String &indexPath, const Aws::String
                                   &errorPage,
                                   const Aws::S3::S3ClientConfiguration
                                   &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPath);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
        client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                   << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「PutBucketWebsite」を参照してください。 <https://docs.aws.amazon.com/goto/SdkForCpp/s3-2006-03-01/PutBucketWebsite>

## CLI

### AWS CLI

amzn-s3-demo-bucket という名前のバケットに静的なウェブサイト設定を適用します。

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-configuration file://website.json
```

website.json ファイルは現在のフォルダ内の JSON ドキュメントで、ウェブサイトのインデックスページとエラーページを指定します。

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[PutBucketWebsite](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
```

```
/**
 * Before running this Java V2 code 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 SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucketName> [indexdoc]\s

            Where:
                bucketName    - The Amazon S3 bucket to set the website
configuration on.\s
                indexdoc - The index document, ex. 'index.html'
                        If not specified, 'index.html' will be set.

            """;

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

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    /**
     * Sets the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
     */
}
```

```
* @param bucketName The name of the S3 bucket to configure.
* @param indexDoc The name of the index document to use for the website
configuration.
*/
public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
    try {
        WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
            .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
            .build();

        PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .websiteConfiguration(websiteConfig)
            .build();

        s3.putBucketWebsite(pubWebsiteReq);
        System.out.println("The call was successful");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「PutBucketWebsite」を参照してください。 <https://docs.aws.amazon.com/goto/SdkForJavaV2/s3-2006-03-01/PutBucketWebsite>

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

ウェブサイト設定を設定します。

```
import {
  PutBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Configure an Amazon S3 bucket to serve a static website.
 * Website access must also be granted separately. For more information
 * on setting the permissions for website access, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/WebsiteAccessPermissionsReqd.html.
 *
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutBucketWebsiteCommand({
    Bucket: bucketName,
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
      IndexDocument: {
        // A suffix that is appended to a request when the request is
        // for a directory.
        Suffix: "index.html",
      },
    },
  },
  );
  try {
    await client.send(command);
    console.log(
      `The bucket "${bucketName}" has been configured as a static website.`
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
```



```
        console.error(
            `Error from S3 while configuring the bucket "${bucketName}" as a static
            website. The bucket doesn't exist.`
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while configuring the bucket "${bucketName}" as a static
            website. ${caught.name}: ${caught.message}`
        );
    } else {
        throw caught;
    }
}
};
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定したバケットのウェブサイトのホスティングを有効にして、インデックスドキュメントを「index.html」、エラードキュメントを「error.html」と指定します。

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PutBucketWebsite](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定したバケットのウェブサイトのホスティングを有効にして、インデックスドキュメントを「index.html」、エラードキュメントを「error.html」と指定します。

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
  -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[PutBucketWebsite](#)」を参照してください。

## Ruby

### SDK for Ruby

#### Note

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

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
  # configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the
  # website.
  # @param error_document [String] The name of the error document to show for 4XX
  # errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
  # false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
```

```
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
why: #{e.message}"
    false
  end
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API の詳細については、AWS SDK for Ruby API リファレンスの「PutBucketWebsite」を参照してください。 <https://docs.aws.amazon.com/goto/SdkForRubyV3/s3-2006-03-01/PutBucketWebsite>

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutObject** を使用する

次のサンプルコードは、PutObject を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

- [オブジェクトストレージの使用開始](#)
- [条件付きリクエストの実行](#)
- [アップロードとダウンロードを追跡する](#)
- [Amazon S3 オブジェクトの整合性の操作](#)

## .NET

### SDK for .NET (v4)

#### Note

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

```
/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
/// upload procedure.</returns>
public async Task<bool> UploadFileAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    try
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
        };
    }
```

```
        var response = await _amazonS3.PutObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
        return false;
    }
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[PutObject](#)」を参照してください。

SDK for .NET

 Note

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

サーバー側の暗号化を使用してオブジェクトをアップロードします。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket with server-side encryption enabled.
/// </summary>
public class ServerSideEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "samplefile.txt";

        // If the AWS Region defined for your default user is different
```

```
// from the Region where your Amazon S3 bucket is located,
// pass the Region name to the Amazon S3 client object's constructor.
// For example: RegionEndpoint.USWest2.
IAmazonS3 client = new AmazonS3Client();

await WritingAnObjectAsync(client, bucketName, keyName);
}

/// <summary>
/// Upload a sample object include a setting for encryption.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
/// to upload a file and apply server-side encryption.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// encrypted object will reside.</param>
/// <param name="keyName">The name for the object that you want to
/// create in the supplied bucket.</param>
public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
{
    try
    {
        var putRequest = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
            ContentBody = "sample text",
            ServerSideEncryptionMethod =
ServerSideEncryptionMethod.AES256,
        };

        var putResponse = await client.PutObjectAsync(putRequest);

        // Determine the encryption state of an object.
        GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
        {
            BucketName = bucketName,
            Key = keyName,
        };
        GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
        ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;
```

```
        Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: '{ex.Message}' when writing an
object");
    }
}
}
```

条件付きリクエストを使用してオブジェクトを配置します。

```
/// <summary>
/// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
/// </summary>
/// <param name="objectKey">The key of the object to upload.</param>
/// <param name="bucket">The source bucket of the object.</param>
/// <param name="content">The content to upload as a string.</param>
/// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
{
    try
    {
        var putObjectRequest = new PutObjectRequest
        {
            BucketName = bucket,
            Key = objectKey,
            ContentBody = content,
            IfNoneMatch = "*"
        };

        var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
        _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
        return putResult.ETag;
    }
}
```

```

        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "PreconditionFailed")
            {
                _logger.LogError("Conditional write failed: Precondition
failed");
            }
            else
            {
                _logger.LogError($"Unexpected error: {e.ErrorCode}");
                throw;
            }
            return string.Empty;
        }
    }
}

```

- API の詳細については、AWS SDK for .NET API リファレンスの「[PutObject](#)」を参照してください。

## Bash

### AWS CLIBash スクリプトを使用した

#### Note

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

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function copy_file_to_bucket

```



```
#
# This function creates a file in the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.
#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

    response=$(aws s3api put-object \
        --bucket "$bucket_name" \
        --body "$source_file" \
        --key "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}
```

- APIの詳細については、AWS CLI コマンドリファレンスの「[PutObject](#)」を参照してください。

## C++

## SDK for C++

 Note

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

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
    needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                       fileName.c_str(),
                                       std::ios_base::in |
std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
        s3Client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
```

```
        std::cout << "Added object '" << fileName << "' to bucket '"  
            << bucketName << "'.";  
    }  
  
    return outcome.IsSuccess();  
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「[PutObject](#)」を参照してください。

## CLI

### AWS CLI

#### 例 1: オブジェクトを Amazon S3 にアップロードする

次の put-object コマンドの例では、オブジェクトを Amazon S3 にアップロードします。

```
aws s3api put-object \  
  --bucket amzn-s3-demo-bucket \  
  --key my-dir/MySampleImage.png \  
  --body MySampleImage.png
```

オブジェクトのアップロードの詳細については、Amazon S3 デベロッパーガイドの「オブジェクトのアップロード < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html>>」を参照してください。

#### 例 2: ビデオファイルを Amazon S3 にアップロードする

次の put-object コマンドの例では、ビデオファイルをアップロードします。

```
aws s3api put-object \  
  --bucket amzn-s3-demo-bucket \  
  --key my-dir/big-video-file.mp4 \  
  --body /media/videos/f-sharp-3-data-services.mp4
```

オブジェクトのアップロードの詳細については、Amazon S3 デベロッパーガイドの「オブジェクトのアップロード < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html>>」を参照してください。

- API の詳細については、「AWS CLI Command Reference」の「[PutObject](#)」を参照してください。

Go

SDK for Go V2

 Note

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

低レベル API を使用してオブジェクトをバケットに配置します。

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}
```

```
// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:     aws.String(objectKey),
            Body:    file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
                log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
                    fileName, bucketName, objectKey, err)
            }
        } else {
            err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
            }
        }
    }
    return err
}
```

Transfer Manager を使用してオブジェクトをバケットにアップロードします。

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
    string, contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:      aws.String(bucket),
        Key:         aws.String(key),
        Body:        bytes.NewReader([]byte(contents)),
        ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
    }
    output, err := actor.S3Manager.Upload(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    } else {
        err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
            Bucket: aws.String(bucket),
            Key:   aws.String(key),
        })
    }
}
```

```
    }, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
            bucket)
    } else {
        outKey = *output.Key
    }
}
return outKey, err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

[S3Client](#) を使用してバケットにファイルをアップロードします。

```
/**
 * Uploads a local file to an AWS S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (object name) to use for the uploaded file
 * @param objectPath  the local file path of the file to be uploaded
 * @return a {@link CompletableFuture} that completes with the {@link
 * PutObjectResponse} when the upload is successful, or throws a {@link
 * RuntimeException} if the upload fails
 */
public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
    bucketName, String key, String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
```

```
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
    return response.whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to upload file", ex);
        }
    });
}
```

[S3TransferManager](#) を使用してバケットに[ファイルをアップロード](#)します。[完全なファイルとテスト](#)を表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String
bucketName,

                                String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```



```
}
```

[S3Client](#) を使用してオブジェクトをバケットにアップロードし、タグを設定します。

```
/**
 * Puts tags on an Amazon S3 object.
 *
 * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
 * @param bucketName The name of the Amazon S3 bucket.
 * @param objectKey The key of the Amazon S3 object.
 * @param objectPath The file path of the object to be uploaded.
 */
public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
    try {
        Tag tag1 = Tag.builder()
            .key("Tag 1")
            .value("This is tag 1")
            .build();

        Tag tag2 = Tag.builder()
            .key("Tag 2")
            .value("This is tag 2")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag1);
        tags.add(tag2);

        Tagging allTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .tagging(allTags)
            .build();

        s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));
    }
}
```

```
        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the tags associated with an object in an Amazon S3 bucket.
     *
     * @param s3 an instance of the S3Client class, which is used to interact
     with the Amazon S3 service
     * @param bucketName the name of the S3 bucket containing the object
     * @param objectKey the key (or name) of the object in the S3 bucket
     * @throws S3Exception if there is an error updating the object's tags
     */
    public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
        try {
            GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
            List<Tag> obTags = getTaggingRes.getTagSet();
            for (Tag sinTag : obTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

            // Replace the object's tags with two new tags.
            Tag tag3 = Tag.builder()
                .key("Tag 3")
                .value("This is tag 3")
                .build();

            Tag tag4 = Tag.builder()
                .key("Tag 4")
                .value("This is tag 4")
                .build();

            List<Tag> tags = new ArrayList<>();
```

```
        tags.add(tag3);
        tags.add(tag4);

        Tagging updatedTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .tagging(updatedTags)
            .build();

        s3.putObjectTagging(taggingRequest1);
        GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
        List<Tag> modTags = getTaggingRes2.tagSet();
        for (Tag sinTag : modTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

/**
 * Retrieves the contents of a file as a byte array.
 *
 * @param filePath the path of the file to be read
 * @return a byte array containing the contents of the file, or null if an
error occurs
 */
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
```

```
        fileInputStream.read(bytesArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    return bytesArray;
}
}
```

[S3Client](#) を使用してオブジェクトをバケットにアップロードし、メタデータを設定します。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code 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 PutObjectMetadata {
    public static void main(String[] args) {
```

```

    final String USAGE = ""

    Usage:
        <bucketName> <objectKey> <objectPath>\s

    Where:
        bucketName - The Amazon S3 bucket to upload an object into.
        objectKey - The object to upload (for example, book.pdf).
        objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
        """;

    if (args.length != 3) {
        System.out.println(USAGE);
        System.exit(1);
    }

    String bucketName = args[0];
    String objectKey = args[1];
    String objectPath = args[2];
    System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
    System.out.println("  in bucket: " + bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    putS3Object(s3, bucketName, objectKey, objectPath);
    s3.close();
}

/**
 * Uploads an object to an Amazon S3 bucket with metadata.
 *
 * @param s3 the S3Client object used to interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket to upload the object to
 * @param objectKey the name of the object to be uploaded
 * @param objectPath the local file path of the object to be uploaded
 */
public static void putS3Object(S3Client s3, String bucketName, String
objectKey, String objectPath) {
    try {
        Map<String, String> metadata = new HashMap<>();

```

```
        metadata.put("author", "Mary Doe");
        metadata.put("version", "1.0.0.0");

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket
" + bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

[S3Client](#) を使用してオブジェクトをバケットにアップロードし、オブジェクトの保持値を設定します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code 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 PutObjectRetention {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <key> <bucketName>\s

            Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object
(for example, bucket1).\s
            "";

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

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    /**
     * Sets the retention period for an object in an Amazon S3 bucket.
     *
     * @param s3      the S3Client object used to interact with the Amazon S3
service
     * @param key     the key (name) of the object in the S3 bucket
     * @param bucket the name of the S3 bucket where the object is stored
     *
     * @throws S3Exception if an error occurs while setting the object retention
period
     */
}

```

```
public static void setRetentionPeriod(S3Client s3, String key, String bucket) {
    try {
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()
            .mode("COMPLIANCE")
            .retainUntilDate(instant)
            .build();

        PutObjectRetentionRequest retentionRequest =
        PutObjectRetentionRequest.builder()
            .bucket(bucket)
            .key(key)
            .bypassGovernanceRetention(true)
            .retention(lockRetention)
            .build();

        // To set Retention on an object, the Amazon S3 bucket must support
object
        // locking, otherwise an exception is thrown.
        s3.putObjectRetention(retentionRequest);
        System.out.print("An object retention configuration was successfully
placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[PutObject](#)」を参照してください。



## JavaScript

### SDK for JavaScript (v3)

#### Note

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

オブジェクトをアップロードします。

```
import { readFile } from "node:fs/promises";

import {
  PutObjectCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Upload a file to an S3 bucket.
 * @param {{ bucketName: string, key: string, filePath: string }}
 */
export const main = async ({ bucketName, key, filePath }) => {
  const client = new S3Client({});
  const command = new PutObjectCommand({
    Bucket: bucketName,
    Key: key,
    Body: await readFile(filePath),
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "EntityTooLarge"
    ) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}. \`

```

```

The object was too large. To upload objects larger than 5GB, use the S3 console
(160GB max) \
or the multipart upload API (5TB max).`,
    );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while uploading object to ${bucketName}. ${caught.name}:
${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

ETag が指定されたものと一致することを条件にオブジェクトをアップロードします。

```

import {
    GetObjectCommand,
    NoSuchKey,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
    const client = new S3Client({});

    try {
        const response = await client.send(
            new GetObjectCommand({
                Bucket: bucketName,
                Key: key,
                IfMatch: eTag,
            }),
        );
        // The Body object also has 'transformToByteArray' and 'transformToWebStream'
        methods.
    }
};

```

```
const str = await response.Body.transformToString();
console.log("Success. Here is text of the file:", str);
} catch (caught) {
  if (caught instanceof NoSuchKey) {
    console.error(
      `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utills/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
```

```
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

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

## Kotlin

### SDK for Kotlin

#### Note

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

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteArray()
        }
}
```

```
S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
    val response = s3.putObject(request)
    println("Tag information is ${response.eTag}")
}
}
```

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

## PHP

### SDK for PHP

#### Note

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

オブジェクトをバケットにアップロードします。

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

$fileName = __DIR__ . "/local-file-" . uniqid();
try {
    $this->s3client->putObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
        'SourceFile' => __DIR__ . '/testfile.txt'
    ]);
    echo "Uploaded $fileName to $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to upload $fileName with error: " . $exception-
    >getMessage();
    exit("Please fix error with file upload before continuing.");
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、単一のファイル「local-sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「sample.txt」というキーを持つオブジェクトを作成します。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-sample.txt
```

例 2: このコマンドは、単一のファイル「sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「sample.txt」というキーを持つオブジェクトを作成します。-Key パラメータを指定しない場合、ファイル名が S3 オブジェクトキーとして使用されます。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

例 3: このコマンドは、単一のファイル「local-sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「prefix/to/sample.txt」というキーを持つオブジェクトを作成します。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -File .\local-sample.txt
```

例 4: このコマンドは、サブディレクトリ「Scripts」内のすべてのファイルをバケット「test-files」にアップロードして、共通のキープレフィックス「SampleScripts」を各オブジェクトに適用します。アップロードされた各ファイルは「SampleScripts/filename」というキーを持ちます。ただし、「filename」の部分はそれぞれ異なります。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix SampleScripts\
```

例 5: このコマンドは、ローカルディレクトリ「Scripts」内のすべての \*.ps1 ファイルをバケット「test-files」にアップロードして、共通のキープレフィックス「SampleScripts」を各オブジェクトに適用します。アップロードされた各ファイルは「SampleScripts/filename.ps1」というキーを持ちます。ただし、「filename」の部分はそれぞれ異なります。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

例 6: このコマンドは、「sample.txt」というキーを持つ、指定されたコンテンツ文字列を含む新しい S3 オブジェクトを作成します。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

例 7: このコマンドは、指定したファイル (ファイル名をキーとして使用) をアップロードして、指定したタグを新しいオブジェクトに適用します。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

例 8: このコマンドは、指定したフォルダを再帰的にアップロードして、指定したタグをすべての新しいオブジェクトに適用します。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"  
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PutObject](#)」を参照してください。

## Tools for PowerShell V5

例 1: このコマンドは、単一のファイル「local-sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「sample.txt」というキーを持つオブジェクトを作成します。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-  
sample.txt
```

例 2: このコマンドは、単一のファイル「sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「sample.txt」というキーを持つオブジェクトを作成します。-Key パラメータを指定しない場合、ファイル名が S3 オブジェクトキーとして使用されます。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

例 3: このコマンドは、単一のファイル「local-sample.txt」を Amazon S3 にアップロードして、バケット「test-files」に「prefix/to/sample.txt」というキーを持つオブジェクトを作成します。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -
File .\local-sample.txt
```

例 4: このコマンドは、サブディレクトリ「Scripts」内のすべてのファイルをバケット「test-files」にアップロードして、共通のキープレフィックス「SampleScripts」を各オブジェクトに適用します。アップロードされた各ファイルは「SampleScripts/filename」というキーを持ちます。ただし、「filename」の部分はそれぞれ異なります。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
SampleScripts\
```

例 5: このコマンドは、ローカルディレクトリ「Scripts」内のすべての \*.ps1 ファイルをバケット「test-files」にアップロードして、共通のキープレフィックス「SampleScripts」を各オブジェクトに適用します。アップロードされた各ファイルは「SampleScripts/filename.ps1」というキーを持ちます。ただし、「filename」の部分はそれぞれ異なります。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
SampleScripts\ -SearchPattern *.ps1
```

例 6: このコマンドは、「sample.txt」というキーを持つ、指定されたコンテンツ文字列を含む新しい S3 オブジェクトを作成します。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
contents"
```

例 7: このコマンドは、指定したファイル (ファイル名をキーとして使用) をアップロードして、指定したタグを新しいオブジェクトに適用します。

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

例 8: このコマンドは、指定したフォルダを再帰的にアップロードして、指定したタグをすべての新しいオブジェクトに適用します。



```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
  -Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[PutObject](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                               that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string.
        When this
                               argument is a string, it is interpreted as a file name,
        which is
                               opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
```

```

        try:
            put_data = open(data, "rb")
        except IOError:
            logger.exception("Expected file name or binary data, got '%s'."
                             data)
            raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'."
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'."
                self.object.key,
                self.object.bucket_name,
            )
            raise
        finally:
            if getattr(put_data, "close", None):
                put_data.close()

```

条件付きリクエストを使用してオブジェクトをアップロードします。

```

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

```

```
def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
    """
    Uploads an object to Amazon S3 with a conditional request. Prevents
    overwrite
    using an IfNoneMatch condition for the object key.

    :param object_key: The key of the object to upload.
    :param source_bucket: The source bucket of the object.
    :param data: The data to upload.
    """
    try:
        self.s3.put_object(
            Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
        )
        print(
            f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
        )
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional write failed: Precondition failed")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise
```

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

## Ruby

### SDK for Ruby

#### Note

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

マネージドアップローダー (Object.upload\_file) を使用してファイルをアップロードします。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
#{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"
```

```
wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
object_key))
return unless wrapper.upload_file(file_path)

puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Object.put を使用してファイルをアップロードします。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
#{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
```

```
return unless success

puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Object.put を使用してファイルをアップロードし、サーバー側の暗号化を追加します。

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
    object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
    #{encryption}."
```

```
end

run_demo if $PROGRAM_NAME == __FILE__
```

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

## Rust

### SDK for Rust

#### Note

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

```
pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
"Get contents of file from application server."
DATA lv_body TYPE xstring.
OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
READ DATASET iv_file_name INTO lv_body.
CLOSE DATASET iv_file_name.

"Upload/put an object to an S3 bucket."
TRY.
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_file_name
        iv_body = lv_body ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

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



## Swift

### SDK for Swift

#### Note

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

```
import AWSS3
import Smithy

public func uploadFile(bucket: String, key: String, file: String) async
throws {
    let fileUrl = URL(fileURLWithPath: file)
    do {
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object. "))
        throw error
    }
}
```

```
import AWSS3
import Smithy

public func createFile(bucket: String, key: String, withData data: Data)
async throws {
    let dataStream = ByteStream.data(data)
```

```
let input = PutObjectInput(  
    body: dataStream,  
    bucket: bucket,  
    key: key  
)  
  
do {  
    _ = try await client.putObject(input: input)  
}  
catch {  
    print("ERROR: ", dump(error, name: "Putting an object."))  
    throw error  
}  
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **PutObjectAcl** を使用する

次のサンプルコードは、PutObjectAcl を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [アクセスコントロールリスト \(ACL\) を管理する](#)

C++

SDK for C++

### Note

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

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
&objectKey, const Aws::String &ownerID,
                                const Aws::String &granteePermission, const
    Aws::String &granteeType,
                                const Aws::String &granteeID, const Aws::String
&granteeEmailAddress,
                                const Aws::String &granteeURI, const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutObjectAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
    request.SetKey(objectKey);
```

```

    Aws::S3::Model::PutObjectAclOutcome outcome =
        s3Client.PutObjectAcl(request);

    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
            << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
    enumeration.
/*!
    \param access: Human readable string.
    \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
    enumeration.
/*!
    \param type: Human readable string.
    \return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")

```

```
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

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

## CLI

### AWS CLI

次のコマンドは、2 人の AWS ユーザー (user1@example.com と user2@example.com) に full control 権限を付与し、read 権限をすべてのユーザーに付与します。

```
aws s3api put-object-acl --bucket amzn-s3-demo-bucket --key file.txt --grant-  
full-control emailaddress=user1@example.com,emailaddress=user2@example.com --  
grant-read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

カスタム ACL の詳細については、<http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> を参照してください (put-object-acl などの s3api ACL コマンドは、同じ略記法を使用します)。

- API の詳細については、AWS CLI コマンドリファレンスの「[PutObjectAcl](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                               that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def put_acl(self, email):
        """
        Applies an ACL to the object that grants read access to an AWS user
        identified
        by email address.

        :param email: The email address of the user to grant access.
        """
        try:
            acl = self.object.Acl()
            # Putting an ACL overwrites the existing ACL, so append new grants
            # if you want to preserve existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
                    "Permission": "READ",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted read access to %s.", email)
        except ClientError:
            logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
            raise
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.  
  " Example: Grant read access to an AWS user  
  " iv_grantread = 'emailAddress=user@example.com'  
  lo_s3->putobjectacl(  
    iv_bucket = iv_bucket_name  
    iv_key = iv_object_key  
    iv_grantread = iv_grantread ).  
  MESSAGE 'Object ACL updated.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
  MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK または CLI で **PutObjectLegalHold** を使用する

次のサンプルコードは、PutObjectLegalHold を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon S3 オブジェクトをロックする](#)

## .NET

### SDK for .NET

#### Note

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

```
/// <summary>
/// Set or modify a legal hold on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="holdStatus">The On or Off status for the legal hold.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectLegalHold(string bucketName,
    string objectKey, ObjectLockLegalHoldStatus holdStatus)
{
    try
    {
        var request = new PutObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            LegalHold = new ObjectLockLegalHold()
            {
                Status = holdStatus
            }
        };

        var response = await _amazonS3.PutObjectLegalHoldAsync(request);
        Console.WriteLine($"\\tModified legal hold for {objectKey} in
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tError modifying legal hold: '{ex.Message}'");
        return false;
    }
}
```



```
}
```

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

## CLI

### AWS CLI

リーガルホールドをオブジェクトに適用するには

次の `put-object-legal-hold` 例では、`doc1.rtf` という名前のバケットのオブジェクトにリーガルホールドを設定します。

```
aws s3api put-object-legal-hold \  
  --bucket amzn-s3-demo-bucket-with-object-lock \  
  --key doc1.rtf \  
  --legal-hold Status=ON
```

このコマンドでは何も出力されません。

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

## Go

### SDK for Go V2

#### Note

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

```
import (  
  "bytes"  
  "context"  
  "errors"  
  "fmt"
```

```
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }

    _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for " + objectKey + " in
    "+bucketName +".");
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[PutObjectLegalHold](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

```
import {
  PutObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Apply a legal hold configuration to the specified object.
 * @param {{ bucketName: string, objectKey: string, legalHoldStatus: "ON" | "OFF" }}
 */
export const main = async ({ bucketName, objectKey, legalHoldStatus }) => {
  if (!["OFF", "ON"].includes(legalHoldStatus.toUpperCase())) {
    throw new Error(
      "Invalid parameter. legalHoldStatus must be 'ON' or 'OFF'."
    );
  }

  const client = new S3Client({});
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: legalHoldStatus,
    },
  });
```

```
    },
  });

  try {
    await client.send(command);
    console.log(
      `Legal hold status set to "${legalHoldStatus}" for "${objectKey}" in
"${bucketName}"`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
"${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
"${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    objectKey: {
      type: "string",
```

```

        required: true,
    },
    legalHoldStatus: {
        type: "string",
        default: "ON",
    },
  },
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
}

```

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

## Python

### SDK for Python (Boto3)

#### Note

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

オブジェクトにリーガルホールドを適用します。

```

def set_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Set a legal hold on a specific file in a bucket.

    Args:

```

```
s3_client: Boto3 S3 client.
bucket: The name of the bucket containing the file.
key: The key of the file to set the legal hold on.
"""
print()
logger.info("Setting legal hold on file [%s] in bucket [%s]", key, bucket)
try:
    before_status = "OFF"
    after_status = "ON"
    s3_client.put_object_legal_hold(
        Bucket=bucket, Key=key, LegalHold={"Status": after_status}
    )
    logger.debug(
        "Legal hold set successfully on file [%s] in bucket [%s]", key,
bucket
    )
    _print_legal_hold_update(bucket, key, before_status, after_status)
except Exception as e:
    logger.error(
        "Failed to set legal hold on file [%s] in bucket [%s]: %s", key,
bucket, e
    )
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

TRY.

" Example: Set legal hold status to ON

```

    " iv_status = 'ON'
    lo_s3->putobjectlegalhold(
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key
        io_legalhold = NEW /aws1/cl_s3_objlocklegalhold(
            iv_status = iv_status ) ).
    MESSAGE 'Object legal hold status set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutObjectLockConfiguration** を使用する

次のサンプルコードは、PutObjectLockConfiguration を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon S3 オブジェクトをロックする](#)

.NET

SDK for .NET

#### Note

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

バケットのオブジェクトロック設定を指定します。



```
/// <summary>
/// Enable object lock on an existing bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to modify.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableObjectLockOnBucket(string bucketName)
{
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
            },
        };

        var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"{bucketName}\tAdded an object lock policy to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
        return false;
    }
}
```

バケットのデフォルトの保存期間を設定します。

```
/// <summary>
/// Set or modify a retention period on an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to modify.</param>
/// <param name="retention">The retention mode.</param>
/// <param name="retainUntilDate">The date for retention until.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
{
    var enabledString = enableObjectLock ? "Enabled" : "Disabled";
    var timeDifference = retainUntilDate.Subtract(DateTime.Now);
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                Rule = new ObjectLockRule()
                {
                    DefaultRetention = new DefaultRetention()
                    {
                        Mode = retention,
                        Days = timeDifference.Days // Can be specified in
days or years but not both.
                    }
                }
            }
        }
    }
}
```

```
};

var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
Console.WriteLine($"\\tAdded a default retention to bucket
{bucketName}.");
return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tError modifying object lock: '{ex.Message}'");
    return false;
}
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの「[PutObjectLockConfiguration](#)」を参照してください。

## CLI

### AWS CLI

バケットにオブジェクトロック設定を設定するには

次の put-object-lock-configuration 例では、指定したバケットに 50 日間のオブジェクトロックを設定します。

```
aws s3api put-object-lock-configuration \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":
{ "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 } } }'
```

このコマンドでは何も出力されません。

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

## Go

## SDK for Go V2

 Note

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

バケットのオブジェクトロック設定を指定します。

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// EnableObjectLockOnBucket enables object locking on an existing bucket.  
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket  
    string) error {  
    // Versioning must be enabled on the bucket before object locking is enabled.  
    verInput := &s3.PutBucketVersioningInput{  
        Bucket: aws.String(bucket),  
        VersioningConfiguration: &types.VersioningConfiguration{
```

```
        MFADelete: types.MFADeleteDisabled,
        Status:    types.BucketVersioningStatusEnabled,
    },
}
_, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
    return err
}

input := &s3.PutObjectLockConfigurationInput{
    Bucket: aws.String(bucket),
    ObjectLockConfiguration: &types.ObjectLockConfiguration{
        ObjectLockEnabled: types.ObjectLockEnabledEnabled,
    },
}
_, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
}

return err
}
```

バケットのデフォルトの保存期間を設定します。

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
```

```
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// ModifyDefaultBucketRetention modifies the default retention period of an
// existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }

    _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }
}
```

```
    return err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

バケットのオブジェクトロック設定を指定します。

```
// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        // Enable versioning on the bucket.
        getClient().putBucketVersioning(putBucketVersioningRequest);
        PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
                .build())
    }
```

```
        .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on
"+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
    }
}
```

バケットのデフォルトの保存期間を設定します。

```
// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .mfaDelete(MFADelete.DISABLED)
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention retention = DefaultRetention.builder()
        .days(1)
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(retention)
        .build();

    ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .rule(lockRule)
```



```
        .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .objectLockConfiguration(objectLockConfiguration)
        .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName
+ ".");
    }
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[PutObjectLockConfiguration](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

バケットのオブジェクトロック設定を指定します。

```
import {
    PutObjectLockConfigurationCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Enable S3 Object Lock for an Amazon S3 bucket.
 * After you enable Object Lock on a bucket, you can't
 * disable Object Lock or suspend versioning for that bucket.
 * @param {{ bucketName: string, enabled: boolean }}
 */
```

```
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
    bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
  });

  try {
    await client.send(command);
    console.log(`Object Lock for "${bucketName}" enabled.`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
        bucket "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
        bucket "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
```

```
        type: "string",
        required: true,
      },
    ];
    const results = parseArgs({ options });
    const { errors } = validateArgs({ options }, results);
    return { errors, results };
  };

  if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
      main(results.values);
    } else {
      console.error(errors.join("\n"));
    }
  }
}
```

バケットのデフォルトの保存期間を設定します。

```
import {
  PutObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Change the default retention settings for an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, retentionDays: string }}
 */
export const main = async ({ bucketName, retentionDays }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: bucketName,
        // The Object Lock configuration that you want to apply to the specified
        bucket.
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",

```

```

        Rule: {
            // The default Object Lock retention mode and period that you want to
            apply
            // to new objects placed in the specified bucket. Bucket settings
            require
            // both a mode and a period. The period can be either Days or Years
            but
            // you must select one.
            DefaultRetention: {
                // In governance mode, users can't overwrite or delete an object
                version
                // or alter its lock settings unless they have special permissions.
                With
                // governance mode, you protect objects against being deleted by
                most users,
                // but you can still grant some users permission to alter the
                retention settings
                // or delete the objects if necessary.
                Mode: "GOVERNANCE",
                Days: Number.parseInt(retentionDays),
            },
        },
    },
    },
    })),
);
console.log(
    `Set default retention mode to "GOVERNANCE" with a retention period of
    ${retentionDays} day(s).`,
);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while setting the default object retention for a bucket.
            The bucket doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while setting the default object retention for a bucket.
            ${caught.name}: ${caught.message}`,
        );
    } else {

```

```
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
        retentionDays: {
            type: "string",
            required: true,
        },
    };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

オブジェクトロック設定を適用します。

```
s3_client.put_object_lock_configuration(  
    Bucket=bucket,  
    ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":  
{}},  
)
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.  
    " Example: Enable object lock with default retention  
    " iv_enabled = 'Enabled'  
    lo_s3->putobjectlockconfiguration(  
        iv_bucket = iv_bucket_name  
        io_objectlockconfiguration = NEW /aws1/cl_s3_objectlockconf(  
            iv_objectlockenabled = iv_enabled ) ).  
    MESSAGE 'Object lock configuration set.' TYPE 'I'.
```

```
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **PutObjectRetention** を使用する

次のサンプルコードは、PutObjectRetention を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [Amazon S3 オブジェクトをロックする](#)

.NET

SDK for .NET

#### Note

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

```
/// <summary>  
/// Set or modify a retention period on an object in an S3 bucket.  
/// </summary>  
/// <param name="bucketName">The bucket of the object.</param>  
/// <param name="objectKey">The key of the object.</param>  
/// <param name="retention">The retention mode.</param>  
/// <param name="retainUntilDate">The date retention expires.</param>  
/// <returns>True if successful.</returns>  
public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
```

```
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }
}
```

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

## CLI

### AWS CLI

オブジェクトのオブジェクト保持設定を設定するには

次の `put-object-retention` の例では、指定されたオブジェクトのオブジェクト保持設定を 2025-01-01 まで設定します。

```
aws s3api put-object-retention \
```



```
--bucket amzn-s3-demo-bucket-with-object-lock \  
--key doc1.rtf \  
--retention '{ "Mode": "GOVERNANCE", "RetainUntilDate":  
"2025-01-01T00:00:00" }'
```

このコマンドでは何も出力されません。

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

## Go

### SDK for Go V2

#### Note

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

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}
```

```
// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
    string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
    error {
    input := &s3.PutObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        Retention: &types.ObjectLockRetention{
            Mode:          retentionMode,
            RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
        },
        BypassGovernanceRetention: aws.Bool(true),
    }

    _, err := actor.S3Client.PutObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time
    zone.
    ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);

    // Print the formatted date string.
    System.out.println("Formatted Date: " + humanReadableDate);
    ObjectLockRetention retention = ObjectLockRetention.builder()
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .retainUntilDate(futureInstant)
        .build();

    PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .retention(retention)
        .build();

    getClient().putObjectRetention(retentionRequest);
    System.out.println("Set retention for "+objectKey+" in "+bucketName+"
until "+ humanReadableDate +".");
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[PutObjectRetention](#)」を参照してください。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

```
import {
  PutObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Place a 24-hour retention period on an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: key,
    BypassGovernanceRetention: false,
    Retention: {
      // In governance mode, users can't overwrite or delete an object version
      // or alter its lock settings unless they have special permissions. With
      // governance mode, you protect objects against being deleted by most
      users,
      // but you can still grant some users permission to alter the retention
      settings
      // or delete the objects if necessary.
      Mode: "GOVERNANCE",
      RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
    },
  });

  try {
    await client.send(command);
    console.log("Object Retention settings updated.");
  }
```

```
    } catch (caught) {
      if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
      ) {
        console.error(
          `Error from S3 while modifying the governance mode and retention period
on an object. The bucket doesn't exist.`,
        );
      } else if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while modifying the governance mode and retention period
on an object. ${caught.name}: ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
```

```
const { errors, results } = loadArgs();
if (!errors) {
  main(results.values);
} else {
  console.error(errors.join("\n"));
}
}
```

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

## PowerShell

### Tools for PowerShell V4

例 1: このコマンドは、指定した S3 バケット内の「testfile.txt」オブジェクトの期限日「2019 年 12 月 31 日 00:00:00」までガバナンス保持モードを有効にします。

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V4)」の「[PutObjectRetention](#)」を参照してください。

### Tools for PowerShell V5

例 1: このコマンドは、指定した S3 バケット内の「testfile.txt」オブジェクトの期限日「2019 年 12 月 31 日 00:00:00」までガバナンス保持モードを有効にします。

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- API の詳細については、「AWS Tools for PowerShell コマンドレットリファレンス (V5)」の「[PutObjectRetention](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

オブジェクト保持を適用します。

```
s3_client.put_object_retention(  
    Bucket=bucket,  
    Key=key,  
    VersionId=version_id,  
    Retention={"Mode": "GOVERNANCE", "RetainUntilDate":  
far_future_date},  
    BypassGovernanceRetention=True,  
)
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.  
    " Example: Set retention mode to GOVERNANCE for 30 days  
    " iv_mode = 'GOVERNANCE'  
    " iv_retain_date should be a timestamp in the future  
    lo_s3->putobjectretention(  

```

```
iv_bucket = iv_bucket_name
iv_key = iv_object_key
io_retention = NEW /aws1/cl_s3_objectlockret(
    iv_mode = iv_mode
    iv_retainuntildate = iv_retain_date )
iv_bypassgovernanceretention = abap_true ).
MESSAGE 'Object retention set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **RestoreObject** を使用する

次のサンプルコードは、RestoreObject を使用する方法を説明しています。

.NET

SDK for .NET

 Note

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

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
```



```

    /// <summary>
    /// This example shows how to restore an archived object in an Amazon
    /// Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class RestoreArchivedObject
    {
        public static void Main()
        {
            string bucketName = "amzn-s3-demo-bucket";
            string objectKey = "archived-object.txt";

            // Specify your bucket region (an example region is shown).
            RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

            IAmazonS3 client = new AmazonS3Client(bucketRegion);
            RestoreObjectAsync(client, bucketName, objectKey).Wait();
        }

        /// <summary>
        /// This method restores an archived object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// RestoreObjectAsync.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object was located before it was archived.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object to restore.</param>
        public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
        {
            try
            {
                var restoreRequest = new RestoreObjectRequest
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Days = 2,
                };
                RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

                // Check the status of the restoration.
                await CheckRestorationStatusAsync(client, bucketName, objectKey);
            }
            catch { }
        }
    }

```

```

    }
    catch (AmazonS3Exception amazonS3Exception)
    {
        Console.WriteLine($"Error: {amazonS3Exception.Message}");
    }
}

/// <summary>
/// This method retrieves the status of the object's restoration.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// GetObjectMetadataAsync.</param>
/// <param name="bucketName">A string representing the name of the Amazon
/// S3 bucket which contains the archived object.</param>
/// <param name="objectKey">A string representing the name of the
/// archived object you want to restore.</param>
public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
{
    GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
    {
        BucketName = bucketName,
        Key = objectKey,
    };

    GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

    var restStatus = response.RestoreInProgress ? "in-progress" :
"finished or failed";
    Console.WriteLine($"Restoration status: {restStatus}");
}
}

```

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

## CLI

### AWS CLI

オブジェクトの復元リクエストを作成するには

次の `restore-object` の例では、指定された Amazon S3 Glacier オブジェクトをバケット `my-glacier-bucket` に 10 日間復元します。

```
aws s3api restore-object \  
  --bucket my-glacier-bucket \  
  --key doc1.rtf \  
  --restore-request Days=10
```

このコマンドでは何も出力されません。

- API の詳細については、AWS CLI コマンドリファレンスの「[RestoreObject](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.RestoreRequest;  
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;  
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.Tier;  
  
/*  
 * For more information about restoring an object, see "Restoring an archived  
 * object" at  
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
```

```
*
* 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 RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value of
Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
obtain this value from the AWS Management Console).\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    /**
     * Restores an S3 object from the Glacier storage class.
     */
}
```

```

    * @param s3                an instance of the {@link S3Client} to be used
    for interacting with Amazon S3
    * @param bucketName        the name of the S3 bucket where the object is
    stored
    * @param keyName            the key (object name) of the S3 object to be
    restored
    * @param expectedBucketOwner the AWS account ID of the expected bucket
    owner
    */
    public static void restoreS3Object(S3Client s3, String bucketName, String
    keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                .days(10)

            .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                .expectedBucketOwner(expectedBucketOwner)
                .bucket(bucketName)
                .key(keyName)
                .restoreRequest(restoreRequest)
                .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK または CLI で **SelectObjectContent** を使用する

次のサンプルコードは、SelectObjectContent を使用方法を説明しています。

### CLI

#### AWS CLI

SQL ステートメントに基づいて Amazon S3 オブジェクトの内容をフィルタリングするには

次の select-object-content 例では、指定された SQL ステートメントで my-data-file.csv オブジェクトをフィルタリングし、出力をファイルに送信します。

```
aws s3api select-object-content \
  --bucket amzn-s3-demo-bucket \
  --key my-data-file.csv \
  --expression "select * from s3object limit 100" \
  --expression-type 'SQL' \
  --input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \
  --output-serialization '{"CSV": {}}' "output.csv"
```

このコマンドでは何も出力されません。

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

### Java

#### SDK for Java 2.x

##### Note

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

次の例は、JSON オブジェクトを使用したクエリを示しています。[完全な例](#)では、CSV オブジェクトの使用も示しています。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
    software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
        LoggerFactory.getLogger(SelectObjectContentExample.class);
    static String BUCKET_NAME = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
        master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
        countries/master/dist/countries.json";

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

```
SelectObjectContentExample selectObjectContentExample = new
SelectObjectContentExample();
try {
    SelectObjectContentExample.setUp();
    selectObjectContentExample.runSelectObjectContentMethodForJSON();
    selectObjectContentExample.runSelectObjectContentMethodForCSV();
} catch (SdkException e) {
    logger.error(e.getMessage(), e);
    System.exit(1);
} finally {
    SelectObjectContentExample.tearDown();
}

}

EventStreamInfo runSelectObjectContentMethodForJSON() {
    // Set up request parameters.
    final String queryExpression = "select * from s3object[*][*] c where
c.area < 350000";
    final String fileType = FILE_JSON;

    InputSerialization inputSerialization = InputSerialization.builder()
        .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
        .compressionType(CompressionType.NONE)
        .build();

    OutputSerialization outputSerialization = OutputSerialization.builder()
        .json(JSONOutput.builder().recordDelimiter(null).build())
        .build();

    // Build the SelectObjectContentRequest.
    SelectObjectContentRequest select = SelectObjectContentRequest.builder()
        .bucket(BUCKET_NAME)
        .key(FILE_JSON)
        .expression(queryExpression)
        .expressionType(ExpressionType.SQL)
        .inputSerialization(inputSerialization)
        .outputSerialization(outputSerialization)
        .build();

    EventStreamInfo eventStreamInfo = new EventStreamInfo();
    // Call the selectObjectContent method with the request and a response
    handler.
    // Supply an EventStreamInfo object to the response handler to gather
    records and information from the response.
```



```

        s3AsyncClient.selectObjectContent(select,
        buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
->
            record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times",
fileType, eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler
    buildResponseHandler(EventStreamInfo eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
        information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
        SelectObjectContentResponseHandler.Visitor.builder()
            .onRecords(r -> {
                logger.info("Record event received.");
                eventStreamInfo.addRecord(r.payload().asUtf8String());
                eventStreamInfo.incrementOnRecordsCalled();
            })
            .onCont(ce -> {
                logger.info("Continuation event received.");
                eventStreamInfo.incrementContinuationEvents();
            })
            .onProgress(pe -> {
                Progress progress = pe.details();
                logger.info("Progress event received:\n bytesScanned:
{} \n bytesProcessed: {} \n bytesReturned: {}",
                    progress.bytesScanned(),
                    progress.bytesProcessed(),
                    progress.bytesReturned());
            })
            .onEnd(ee -> logger.info("End event received. "))
            .onStats(se -> {
                logger.info("Stats event received.");
            })
    }

```

```
        eventStreamInfo.addStats(se.details());
    })
    .build();

    // Build the SelectObjectContentResponseHandler with the visitor that
    // processes the stream.
    return SelectObjectContentResponseHandler.builder()
        .subscriber(visitor).build();
}

// The EventStreamInfo class is used to store information gathered while
// processing the response stream.
static class EventStreamInfo {
    private final List<String> records = new ArrayList<>();
    private Integer countOnRecordsCalled = 0;
    private Integer countContinuationEvents = 0;
    private Stats stats;

    void incrementOnRecordsCalled() {
        countOnRecordsCalled++;
    }

    void incrementContinuationEvents() {
        countContinuationEvents++;
    }

    void addRecord(String record) {
        records.add(record);
    }

    void addStats(Stats stats) {
        this.stats = stats;
    }

    public List<String> getRecords() {
        return records;
    }

    public Integer getCountOnRecordsCalled() {
        return countOnRecordsCalled;
    }

    public Integer getCountContinuationEvents() {
        return countContinuationEvents;
    }
}
```

```
    }

    public Stats getStats() {
        return stats;
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **UploadPart** を使用する

次のサンプルコードは、UploadPart を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [チェックサムの使用](#)
- [Amazon S3 オブジェクトの整合性の操作](#)

C++

SDK for C++

 Note

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

```
//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
```

```

    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
checksum algorithm,
                                ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
&bucket,
                                                    const Aws::String &key,
                                                    const Aws::String
&uploadID,
                                                    int partNumber,

Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                    const Aws::String
&calculatedHash,
                                                    const
std::shared_ptr<Aws::IOStream> &body,
                                                    const Aws::S3::S3Client
&client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);

```

```

        break;
    case Aws::S3::Model::ChecksumAlgorithm::SHA1:
        request.SetChecksumSHA1(calculatedHash);
        break;
    case Aws::S3::Model::ChecksumAlgorithm::SHA256:
        request.SetChecksumSHA256(calculatedHash);
        break;
    }
}

return client.UploadPart(request);
}

```

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

## CLI

### AWS CLI

次のコマンドは、create-multipart-upload コマンドで開始されたマルチパートアップロードの最初の部分をアップロードします。

```
aws s3api upload-part --bucket amzn-s3-demo-bucket --key
'multipart/01' --part-number 1 --body part01 --upload-id
'dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P'
```

body オプションは、アップロードするローカルファイルの名前またはパスを指定します (file://プレフィックスは使用しないでください)。各パートの最小サイズは 5 MB です。アップロード ID は create-multipart-upload によって返され、list-multipart-uploads を使用して取得することもできます。バケットとキーは、マルチパートアップロードの作成時に指定されます。

出力:

```
{
  "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

後でできるように、各パートの ETag 値を保存します。これらはマルチパートアップロードを完了するために必要です。

- API の詳細については、AWS CLI コマンドリファレンスの「[UploadPart](#)」を参照してください。

## Rust

### SDK for Rust

#### Note

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

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send();
```

```
        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
            .build(),
    );
}
```

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **UploadPartCopy** を使用する

次のサンプルコードは、UploadPartCopy を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [マルチパートコピーを実行する](#)

CLI

AWS CLI

既存のオブジェクトからデータソースとしてデータをコピーしてオブジェクトの一部をアップロードするには

次の upload-part-copy 例では、既存のオブジェクトからデータをデータソースとしてコピーし、一部をアップロードします。

```
aws s3api upload-part-copy \
  --bucket amzn-s3-demo-bucket \
  --key "Map_Data_June.mp4" \
  --copy-source "amzn-s3-demo-bucket/copy_of_Map_Data_June.mp4" \
  --part-number 1 \
  --upload-
id "bq0tdE1CDpWQYRPLHuNG50xAT6pA5D.m_RiBy0gg0H6b13pVRY7QjvL1f75iFdJqp_2wztk5hvpUM2SesXgrz"
```

出力:

```
{
  "CopyPartResult": {
    "LastModified": "2019-12-13T23:16:03.000Z",
    "ETag": "\"711470fc377698c393d94aed6305e245\""
  }
}
```



```
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(toBucket)
        .key(key)
        .build();

    getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
        .thenApply(createMultipartUploadResponse -> {
            String uploadId = createMultipartUploadResponse.uploadId();
            System.out.println("Upload ID: " + uploadId);

            UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                .sourceBucket(bucketName)
                .destinationBucket(toBucket)
                .sourceKey(key)
                .destinationKey(key)
                .uploadId(uploadId) // Use the valid uploadId.
                .partNumber(1) // Ensure the part number is correct.
                .copySourceRange("bytes=0-1023") // Adjust range as needed
                .build();

            return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
        })
}
```

```
        .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
        .whenComplete((uploadPartCopyResponse, exception) -> {
            if (exception != null) {
                // Handle any exceptions.
                logger.error("Error during upload part copy: " +
exception.getMessage());
            } else {
                // Successfully completed the upload part copy.
                System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
            }
        });
    return null;
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用した Amazon S3 のシナリオ

次のコード例は、AWS SDK を使用して Amazon S3 で一般的なシナリオを実装する方法を示しています。これらのシナリオは、Amazon S3 内で複数の関数を呼び出すか、その他の AWS のサービスと組み合わせて、特定のタスクを実行する方法を説明しています。各シナリオには、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

シナリオは、サービスアクションをコンテキストで理解するのに役立つ中級レベルの経験を対象としています。

### 例

- [バケットが存在するかどうかを確認する](#)
- [オブジェクトが存在するかどうかを確認します。](#)
- [AWS SDK を使用して、テキストを音声に変換し、テキストに戻す](#)
- [AWS SDK を使用して Amazon S3 の署名付き URL を作成する](#)

- [ユーザーがラベルを使用して写真を管理できる写真アセット管理アプリケーションの作成](#)
- [AWS SDK を使用して Amazon S3 オブジェクトを一覧表示するウェブページ](#)
- [Amazon Textract エクスプローラーアプリケーションを作成する](#)
- [AWS SDK を使用して、指定した Amazon S3 バケットからすべてのオブジェクトを削除します。](#)
- [AWS SDK を使用して Amazon S3 への不完全なマルチパートアップロードを削除する](#)
- [AWS SDK を使用して、Amazon Rekognition で画像内の PPE を検出する](#)
- [AWS SDK を使用して、画像から抽出されたテキスト内のエンティティを検出する](#)
- [AWS SDK を使用して、画像内の顔を検出する](#)
- [AWS SDK を使用して、Amazon Rekognition で画像内のオブジェクトを検出する](#)
- [AWS SDK を使用して、Amazon Rekognition で、動画内の人やオブジェクトを検出する](#)
- [Amazon Simple Storage Service \(Amazon S3\) バケットから S3 の「ディレクトリ」をダウンロードする](#)
- [Amazon Simple Storage Service \(Amazon S3\) バケットからすべてのオブジェクトを、ローカルディレクトリにダウンロードする](#)
- [サイズが不明なストリームを、AWS SDK を使用して Amazon S3 オブジェクトからダウンロードする](#)
- [AWS SDK を使用してマルチリージョンアクセスポイントから Amazon S3 オブジェクトを取得する](#)
- [AWS SDK を使用して、Amazon S3 バケットから If-Modified-Since を指定してオブジェクトを取得する](#)
- [AWS SDK を使用して Amazon S3 オブジェクトの暗号化の使用を開始する](#)
- [AWS SDK を使用して Amazon S3 オブジェクトのタグの使用を開始する](#)
- [ビッグデータ処理クラスターの使用開始](#)
- [設定管理の使用開始](#)
- [ドキュメントテキスト抽出の使用開始](#)
- [機械学習特徴量ストアの使用開始](#)
- [オブジェクトストレージの使用開始](#)
- [クエリ分析の使用開始](#)
- [AWS SDK を使用して Amazon S3 オブジェクトロック機能进行操作する](#)
- [AWS SDK を使用して Amazon S3 の条件付きリクエストを行う](#)

- [AWS SDK を使用して Amazon S3 バケットのアクセスコントロールリスト \(ACL\) を管理する](#)
- [AWS SDK で、Amazon S3 を使用して大量の Amazon SQS メッセージを管理する](#)
- [AWS SDK を使用して、バージョン管理されている Amazon S3 オブジェクトを Lambda 関数でバッチで管理する](#)
- [AWS SDK を使用して Amazon S3 の URI を解析する](#)
- [AWS SDK を使用して Amazon S3 オブジェクトのマルチパートコピーを実行する](#)
- [AWS SDK を使用して Amazon S3 イベント通知を受信して処理する](#)
- [AWS SDK を使用して、EXIF とその他の画像情報を保存する](#)
- [AWS SDK を使用して S3 イベント通知を Amazon EventBridge に送信する](#)
- [AWS SDK を使用して Amazon S3 オブジェクトのアップロードまたはダウンロードを追跡する](#)
- [S3 Object Lambda でアプリケーションのデータを変換する](#)
- [AWS SDK による単体テストと統合テストのアプローチ例](#)
- [ローカルディレクトリを Amazon Simple Storage Service \(Amazon S3\) バケットに、再帰的にアップロードする](#)
- [AWS SDK を使用して Amazon S3 との間で大きなファイルをアップロードまたはダウンロードする](#)
- [サイズが不明なストリームを AWS SDK を使用して Amazon S3 オブジェクトにアップロードする](#)
- [チェックサムを使用した AWS SDK での Amazon S3 オブジェクトの操作](#)
- [AWS SDK を使用して Amazon S3 オブジェクトの整合性機能进行操作する](#)
- [AWS SDK を使用して、Amazon S3 のバージョン管理されたオブジェクト进行操作する](#)

## バケットが存在するかどうかを確認する

次のコード例は、バケットが存在するかどうかを確認する方法を示しています。

Java

SDK for Java 2.x

### Note

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

SDK for Java V1 [AmazonS3Client#doesBucketExistV2 \(文字列\)](#) メソッドの代わりに、次の `doesBucketExists` メソッドを使用できます。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;

public class DoesBucketExist {
    private static final Logger logger =
        LoggerFactory.getLogger(DoesBucketExist.class);

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with the
        bucket name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName,
        s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
     * global namespace; use this method to
     * 

- determine if a specified bucket name already exists, and therefore can't
        be used to create a new bucket.


     * <p>
     * Internally this method uses the <a
     * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
     * services/s3/
     * S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
     * a>
     * operation to determine whether the bucket exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's <a
```

```

    * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
    *
    * @param bucketName    The name of the bucket to check.
    * @param s3SyncClient An <code>S3Client</code> instance. The method checks
for the bucket in the AWS Region
    *
    *                      configured on the instance.
    * @return The value true if the specified bucket exists in Amazon S3; the
value false if there is no bucket in
    *
    *          Amazon S3 with that name.
    */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
empty string.", "");
            s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
            return true;
        } catch (AwsServiceException ase) {
            // A redirect error or an AccessDenied exception means the bucket
exists but it's not in this region
            // or we don't have permissions to it.
            if ((ase.statusCode() == HttpStatus.MOVED_PERMANENTLY) ||
"AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
                return true;
            }
            if (ase.statusCode() == HttpStatus.NOT_FOUND) {
                return false;
            }
            throw ase;
        }
    }
}

```

- APIの詳細については、AWS SDK for Java 2.x API リファレンスの「[GetBucketAcl](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

オブジェクトが存在するかどうかを確認します。

次のコード例は、オブジェクトが存在するかどうかを確認する方法を示しています。

Java

SDK for Java 2.x

 Note

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

SDK for Java V1 [AmazonS3Client#doesObjectExist\(文字列、文字列\)](#) メソッドの代わりに、次の `doesObjectExist` メソッドを使用できます。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;
import software.amazon.awssdk.utils.Validate;

public class DoesObjectExist {
    private static final Logger logger =
        LoggerFactory.getLogger(DoesObjectExist.class);

    public static void main(String[] args) {
        DoesObjectExist doesObjectExist = new DoesObjectExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
        final String key = "my-key"; // Replace with your object key.

        boolean exists = doesObjectExist.doesObjectExist(bucketName, key,
            s3SyncClient);
        logger.info("Object exists: {}", exists);
    }
}
```

```

/**
 * Checks if the specified object exists in the specified bucket.
 * <p>
 * Internally this method uses the
 * <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#headObject(java.util.function.Consumer)">S3Client.headObject</a>
 * operation to determine whether the object exists.
 * <p>
 * This method is equivalent to the AWS SDK for Java V1's
 * <a href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/
amazonaws/services/s3/AmazonS3Client.html#doesObjectExist-java.lang.String-
java.lang.String-">AmazonS3Client#doesObjectExist(String, String)</a>.
 * <p>
 * <b>Note:</b> This method returns {@code false} only when S3 responds with
404 (NoSuchKey). If the caller
 * does not have {@code s3:ListBucket} permission on the bucket, S3 may
return 403 instead of 404 for
 * non-existent objects, which will be thrown as an exception.
 *
 * @param bucketName The name of the bucket containing the object.
 * @param key The key of the object to check.
 * @param s3SyncClient An {@code S3Client} instance.
 * @return {@code true} if the object exists; {@code false} if it does not
exist.
 */
public boolean doesObjectExist(String bucketName, String key, S3Client
s3SyncClient) {
    try {
        Validate.notEmpty(bucketName, "The bucket name must not be null or an
empty string.", "");
        Validate.notEmpty(key, "The object key must not be null or an empty
string.", "");
        s3SyncClient.headObject(HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build());
        return true;
    } catch (NoSuchKeyException e) {
        return false;
    }
}
}

```



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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して、テキストを音声に変換し、テキストに戻す

次のコード例は、以下の操作方法を示しています。

- Amazon Polly を使用して、プレーンテキスト (UTF-8) 入力ファイルを音声ファイルに合成します。
- Amazon S3 バケットに音声ファイルをアップロードします。
- Amazon Transcribe を使用して、音声ファイルをテキストに変換します。
- テキストを表示します。

### Rust

#### SDK for Rust

Amazon Polly を使用して、プレーンテキスト (UTF-8) 入力ファイルを音声ファイルに合成し、音声ファイルを Amazon S3 バケットにアップロードし、Amazon Transcribe を使用して音声ファイルをテキストに変換し、テキストを表示します。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

#### この例で使用されているサービス

- Amazon Polly
- Amazon S3
- Amazon Transcribe

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 の署名付き URL を作成する

次のコード例は、Amazon S3 の署名付き URL を作成し、オブジェクトをアップロードする方法を示しています。

.NET

SDK for .NET

### Note

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

Amazon S3 アクションを期間限定で実行できる署名付き URL を生成します。

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

public class GenPresignedUrl
{
    public static void Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string objectKey = "sample.txt";

        // Specify how long the presigned URL lasts, in hours
        const double timeoutDuration = 12;

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
        // pass the Region to the constructor for the client. For
        // example: new AmazonS3Client(RegionEndpoint.USEast1);
```

```

        // If using the Region us-east-1, and server-side encryption with AWS
        KMS, you must specify Signature Version 4.
        // Region us-east-1 defaults to Signature Version 2 unless explicitly
        set to Version 4 as shown below.
        // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
        // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
        TAWSConfigsS3.html
        AWSConfigsS3.UseSignatureVersion4 = true;
        IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

        string urlString = GeneratePresignedURL(s3Client, bucketName,
objectKey, timeoutDuration);
        Console.WriteLine($"The generated URL is: {urlString}.");
    }

    /// <summary>
    /// Generate a presigned URL that can be used to access the file named
    /// in the objectKey parameter for the amount of time specified in the
    /// duration parameter.
    /// </summary>
    /// <param name="client">An initialized S3 client object used to call
    /// the GetPresignedUrl method.</param>
    /// <param name="bucketName">The name of the S3 bucket containing the
    /// object for which to create the presigned URL.</param>
    /// <param name="objectKey">The name of the object to access with the
    /// presigned URL.</param>
    /// <param name="duration">The length of time for which the presigned
    /// URL will be valid.</param>
    /// <returns>A string representing the generated presigned URL.</returns>
    public static string GeneratePresignedURL(IAmazonS3 client, string
bucketName, string objectKey, double duration)
    {
        string urlString = string.Empty;
        try
        {
            var request = new GetPreSignedUrlRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Expires = DateTime.UtcNow.AddHours(duration),
            };
            urlString = client.GetPreSignedURL(request);
        }
    }

```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error: '{ex.Message}'");
        }

        return urlString;
    }
}
```

署名済み URL を生成し、その URL を使用してアップロードを実行します。

```
using System;
using System.IO;
using System.Net.Http;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket using a presigned URL. The code first
/// creates a presigned URL and then uses it to upload an object to an
/// Amazon S3 bucket using that URL.
/// </summary>
public class UploadUsingPresignedURL
{
    private static HttpClient httpClient = new HttpClient();

    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "samplefile.txt";
        string filePath = $"source\\{keyName}";

        // Specify how long the signed URL will be valid in hours.
        double timeoutDuration = 12;

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
        // pass the Region to the constructor for the client. For
```

```
// example: new AmazonS3Client(RegionEndpoint.USEast1);

// If using the Region us-east-1, and server-side encryption with AWS
KMS, you must specify Signature Version 4.
// Region us-east-1 defaults to Signature Version 2 unless explicitly
set to Version 4 as shown below.
// For more details, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingAWSSDK.html#specify-signature-version
// and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/TAWSConfigsS3.html
AWSConfigsS3.UseSignatureVersion4 = true;
IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);

var url = GeneratePreSignedURL(client, bucketName, keyName,
timeoutDuration);
var success = await UploadObject(filePath, url);

if (success)
{
    Console.WriteLine("Upload succeeded.");
}
else
{
    Console.WriteLine("Upload failed.");
}
}

/// <summary>
/// Uploads an object to an Amazon S3 bucket using the presigned URL
passed in
/// the url parameter.
/// </summary>
/// <param name="filePath">The path (including file name) to the local
/// file you want to upload.</param>
/// <param name="url">The presigned URL that will be used to upload the
/// file to the Amazon S3 bucket.</param>
/// <returns>A Boolean value indicating the success or failure of the
/// operation, based on the HttpResponseMessage.</returns>
public static async Task<bool> UploadObject(string filePath, string url)
{
    using var streamContent = new StreamContent(
        new FileStream(filePath, FileMode.Open, FileAccess.Read));

    var response = await httpClient.PutAsync(url, streamContent);
}
```

```
        return response.IsSuccessStatusCode;
    }

    /// <summary>
    /// Generates a presigned URL which will be used to upload an object to
    /// an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetPreSignedURL.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
    /// presigned URL will point.</param>
    /// <param name="objectKey">The name of the file that will be uploaded.</
param>
    /// <param name="duration">How long (in hours) the presigned URL will
    /// be valid.</param>
    /// <returns>The generated URL.</returns>
    public static string GeneratePreSignedURL(
        IAmazonS3 client,
        string bucketName,
        string objectKey,
        double duration)
    {
        var request = new GetPreSignedUrlRequest
        {
            BucketName = bucketName,
            Key = objectKey,
            Verb = HttpVerb.PUT,
            Expires = DateTime.UtcNow.AddHours(duration),
        };

        string url = client.GetPreSignedURL(request);
        return url;
    }
}
```

## SDK for .NET (v4)

 Note

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

ブラウザから直接アップロードするための、署名付き POST URL を作成して使用します。

```
/// <summary>
/// Scenario demonstrating the complete workflow for presigned POST URLs:
/// 1. Create an S3 bucket
/// 2. Create a presigned POST URL
/// 3. Upload a file using the presigned POST URL
/// 4. Clean up resources
/// </summary>
public class CreatePresignedPostBasics
{
    public static ILogger<CreatePresignedPostBasics> _logger = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static UiMethods _uiMethods = null!;
    public static IHttpClientFactory _httpClientFactory = null!;
    public static bool _isInteractive = true;
    public static string? _bucketName;
    public static string? _objectKey;

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host">The IHost instance.</param>
    public static void SetupServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<CreatePresignedPostBasics>(loggerFactory);

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();
        _httpClientFactory =
            host.Services.GetRequiredService<IHttpClientFactory>();
        _uiMethods = new UiMethods();
    }
}
```

```
}

/// <summary>
/// Perform the actions defined for the Amazon S3 Presigned POST scenario.
/// </summary>
/// <param name="args">Command line arguments.</param>
/// <returns>A Task object.</returns>
public static async Task Main(string[] args)
{
    _isInteractive = !args.Contains("--non-interactive");

    // Set up dependency injection for Amazon S3
    using var host =
        Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddHttpClient()
            )
            .Build();

    SetupServices(host);

    try
    {
        // Display overview
        _uiMethods.DisplayOverview();
        _uiMethods.PressEnter(_isInteractive);

        // Step 1: Create bucket
        await CreateBucketAsync();
        _uiMethods.PressEnter(_isInteractive);

        // Step 2: Create presigned URL
        _uiMethods.DisplayTitle("Step 2: Create presigned POST URL");
        var response = await CreatePresignedPostAsync();
        _uiMethods.PressEnter(_isInteractive);

        // Step 3: Display URL and fields
        _uiMethods.DisplayTitle("Step 3: Presigned POST URL details");
        DisplayPresignedPostFields(response);
        _uiMethods.PressEnter(_isInteractive);

        // Step 4: Upload file
```



```

        _uiMethods.DisplayTitle("Step 4: Upload test file using presigned
POST URL");
        await UploadFileAsync(response);
        _uiMethods.PressEnter(_isInteractive);

        // Step 5: Verify file exists
        await VerifyFileExistsAsync();
        _uiMethods.PressEnter(_isInteractive);

        // Step 6: Cleanup
        _uiMethods.DisplayTitle("Step 6: Clean up resources");
        await CleanupAsync();

        _uiMethods.DisplayTitle("S3 Presigned POST Scenario completed
successfully!");
        _uiMethods.PressEnter(_isInteractive);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error in scenario");
        Console.WriteLine($"Error: {ex.Message}");

        // Attempt cleanup if there was an error
        if (!string.IsNullOrEmpty(_bucketName))
        {
            _uiMethods.DisplayTitle("Cleaning up resources after error");
            await _s3Wrapper.DeleteBucketAsync(_bucketName);
            Console.WriteLine($"Cleaned up bucket: {_bucketName}");
        }
    }
}

/// <summary>
/// Create an S3 bucket for the scenario.
/// </summary>
private static async Task CreateBucketAsync()
{
    _uiMethods.DisplayTitle("Step 1: Create an S3 bucket");

    // Generate a default bucket name for the scenario
    var defaultBucketName = $"presigned-post-demo-
{DateTime.Now:yyyyMMddHHmmss}".ToLower();

    // Prompt user for bucket name or use default in non-interactive mode

```

```
        _bucketName = _uiMethods.GetUserInput(
            $"Enter S3 bucket name (or press Enter for '{defaultBucketName}'):",
            defaultBucketName,
            _isInteractive);

        // Basic validation to ensure bucket name is not empty
        if (string.IsNullOrEmpty(_bucketName))
        {
            _bucketName = defaultBucketName;
        }

        Console.WriteLine($"Creating bucket: {_bucketName}");

        await _s3Wrapper.CreateBucketAsync(_bucketName);

        Console.WriteLine($"Successfully created bucket: {_bucketName}");
    }

    /// <summary>
    /// Create a presigned POST URL.
    /// </summary>
    private static async Task<CreatePresignedPostResponse>
    CreatePresignedPostAsync()
    {
        _objectKey = "example-upload.txt";
        var expiration = DateTime.UtcNow.AddMinutes(10); // Short expiration for
        the demo

        Console.WriteLine($"Creating presigned POST URL for {_bucketName}/
{_objectKey}");
        Console.WriteLine($"Expiration: {expiration} UTC");

        var s3Client = _s3Wrapper.GetS3Client();

        var response = await _s3Wrapper.CreatePresignedPostAsync(
            s3Client, _bucketName!, _objectKey, expiration);

        Console.WriteLine("Successfully created presigned POST URL");
        return response;
    }

    /// <summary>
    /// Upload a file using the presigned POST URL.
```

```
    /// </summary>
    private static async Task UploadFileAsync(CreatePresignedPostResponse
response)
    {

        // Create a temporary test file to upload
        string testFilePath = Path.GetTempFileName();
        string testContent = "This is a test file for the S3 presigned POST
scenario.";

        await File.WriteAllTextAsync(testFilePath, testContent);
        Console.WriteLine($"Created test file at: {testFilePath}");

        // Upload the file using the presigned POST URL
        Console.WriteLine("\nUploading file using the presigned POST URL...");
        var uploadResult = await UploadFileWithPresignedPostAsync(response,
testFilePath);

        // Display the upload result
        if (uploadResult.Success)
        {
            Console.WriteLine($"Upload successful! Status code:
{uploadResult.StatusCode}");
        }
        else
        {
            Console.WriteLine($"Upload failed with status code:
{uploadResult.StatusCode}");
            Console.WriteLine($"Error: {uploadResult.Response}");
            throw new Exception("File upload failed");
        }

        // Clean up the temporary file
        File.Delete(testFilePath);
        Console.WriteLine("Temporary file deleted");
    }

    /// <summary>
    /// Helper method to upload a file using a presigned POST URL.
    /// </summary>
    private static async Task<(bool Success, HttpStatusCode StatusCode, string
Response)> UploadFileWithPresignedPostAsync(
        CreatePresignedPostResponse response,
        string filePath)
```

```
{
    try
    {
        _logger.LogInformation("Uploading file {filePath} using presigned
POST URL", filePath);

        using var httpClient = _httpClientFactory.CreateClient();
        using var formContent = new MultipartFormDataContent();

        // Add all the fields from the presigned POST response
        foreach (var field in response.Fields)
        {
            formContent.Add(new StringContent(field.Value), field.Key);
        }

        // Add the file content
        var fileStream = File.OpenRead(filePath);
        var fileName = Path.GetFileName(filePath);
        var fileContent = new StreamContent(fileStream);
        fileContent.Headers.ContentType = new MediaTypeHeaderValue("text/
plain");
        formContent.Add(fileContent, "file", fileName);

        // Send the POST request
        var httpResponse = await httpClient.PostAsync(response.Url,
formContent);
        var responseContent = await httpResponse.Content.ReadAsStringAsync();

        // Log and return the result
        _logger.LogInformation("Upload completed with status code
{statusCode}", httpResponse.StatusCode);

        return (httpResponse.IsSuccessStatusCode, httpResponse.StatusCode,
responseContent);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error uploading file");
        return (false, HttpStatusCode.InternalServerError, ex.Message);
    }
}

/// <summary>
/// Verify that the uploaded file exists in the S3 bucket.
```

```
/// </summary>
private static async Task VerifyFileExistsAsync()
{
    _uiMethods.DisplayTitle("Step 5: Verify uploaded file exists");

    Console.WriteLine($"Checking if file exists at {_bucketName}/
{_objectKey}...");

    try
    {
        var metadata = await _s3Wrapper.GetObjectMetadataAsync(_bucketName!,
            _objectKey!);

        Console.WriteLine($"File verification successful! File exists in the
bucket.");
        Console.WriteLine($"File size: {metadata.ContentLength} bytes");
        Console.WriteLine($"File type: {metadata.Headers.ContentType}");
        Console.WriteLine($"Last modified: {metadata.LastModified}");
    }
    catch (AmazonS3Exception ex) when (ex.StatusCode ==
        System.Net.HttpStatusCode.NotFound)
    {
        Console.WriteLine($"Error: File was not found in the bucket.");
        throw;
    }
}

private static void DisplayPresignedPostFields(CreatePresignedPostResponse
response)
{
    Console.WriteLine($"Presigned POST URL: {response.Url}");
    Console.WriteLine("Form fields to include:");

    foreach (var field in response.Fields)
    {
        Console.WriteLine($" {field.Key}: {field.Value}");
    }
}

/// <summary>
/// Clean up resources created by the scenario.
/// </summary>
private static async Task CleanupAsync()
{

```

```
        if (!string.IsNullOrEmpty(_bucketName))
        {
            Console.WriteLine($"Deleting bucket {_bucketName} and its
contents...");
            bool result = await _s3Wrapper.DeleteBucketAsync(_bucketName);

            if (result)
            {
                Console.WriteLine("Bucket deleted successfully");
            }
            else
            {
                Console.WriteLine("Failed to delete bucket - it may have been
already deleted");
            }
        }
    }
}
```

## C++

### SDK for C++

#### Note

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

オブジェクトをダウンロードするための署名付き URL を生成します。

```
//! Routine which demonstrates creating a pre-signed URL to download an object
from an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
    \param bucketName: Name of the bucket.
    \param key: Name of an object key.
    \param expirationSeconds: Expiration in seconds for pre-signed URL.
    \param clientConfig: Aws client configuration.
    \return Aws::String: A pre-signed URL.
*/
```

```

Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String
    &bucketName,
                                                    const Aws::String &key,
                                                    uint64_t expirationSeconds,
                                                    const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_GET,
                                                    expirationSeconds);
}

```

libcurl を使用してダウンロードします。

```

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
    \param presignedURL: A pre-signed URL to get an object from a bucket.
    \param resultString: A string to hold the result.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                    Aws::String &resultString) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }
}

```

```
}

result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_URL" << std::endl;
    return false;
}

result = curl_easy_perform(curl);

if (result != CURLE_OK) {
    std::cerr << "Failed to perform CURL request" << std::endl;
    return false;
}

resultString = outWriteString.str();

if (resultString.find("<?xml") == 0) {
    std::cerr << "Failed to get object, response:\n" << resultString <<
std::endl;
    return false;
}

return true;
}
```

オブジェクトをアップロードするための署名付き URL を生成します。

```
//! Routine which demonstrates creating a pre-signed URL to upload an object to
an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
    \param bucketName: Name of the bucket.
    \param key: Name of an object key.
```



```

    \param clientConfig: Aws client configuration.
    \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String
    &bucketName,
                                                    const Aws::String &key,
                                                    uint64_t expirationSeconds,
                                                    const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_PUT,
                                                    expirationSeconds);
}

```

libcurl を使用してアップロードします。

```

static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
    \param presignedURL: A pre-signed URL to put an object in a bucket.
    \param data: Body of the putObject request.
    \return bool: Function succeeded.
*/

```

```
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                  const Aws::String &data) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }
}
```

```
result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_URL" << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
    return false;
}

result = curl_easy_perform(curl);

if (result != CURLE_OK) {
    std::cerr << "Failed to perform CURL request" << std::endl;
    return false;
}

std::string outString = outWriteString.str();
if (outString.empty()) {
    std::cout << "Successfully put object." << std::endl;
    return true;
} else {
    std::cout << "A server error was encountered, output:\n" << outString
                << std::endl;
    return false;
}
}
```

Go

SDK for Go V2

 Note

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

## S3 署名アクションをラップする関数を作成します。

```
import (
    "context"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
// actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
// S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
// client.
type Presigner struct {
    PresignClient *s3.PresignClient
}

// GetObject makes a presigned request that can be used to get an object from a
// bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
(*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignGetObject(ctx,
        &s3.GetObjectInput{
            Bucket: aws.String(bucketName),
            Key:   aws.String(objectKey),
        }, func(opts *s3.PresignOptions) {
            opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return request, err
}
```

```
}

// PutObject makes a presigned request that can be used to put an object in a
// bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
(*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignPutObject(ctx,
        &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
        }, func(opts *s3.PresignOptions) {
            opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return request, err
}

// DeleteObject makes a presigned request that can be used to delete an object
// from a bucket.
func (presigner Presigner) DeleteObject(ctx context.Context, bucketName string,
    objectKey string) (*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignDeleteObject(ctx,
        &s3.DeleteObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to delete object %v. Here's why:
        %v\n", objectKey, err)
    }
    return request, err
}
```

```
func (presigner Presigner) PresignPostObject(ctx context.Context, bucketName
    string, objectKey string, lifetimeSecs int64) (*s3.PresignedPostRequest, error)
{
    request, err := presigner.PresignClient.PresignPostObject(ctx,
    &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    }, func(options *s3.PresignPostOptions) {
        options.Expires = time.Duration(lifetimeSecs) * time.Second
    })
    if err != nil {
        log.Printf("Couldn't get a presigned post request to put %v:%v. Here's why: %v
    \n", bucketName, objectKey, err)
    }
    return request, nil
}
```

署名付き URL を生成して使用し、S3 オブジェクトをアップロード、ダウンロード、削除するインタラクティブな例を実行します。

```
import (
    "bytes"
    "context"
    "io"
    "log"
    "mime/multipart"
    "net/http"
    "os"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)
```

```
// RunPresigningScenario is an interactive example that shows you how to get
// presigned
// HTTP requests that you can use to move data into and out of Amazon Simple
// Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and
// can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local
// file to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the
// object to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
// mocked
// during testing.
func RunPresigningScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner, httpRequester IHttpRequester) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := questioner.(*demotools.MockQuestioner)
            if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
                log.Println(r)
            }
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon S3 presigning demo.")
    log.Println(strings.Repeat("-", 88))
}
```

```
s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}
presignClient := s3.NewPresignClient(s3Client)
presigner := actions.Presigner{PresignClient: presignClient}

bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
    "you own or one you want to create:", demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
    demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
    presignedPutRequest.Method,
    presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
    panic(err)
}
```



```
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Now we'll create a new request to put the same object using a
presigned post request")
questioner.Ask("Press Enter when you're ready.")
presignPostRequest, err := presigner.PresignPostObject(ctx, bucketName,
uploadKey, 60)
if err != nil {
```

```
    panic(err)
}
log.Printf("Got a presigned post request to url %v with values %v\n",
presignPostRequest.URL, presignPostRequest.Values)
log.Println("Using net/http multipart to send the request...")
uploadFile, err = os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
multiPartResponse, err := sendMultipartRequest(presignPostRequest.URL,
presignPostRequest.Values, uploadFile, uploadKey, httpRequester)
if err != nil {
    panic(err)
}
log.Printf("Presign post object %v with presigned URL returned %v.", uploadKey,
multiPartResponse.StatusCode)

log.Println("Let's presign a request to delete the object.")
questioner.Ask("Press Enter when you're ready.")
presignedDelRequest, err := presigner.DeleteObject(ctx, bucketName, uploadKey)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedDelRequest.Method,
presignedDelRequest.URL)
log.Println("Using net/http to send the request...")
delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.\n",
presignedDelRequest.Method,
uploadKey, delResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

この例で HTTP リクエストを行うために使用する HTTP リクエストラッパーを定義します。

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
// during
// unit testing.
type IHttpRequester interface {
    Get(url string) (resp *http.Response, err error)
    Post(url, contentType string, body io.Reader) (resp *http.Response, err error)
    Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
    error)
    Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
// scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
    return http.Get(url)
}

func (httpReq HttpRequester) Post(url, contentType string, body io.Reader) (resp
    *http.Response, err error) {
    postRequest, err := http.NewRequest("POST", url, body)
    if err != nil {
        return nil, err
    }
    postRequest.Header.Set("Content-Type", contentType)
    return http.DefaultClient.Do(postRequest)
}

func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
    (resp *http.Response, err error) {
    putRequest, err := http.NewRequest("PUT", url, body)
    if err != nil {
        return nil, err
    }
    putRequest.ContentLength = contentLength
    return http.DefaultClient.Do(putRequest)
}

func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
    {
    delRequest, err := http.NewRequest("DELETE", url, nil)
    if err != nil {
```

```
    return nil, err
}
return http.DefaultClient.Do(delRequest)
}
```

## Java

### SDK for Java 2.x

#### Note

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

以下は、署名付き URL を作成し、HTTP クライアントライブラリで URL を使用する 3 つの例を示しています。

- 3 つの HTTP クライアントライブラリで URL を使用する HTTP GET リクエスト
- 3 つの HTTP クライアントライブラリで URL を使用する、メタデータ付きヘッダーを持つ HTTP PUT リクエスト
- 1 つの HTTP クライアントライブラリで URL を使用する、クエリパラメータを含む HTTP PUT リクエスト

オブジェクトの署名付き URL を生成して、オブジェクトをダウンロードします (GET リクエスト)。

インポート。

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

URL を生成する。

```
/* Create a pre-signed URL to download an object in a subsequent GET request.
 */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest =
            GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL will
                expire in 10 minutes.
                .getObjectRequest(objectRequest)
                .build();
```

```
        PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]",
presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}
```

次の 3 つの方法のいずれかを使用して、オブジェクトをダウンロードします。

JDK HttpURLConnection (v1.1 以降) クラスを使用してダウンロードを行います。

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

JDK HttpClient (v1.1 以降) クラスを使用してダウンロードを行います。

```
/* Use the JDK HttpClient (since v1.1) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
```

```
        ByteArrayOutputStream byteArrayOutputStream = new
        ByteArrayOutputStream(); // Capture the response body to a byte array.

        HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
        HttpClient httpClient = HttpClient.newHttpClient();
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpResponse<InputStream> response = httpClient.send(requestBuilder
                .uri(presignedUrl.toURI())
                .GET()
                .build(),
                HttpResponse.BodyHandlers.ofInputStream());

            IoUtils.copy(response.body(), byteArrayOutputStream);

            logger.info("HTTP response code is " + response.statusCode());
        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
}
```

AWS SDK for Java SdkHttpClient クラスを使用してダウンロードを行います。

```
/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToGet(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
    ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
```

```
        HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
        response.responseBody().ifPresentOrElse(
            abortableInputStream -> {
                try {
                    IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            },
            () -> logger.error("No response body."));

        logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
    }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

アップロード用に、ヘッダーにメタデータを含む署名付き URL を生成して、ファイルをアップロードします (PUT リクエスト)。

インポート。

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
```



```
import
    software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

URL を生成する。

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName,
    Map<String, String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest =
            PutObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL
                expires in 10 minutes.
                .putObjectRequest(objectRequest)
                .build();
```

```

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

次の 3 つの方法のいずれかを使用して、オブジェクトをアップロードします。

JDK HttpURLConnection (v1.1 以降) クラスを使用してアップロードを行います。

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useHttpURLConnectionToPut(String presignedUrlString, File
fileToPut, Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-
meta-" + k, v));
        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is
8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
                buffer.clear();
            }
        } catch (IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
}

```

```
    }

    out.close();
    connection.getResponseCode();
    logger.info("HTTP response code is " + connection.getResponseCode());

} catch (S3Exception | IOException e) {
    logger.error(e.getMessage(), e);
}
}
```

JDK `HttpClient` (v11 以降) クラスを使用してアップロードを行います。

```
/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(new URL(presignedUrlString).toURI())

            .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
            .build(),
            HttpResponse.BodyHandlers.discarding());

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

AWS for Java V2 `SdkHttpClient` クラスを使用してアップロードを行います。

```
/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
```

```
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" +
k, v));
        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

アップロード用に、クエリパラメータを含む署名付き URL を生成して、ファイルをアップロードします (PUT リクエスト)。

インポート。

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.awscore.AWSRequestOverrideConfiguration;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
```

```

import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;

```

URL を生成する。

```

/**
 * Creates a presigned URL to use in a subsequent HTTP PUT request. The code
 * adds query parameters
 * to the request instead of using headers. By using query parameters, you
 * do not need to add the
 * the parameters as headers when the PUT request is eventually sent.
 *
 * @param bucketName Bucket name where the object will be uploaded.
 * @param keyName Key name of the object that will be uploaded.
 * @param queryParams Query string parameters to be added to the presigned
 * URL.
 * @return
 */
public String createPresignedUrl(String bucketName, String keyName,
    Map<String, String> queryParams) {
    try (S3Presigner presigner = S3Presigner.create()) {
        // Create an override configuration to store the query parameters.

```

```

        AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder
= AwsRequestOverrideConfiguration.builder();

queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);

        PutObjectRequest objectRequest = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)

.overrideConfiguration(overrideConfigurationBuilder.build()) // Add the override
configuration.

                .build();

        PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL
expires in 10 minutes.
                .putObjectRequest(objectRequest)
                .build();

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

AWS for Java V2 SdkHttpClient クラスを使用してアップロードを行います。

```

/**
 * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT
request. Since the
 * URL contains the query parameters, no headers are needed for metadata, SSE
settings, or ACL settings.
 *
 * @param presignedUrlString The URL for the PUT request.

```

```
* @param fileToPut File to upload
*/
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut)
{
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());

        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

署名付き URL を作成して、オブジェクトをバケットにアップロードします。

```
import https from "node:https";

import { XMLParser } from "fast-xml-parser";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" }),
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Make a PUT request to the provided URL.
 *
 * @param {string} url
 * @param {string} data
 */
const put = (url, data) => {
  return new Promise((resolve, reject) => {
```



```

const req = https.request(
  url,
  { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
  (res) => {
    let responseBody = "";
    res.on("data", (chunk) => {
      responseBody += chunk;
    });
    res.on("end", () => {
      const parser = new XMLParser();
      if (res.statusCode >= 200 && res.statusCode <= 299) {
        resolve(parser.parse(responseBody, true));
      } else {
        reject(parser.parse(responseBody, true));
      }
    });
  },
);
req.on("error", (err) => {
  reject(err);
});
req.write(data);
req.end();
});
};

/**
 * Create two presigned urls for uploading an object to an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      key,
      region,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,

```

```

        key,
    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");

    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
} catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
        console.error(
            `There was an error getting your credentials. Are your local credentials
            configured?\n${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

署名付き URL を作成して、オブジェクトをバケットからダウンロードします。

```

import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
    getSignedUrl,
    S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
    const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
    const presigner = new S3RequestPresigner({
        credentials: fromIni(),
        region,
    });
    const request = new HttpRequest("GET", url.href);
    const signedUrl = await presigner.getSignedUrl(request);
    return signedUrl;
};

```

```
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(new HttpRequest(url));
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Create two presigned urls for downloading an object from an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      region,
      key,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    console.log("Presigned URL without client");
    console.log(noClientUrl);
    console.log("\n");

    console.log("Presigned URL with client");
    console.log(clientUrl);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
```

```
        `There was an error getting your credentials. Are your local credentials
        configured?\n${caught.name}: ${caught.message}` ,
    );
    } else {
        throw caught;
    }
}
};
```

- 詳細については、「[AWS SDK for JavaScript デベロッパーガイド](#)」を参照してください。

## Kotlin

### SDK for Kotlin

#### Note

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

GetObject の署名済みリクエストを作成し、その URL を使用してオブジェクトをダウンロードします。

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET
    request to retrieve the object.
```

```
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

詳細オプションを使用して GetObject 署名済みリクエストを作成します。

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be
            used 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

PutObject の署名済みリクエストを作成し、それを使用してオブジェクトをアップロードします。

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
```

```
val unsignedRequest =
    PutObjectRequest {
        bucket = bucketName
        key = keyName
    }

// Presign the request.
val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

// Use the URL and any headers from the presigned HttpRequest in a subsequent
// HTTP PUT request to retrieve the object.
// Create a PUT request using the OkHttpClient API.
val putRequest =
    Request
        .Builder()
        .url(presignedRequest.url.toString())
        .apply {
            presignedRequest.headers.forEach { key, values ->
                header(key, values.joinToString(", "))
            }
        }
        .put(content.toRequestBody())
        .build()

val response = OkHttpClient().newCall(putRequest).execute()
assert(response.isSuccessful)
}
```

- 詳細については、「[AWS SDK for Kotlin デベロッパーガイド](#)」を参照してください。

## PHP

### SDK for PHP

#### Note

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

```
namespace S3;
use Aws\Exception\AwsException;
```

```
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the
S3 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in
the given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}
```

```
$runner = new PresignedURL();
$runner->run();

namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }
}
```



```
public function getClient(): S3Client
{
    return $this->client;
}

public function setClient(S3Client $client)
{
    $this->client = $client;
}

public function emptyAndDeleteBucket($bucketName, array $args = [])
{
    try {
        $objects = $this->listAllObjects($bucketName, $args);
        $this->deleteObjects($bucketName, $objects, $args);
        if ($this->verbose) {
            echo "Deleted all objects and folders from $bucketName.\n";
        }
        $this->deleteBucket($bucketName, $args);
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "\nPlease fix error with bucket deletion before continuing.
\n";
        }
        throw $exception;
    }
}

public function createBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->createBucket($parameters);
        if ($this->verbose) {
            echo "Created the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
```

```

        echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
        echo "Please fix error with bucket creation before continuing.";
    }
    throw $exception;
}

}

public function putObject(string $bucketName, string $key, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $this->client->putObject($parameters);
        if ($this->verbose) {
            echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object uploading before continuing.";
        }
        throw $exception;
    }
}

public function getObject(string $bucketName, string $key, array $args = []):
Result
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $object = $this->client->getObject($parameters);
        if ($this->verbose) {
            echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    }
    } catch (AwsException $exception) {

```

```
        if ($this->verbose) {
            echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object downloading before
continuing.";
        }
        throw $exception;
    }
    return $object;
}

public function copyObject($bucketName, $key, $copySource, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
    try {
        $this->client->copyObject($parameters);
        if ($this->verbose) {
            echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object copying before continuing.";
        }
        throw $exception;
    }
}

public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
    try {
        $objects = $this->client->listObjectsV2($parameters);
        if ($this->verbose) {
            echo "Retrieved the list of objects from: $bucketName.\n";
        }
    }
}
```

```
    }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with list objects before continuing.";
        }
        throw $exception;
    }
    return $objects;
}

public function listAllObjects($bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);

    $contents = [];
    $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

    foreach ($paginator as $result) {
        if($result['KeyCount'] == 0){
            break;
        }
        foreach ($result['Contents'] as $object) {
            $contents[] = $object;
        }
    }
    return $contents;
}

public function deleteObjects(string $bucketName, array $objects, array $args
= [])
{
    $listOfObjects = array_map(
        function ($object) {
            return ['Key' => $object];
        },
        array_column($objects, 'Key')
    );
    if(!$listOfObjects){
```

```

        return;
    }

    $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
    try {
        $this->client->deleteObjects($parameters);
        if ($this->verbose) {
            echo "Deleted the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}

public function deleteBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->deleteBucket($parameters);
        if ($this->verbose) {
            echo "Deleted the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket deletion before continuing.";
        }
        throw $exception;
    }
}

public function deleteObject(string $bucketName, string $fileName, array
$args = [])

```

```

    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
        $args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }
}

public function listBuckets(array $args = [])
{
    try {
        $buckets = $this->client->listBuckets($args);
        if ($this->verbose) {
            echo "Retrieved all " . count($buckets) . "\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket lists before continuing.";
        }
        throw $exception;
    }
    return $buckets;
}

public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
{

```

```
        $request = $this->client->createPresignedRequest($command, $expires,
$options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }

    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
    }
}
```

## Python

### SDK for Python (Boto3)

#### Note

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

S3 アクションを期間限定で実行できる署名付き URL を生成します。Requests パッケージを使用して、URL でリクエストを行います。

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)

def generate_presigned_url(s3_client, client_method, method_parameters,
                           expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
            ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method
```



```
    )
    raise
    return url

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key",
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
        if response.status_code == 200:
            with open(args.key.split("/")[-1], 'wb') as object_file:
                object_file.write(response.content)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:
            with open(args.key, "rb") as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(
```

```

        f"Couldn't find {args.key}. For a PUT operation, the key must be
the "
        f"name of a file that exists on your computer."
    )

    if response is not None:
        print(f"Status: {response.status_code}\nReason: {response.reason}")

    print("-" * 88)

if __name__ == "__main__":
    usage_demo()

```

署名付き POST リクエストを生成して、ファイルをアップロードします。

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.
        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                    required access data.
        """
        try:

```

```
        response = self.bucket.meta.client.generate_presigned_post(
            Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
        )
        logger.info("Got presigned POST URL: %s", response["url"])
    except ClientError:
        logger.exception(
            "Couldn't get a presigned POST URL for bucket '%s' and object '%s'",
            self.bucket.name,
            object_key,
        )
        raise
    return response
```

## Ruby

### SDK for Ruby

#### Note

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

```
require 'aws-sdk-s3'
require 'net/http'

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
```

```
puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
why: #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request('PUT', presigned_url.request_uri, object_content,
'content_type' => '')
  end

  case response
  when Net::HTTPSuccess
    puts 'Content uploaded!'
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

## Rust

### SDK for Rust

#### Note

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

GET S3 オブジェクトへの事前署名リクエストを作成します。

```

/// Generate a URL for a presigned GET request.
async fn get_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;

    println!("Object URI: {}", presigned_request.uri());
    let valid_until = chrono::offset::Local::now() + expires_in;
    println!("Valid until: {valid_until}");

    Ok(())
}

```

PUT S3 オブジェクトへの事前署名リクエストを作成します。

```

async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<String, S3ExampleError> {
    let expires_in: std::time::Duration =
        std::time::Duration::from_secs(expires_in);
    let expires_in: aws_sdk_s3::presigning::PresigningConfig =
        PresigningConfig::expires_in(expires_in).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to convert expiration to PresigningConfig: {err:?}"
            ))
        })?;
    let presigned_request = client
        .put_object()
        .bucket(bucket)
        .key(object)

```

```

        .presigned(expires_in)
        .await?;

Ok(presigned_request.uri().into())
}

```

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

署名付きリクエストを作成してS3 オブジェクトを取得します。

```

" iv_bucket_name is the bucket name
" iv_key is the object name like "myfile.txt"

DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

"Upload a nice Hello World file to an S3 bucket."
TRY.
    DATA(lv_contents) = cl_abap_codepage=>convert_to( 'Hello, World' ).
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_contents
        iv_contenttype = 'text/plain' ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" now generate a presigned URL with a 600-second expiration
DATA(lo_presigner) = lo_s3->get_presigner( iv_expires_sec = 600 ).
" the presigner getobject() method has the same signature as
" lo_s3->getobject(), but it doesn't actually make the call.

```

```
" to the service. It just prepares a presigned URL for a future call
DATA(lo_presigned_req) = lo_presigner->getobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key ).

" You can provide this URL to a web page, user, email etc so they
" can retrieve the file. The URL will expire in 10 minutes.
ov_url = lo_presigned_req->get_url( ).
```

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## ユーザーがラベルを使用して写真を管理できる写真アセット管理アプリケーションの作成

以下のコード例は、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションの作成方法を示しています。

### .NET

#### SDK for .NET

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、[GitHub](#) で完全な例を参照してください。

この例のソースについて詳しくは、[AWS コミュニティ](#) でブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

## C++

### SDK for C++

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、[GitHub](#) で完全な例を参照してください。

この例のソースについて詳しくは、[AWS コミュニティ](#)でブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

## Java

### SDK for Java 2.x

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、[GitHub](#) で完全な例を参照してください。

この例のソースについて詳しくは、[AWS コミュニティ](#)でブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3



- Amazon SNS

## JavaScript

### SDK for JavaScript (v3)

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、[GitHub](#) で完全な例を参照してください。

この例のソースについて詳しくは、[AWS コミュニティ](#)でブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

## Kotlin

### SDK for Kotlin

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、[GitHub](#) で完全な例を参照してください。

この例のソースについて詳しくは、[AWS コミュニティ](#)でブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

## PHP

### SDK for PHP

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、[GitHub](#) で完全な例を参照してください。

この例のソースについて詳しくは、[AWS コミュニティ](#) でブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

## Rust

### SDK for Rust

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、[GitHub](#) で完全な例を参照してください。

この例のソースについて詳しくは、[AWS コミュニティ](#) でブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK を使用した Amazon S3 での開発](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 オブジェクトを一覧表示するウェブページ

次のコード例は、ウェブページに Amazon S3 オブジェクトを一覧表示する方法を示しています。

### JavaScript

#### SDK for JavaScript (v3)

#### Note

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

以下のコードは、AWS SDK を呼び出す、関連する React コンポーネントです。このコンポーネントを含むアプリケーションの実行可能なバージョンは、前述の GitHub リンクにあります。

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  type ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
```

```
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

useEffect(() => {
  const client = new S3Client({
    region: "us-east-1",
    // Unless you have a public bucket, you'll need access to a private bucket.
    // One way to do this is to create an Amazon Cognito identity pool, attach
    // a role to the pool,
    // and grant the role access to the 's3:GetObject' action.
    //
    // You'll also need to configure the CORS settings on the bucket to allow
    // traffic from
    // this example site. Here's an example configuration that allows all
    // origins. Don't
    // do this in production.
    // [
    //   {
    //     "AllowedHeaders": ["*"],
    //     "AllowedMethods": ["GET"],
    //     "AllowedOrigins": ["*"],
    //     "ExposeHeaders": [],
    //   },
    // ],
    //
    credentials: fromCognitoIdentityPool({
      clientConfig: { region: "us-east-1" },
      identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
    }),
  });
  const command = new ListObjectsCommand({ Bucket: "bucket-name" });
  client.send(command).then(({ Contents }) => setObjects(Contents || []));
}, []);

return (
  <div className="App">
    {objects.map((o) => (
      <div key={o.ETag}>{o.Key}</div>
    ))}
  </div>
);
}

export default App;
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## Amazon Textract エクスプローラーアプリケーションを作成する

次のコード例は、インタラクティブアプリケーションで Amazon Textract の出力を調べる方法を示しています。

### JavaScript

#### SDK for JavaScript (v3)

AWS SDK for JavaScript を使用し、Amazon Textract を使用してドキュメント画像からデータを抽出し、インタラクティブなウェブページに表示する React アプリケーションを構築する方法を示します。この例はウェブブラウザで実行され、認証情報に認証された Amazon Cognito ID が必要です。Amazon Simple Storage Service (Amazon S3) をストレージに使用し、通知のために、Amazon Simple Notification Service (Amazon SNS) トピックにサブスクライブした Amazon Simple Queue Service (Amazon SQS) キューをポーリングします。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

#### この例で使用されているサービス

- Amazon Cognito ID
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

## Python

### SDK for Python (Boto3)

Amazon Textract で AWS SDK for Python (Boto3) を使用して、ドキュメントイメージ内のテキスト、フォーム、および表要素を検出する方法を示します。入力イメージと Amazon Textract 出力は、検出された要素を探索できる Tkinter アプリケーションに表示されます。

- Amazon Textract にドキュメントイメージを送信し、検出された要素の出力を調べます。
- Amazon Textract に直接イメージを送信するか、Amazon Simple Storage Service (Amazon S3) バケットを通じてイメージを送信します。
- 非同期 API を使用して、ジョブの完了時に Amazon Simple Notification Service (Amazon SNS) トピックに通知を発行するジョブを開始します。
- Amazon Simple Queue Service (Amazon SQS) キューにジョブ完了メッセージについてポーリングし、結果を表示します。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Amazon Cognito ID
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK を使用した Amazon S3 での開発](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して、指定した Amazon S3 バケットからすべてのオブジェクトを削除します。

次のコード例は、Amazon S3 バケットからすべてのオブジェクトを削除する方法を示しています。

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

Amazon S3 バケットからすべてのオブジェクトを削除します。

```
import {
  DeleteObjectsCommand,
  paginateListObjectsV2,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 *
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  try {
    console.log(`Deleting all objects in bucket: ${bucketName}`);

    const paginator = paginateListObjectsV2(
      { client },
      {
        Bucket: bucketName,
      },
    );

    const objectKeys = [];
    for await (const { Contents } of paginator) {
      objectKeys.push(...Contents.map((obj) => ({ Key: obj.Key })));
    }

    const deleteCommand = new DeleteObjectsCommand({
      Bucket: bucketName,
      Delete: { Objects: objectKeys },
    });
```

```
});

await client.send(deleteCommand);

console.log(`All objects deleted from bucket: ${bucketName}`);
} catch (caught) {
  if (caught instanceof Error) {
    console.error(
      `Failed to empty ${bucketName}. ${caught.name}: ${caught.message}`,
    );
  }
}
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    bucketName: {
      type: "string",
    },
  };

  const { values } = parseArgs({ options });
  main(values);
}
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の以下のトピックを参照してください。
  - [DeleteObjects](#)
  - [ListObjectsV2](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。



## AWS SDK を使用して Amazon S3 への不完全なマルチパートアップロードを削除する

次のコード例は、不完全な Amazon S3 マルチパートアップロードを削除または停止する方法を示しています。

Java

SDK for Java 2.x

### Note

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

何らかの理由で進行中または不完全なマルチパートアップロードを停止するには、次の例に示すように、リストアップロードを取得してから削除します。

```
/**
 * Aborts all incomplete multipart uploads from the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket
 */
public static void abortIncompleteMultipartUploadsFromList(String bucketName)
{
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(upload.key())
            .expectedBucketOwner(accountId)
            .uploadId(upload.uploadId())
```

```

        .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}

```

日付の前後に開始された不完全なマルチパートアップロードを削除するには、次の例に示すように、特定の時点に基づいてマルチパートアップロードを選択的に削除します。

```

/**
 * Aborts incomplete multipart uploads older than the specified point in
time.
 *
 * @param bucketName the name of the S3 bucket
 * @param pointInTime the cutoff time; uploads initiated before this are
aborted
 */
static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
        if (upload.initiated().isBefore(pointInTime)) {
            abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())

```

```

        .expectedBucketOwner(accountId)
        .uploadId(upload.uploadId())
        .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}
}

```

マルチパートアップロードの開始後にアップロード ID にアクセスできる場合は、その ID を使用して進行中のアップロードを削除できます。

```

/**
 * Aborts a multipart upload using the upload ID.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 */
static void abortMultipartUploadUsingUploadId(String bucketName, String key)
{
    String uploadId = startUploadReturningUploadId(bucketName, key);
    AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
    }
}

```

特定の日数だけ古い不完全なマルチパートアップロードを一貫して削除するには、バケットのバケットライフサイクル設定を設定します。次の例は、7 日以上経過した不完全なアップロードを削除するルールを作成する方法を示しています。

```
/**
 * Configures a lifecycle rule to abort incomplete multipart uploads after 7
 days.
 *
 * @param bucketName the name of the S3 bucket
 */
static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
    Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());

    PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
    } else {
        logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
    }
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [AbortMultipartUpload](#)
  - [ListMultipartUploads](#)
  - [PutBucketLifecycleConfiguration](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して、Amazon Rekognition で画像内の PPE を検出する

次のコード例は、Amazon Rekognition を使用してイメージ内の個人用防護具 (PPE) を検出するアプリケーションを構築する方法を示しています。

### Java

#### SDK for Java 2.x

AWS Lambda 個人用保護具のイメージを検出する関数の作成方法を示します。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して、画像から抽出されたテキスト内のエンティティを検出する

次のコード例は、Amazon Comprehend を使用して、Amazon S3 に格納されている画像から Amazon Textract によって抽出されたテキスト内のエンティティを検出する方法を示しています。

### Python

#### SDK for Python (Boto3)

Jupyter Notebook で AWS SDK for Python (Boto3) を使用して、画像から抽出されたテキスト内のエンティティを検出する方法を示しています。この例では、Amazon Textract を使用

して Amazon Simple Storage Service (Amazon S3) に保存されている画像からテキストを抽出し、Amazon Comprehend を使用して、抽出されたテキスト内のエンティティを検出します。

この例は Jupyter Notebook であり、ノートブックをホストできる環境で実行する必要があります。Amazon SageMaker AI を使用してサンプルを実行する方法については、「[TextractAndComprehendNotebook.ipynb](#)」の手順を参照してください。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Amazon Comprehend
- Amazon S3
- Amazon Textract

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK を使用した Amazon S3 での開発](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して、画像内の顔を検出する

次のコード例は、以下の操作方法を示しています。

- イメージを Amazon S3 バケットに保存します。
- Amazon Rekognition を使用して、年齢層、性別、感情 (笑顔など) などの顔の詳細を検出します。
- これらの詳細を表示します。

### Rust

#### SDK for Rust

アップロードプレフィックスを付け、Amazon S3 バケット内でイメージを保存し、Amazon Rekognition を使用して、年齢層、性別、感情 (笑顔など) などの顔の詳細を検出し、それらの詳細を表示します。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して、Amazon Rekognition で画像内のオブジェクトを検出する

次のコード例は、Amazon Rekognition を使用して画像内からカテゴリ別にオブジェクトを検出するアプリケーションを構築する方法を示しています。

### .NET

#### SDK for .NET

Amazon Simple Storage Service (Amazon S3) バケットにあるイメージ内から、Amazon Rekognition を使用してカテゴリ別にオブジェクトを識別するアプリケーションを、Amazon Rekognition .NET API を使用して作成する方法を示します。アプリケーションは、Amazon Simple Email Service (Amazon SES) を使用して、結果を記載した E メール通知を管理者に送信します。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

### Java

#### SDK for Java 2.x

Amazon Rekognition Java API を使用して、Amazon Rekognition を使用し、 Amazon Simple Storage Service (Amazon S3) バケットにあるイメージ内でカテゴリ別にオブジェク

トを識別するアプリケーションを作成する方法について説明します。アプリケーションは Amazon Simple Email Service (Amazon SES) を使用して、結果を含む E メール通知を管理者に送信します。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

## JavaScript

### SDK for JavaScript (v3)

AWS SDK for JavaScript で Amazon Rekognition を使用し、Amazon Rekognition を使用して Amazon Simple Storage Service (Amazon S3) バケット 内にある画像内でカテゴリ別にオブジェクトを識別するアプリケーションを作成します。アプリケーションは Amazon Simple Email Service (Amazon SES) を使用して、結果を含む E メール通知を管理者に送信します。

以下ではその方法を説明しています。

- Amazon Cognito を使用して認証されていないユーザーを作成します。
- Amazon Rekognition を使用して、オブジェクトのイメージを分析します。
- Amazon SES の E メールアドレスを検証します。
- Amazon SES を使用して、E メール通知を送信します。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES



## Kotlin

### SDK for Kotlin

Amazon Simple Storage Service (Amazon S3) バケットにある画像内からカテゴリ別にオブジェクトを Amazon Rekognition を使用して識別するアプリケーションを、Amazon Rekognition Kotlin API を使用して作成する方法を示します。アプリケーションは、Amazon Simple Email Service (Amazon SES) を使用して、結果を記載した E メール通知を管理者に送信します。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Python

### SDK for Python (Boto3)

AWS SDK for Python (Boto3) を使用して、以下のことができるウェブアプリケーションを作成する方法を示します。

- Amazon Simple Storage Service (Amazon S3) バケットに、写真をアップロードします。
- Amazon Rekognition を使用して、写真を分析およびラベル付けします。
- Amazon Simple Email Service (Amazon SES) を使用して、イメージ分析の E メールレポートを送信します。

この例には、React で構築された JavaScript で記述されたウェブページと、Flask-RESTful で構築された Python で記述された REST サービスの 2 つの主要なコンポーネントが含まれています。

React ウェブページを使用すると、次のことができます。

- S3 バケットに保存されているイメージのリストを表示します。
- イメージを S3 バケットにアップロードします。

- イメージ内で検出された項目を識別するイメージとラベルを表示します。
- S3 バケット内のすべてのイメージのレポートを取得し、レポートの E メールを送信します。

ウェブページが REST サービスを呼び出します。サービスはリクエストを AWS に送信して、以下のアクションを実行します。

- S3 バケット内のイメージのリストを取得し、フィルタリングします。
- Amazon S3 バケットに写真をアップロードします。
- Amazon Rekognition を使用して個々の写真を分析し、写真で検出された項目を識別するラベルのリストを取得します。
- S3 バケット内のすべての写真を分析し、Amazon SES を使用してレポートを E メールで送信します。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して、Amazon Rekognition で、動画内の人やオブジェクトを検出する

次のコード例は、Amazon Rekognition で動画内の人やオブジェクトを検出する方法を示します。

Java

SDK for Java 2.x

Amazon Rekognition Java API を使用し、Amazon Simple Storage Service (Amazon S3) バケットにあるビデオ内で顔やオブジェクトを検出するアプリケーションを作成する方法につ

いて説明します。アプリケーションは Amazon Simple Email Service (Amazon SES) を使用して、結果を含む E メール通知を管理者に送信します。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

## Python

### SDK for Python (Boto3)

Amazon Rekognition を使用して、非同期検出ジョブを開始して、動画内の顔、オブジェクト、人を検出します。この例では、ジョブが完了し、Amazon Simple Queue Service (Amazon SQS) キューをトピックにサブスクライブしたときに、Amazon Simple Notification Service (Amazon SNS) トピックに通知するように Amazon Rekognition を設定します。キューがジョブに関するメッセージを受信すると、ジョブが取得され、結果が出力されます。

この例は [GitHub](#) で最もよく見られます。完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、「[AWS SDK を使用した Amazon S3 での開発](#)」を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## Amazon Simple Storage Service (Amazon S3) バケットから S3 の「ディレクトリ」をダウンロードする

次のコード例は、Amazon S3 バケットの「ディレクトリ」の内容をダウンロードしてフィルタリングする方法を示しています。

Java

SDK for Java 2.x

### Note

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

この例では、AWS SDK for Java 2.x で [S3TransferManager](#) を使用して Amazon S3 バケットから「ディレクトリ」をダウンロードする方法を示しています。また、リクエストで [DownloadFilters](#) を使用する方法も示します。

```
/**
 * For standard buckets, S3 provides the illusion of a directory structure
 * through the use of keys. When you upload
 * an object to an S3 bucket, you specify a key, which is essentially the
 * "path" to the object. The key can contain
 * forward slashes ("/") to make it appear as if the object is stored in a
 * directory structure, but this is just a
 * logical representation, not an actual directory.
 * <p><pre>
 * In this example, our S3 bucket contains the following objects:
 *
 * folder1/file1.txt
 * folder1/file2.txt
 * folder1/file3.txt
 * folder2/file1.txt
 * folder2/file2.txt
 * folder2/file3.txt
 * folder3/file1.txt
 * folder3/file2.txt
 * folder3/file3.txt
 *
 * When method `downloadS3Directories` is invoked with
```

```

* `destinationPathURI` set to `/test`, the downloaded
* directory looks like:
*
* |- test
*   |- folder1
*     |- file1.txt
*     |- file2.txt
*     |- file3.txt
*   |- folder3
*     |- file1.txt
*     |- file2.txt
*     |- file3.txt
* </pre>
*
* @param transferManager    An S3TransferManager instance.
* @param destinationPathURI local directory to hold the downloaded S3
'directories' and files.
* @param bucketName        The S3 bucket that contains the 'directories' to
download.
* @return The number of objects (files, in this case) that were downloaded.
*/
public Integer downloadS3Directories(S3TransferManager transferManager,
                                     URI destinationPathURI, String
bucketName) {

    // Define the filters for which 'directories' we want to download.
    DownloadFilter folder1Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder1/");
    DownloadFilter folder3Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder3/");
    DownloadFilter folderFilter = s3Object ->
folder1Filter.or(folder3Filter).test(s3Object);

    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                                .destination(Paths.get(destinationPathURI))
                                .bucket(bucketName)
                                .filter(folderFilter)
                                .build());

    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    Integer numFilesInFolder1 =
Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;

```

```
Integer numFilesInFolder3 =
Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

try {
    assert numFilesInFolder1 == 3;
    assert numFilesInFolder3 == 3;
    assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2`
was not downloaded.
} catch (AssertionError e) {
    logger.error("An assertion failed.");
}

completedDirectoryDownload.failedTransfers()
    .forEach(fail -> logger.warn("Object failed to transfer [{ }]",
fail.exception().getMessage()));
return numFilesInFolder1 + numFilesInFolder3;
}
```

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## Amazon Simple Storage Service (Amazon S3) バケットからすべてのオブジェクトを、ローカルディレクトリにダウンロードする

次のコード例は、Amazon Simple Storage Service (Amazon S3) バケットからすべてのオブジェクトをローカルディレクトリにダウンロードする方法を示しています。

Java

SDK for Java 2.x

### Note

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

[S3TransferManager](#) を使用して、同じ S3 バケットにある [すべての S3 オブジェクトをダウンロード](#) します。 [完全なファイル](#) と [テスト](#) を表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
        URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
            .destination(Paths.get(destinationPathURI))
            .bucket(bucketName)
            .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
            .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## サイズが不明なストリームを、AWS SDK を使用して Amazon S3 オブジェクトからダウンロードする

次のコード例は、サイズが不明なストリームを Amazon S3 オブジェクトからダウンロードする方法を示しています。

Swift

SDK for Swift

### Note

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

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams

/// Download a file from the specified bucket.
///
/// - Parameters:
///   - bucket: The Amazon S3 bucket name to get the file from.
///   - key: The name (or path) of the file to download from the bucket.
///   - destPath: The pathname on the local filesystem at which to store
///     the downloaded file.
func downloadFile(bucket: String, key: String, destPath: String?) async
throws {
    let fileURL: URL

    // If no destination path was provided, use the key as the name to use
    // for the file in the downloads folder.
```



```
if destPath == nil {
    do {
        try fileURL = FileManager.default.url(
            for: .downloadsDirectory,
            in: .userDomainMask,
            appropriateFor: URL(string: key),
            create: true
        ).appendingPathComponent(key)
    } catch {
        throw TransferError.directoryError
    }
} else {
    fileURL = URL(fileURLWithPath: destPath!)
}

let config = try await S3Client.S3ClientConfiguration(region: region)
let s3Client = S3Client(config: config)

// Create a `FileHandle` referencing the local destination. Then
// create a `ByteStream` from that.

FileManager.default.createFile(atPath: fileURL.path, contents: nil,
attributes: nil)
let fileHandle = try FileHandle(forWritingTo: fileURL)

// Download the file using `GetObject`.

let getInput = GetObjectInput(
    bucket: bucket,
    key: key
)

do {
    let getOutput = try await s3Client.getObject(input: getInput)

    guard let body = getOutput.body else {
        throw TransferError.downloadError("Error: No data returned for
download")
    }

    // If the body is returned as a `Data` object, write that to the
    // file. If it's a stream, read the stream chunk by chunk,
    // appending each chunk to the destination file.
```

```
switch body {
case .data:
    guard let data = try await body.readData() else {
        throw TransferError.downloadError("Download error")
    }

    // Write the `Data` to the file.

    do {
        try data.write(to: fileURL)
    } catch {
        throw TransferError.writeError
    }
    break

case .stream(let stream as ReadableStream):
    while (true) {
        let chunk = try await stream.readAsync(upToCount: 5 * 1024 *
1024)

        guard let chunk = chunk else {
            break
        }

        // Write the chunk to the destination file.

        do {
            try fileHandle.write(contentsOf: chunk)
        } catch {
            throw TransferError.writeError
        }
    }

    break
default:
    throw TransferError.downloadError("Received data is unknown
object type")
}
} catch {
    throw TransferError.downloadError("Error downloading the file:
\\(error)")
}

print("File downloaded to \\(fileURL.path).")
```

```
}
```

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用してマルチリージョンアクセスポイントから Amazon S3 オブジェクトを取得する

次のコード例は、マルチリージョンアクセスポイントからオブジェクトを取得する方法を示しています。

Kotlin

SDK for Kotlin

### Note

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

非対称 Sigv4 (Sigv4a) 署名アルゴリズムを使用するように S3 クライアントを設定します。

```
suspend fun createS3Client(): S3Client {  
    // Configure your S3Client to use the Asymmetric SigV4 (SigV4a)  
    signing algorithm.  
    val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)  
    val s3 = S3Client.fromEnvironment {  
        authSchemes = listOf(sigV4aScheme)  
    }  
    return s3  
}
```

バケット名の代わりにマルチリージョンアクセスポイント ARN を使用してオブジェクトを取得します。

```
suspend fun getObjectFromMrap(
```

```
s3: S3Client,
mrapArn: String,
keyName: String,
): String? {
    val request = GetObjectRequest {
        bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
        key = keyName
    }

    var stringObj: String? = null
    s3.getObject(request) { resp ->
        stringObj = resp.body?.decodeToString()
        if (stringObj != null) {
            println("Successfully read $keyName from $mrapArn")
        }
    }
    return stringObj
}
```

- 詳細については、「[AWS SDK for Kotlin デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[GetObject](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して、Amazon S3 バケットから If-Modified-Since を指定してオブジェクトを取得する

次のコード例は、S3 バケット内のオブジェクトからデータを読み取る方法を示しています。ただし、そのバケットが前回の取得時刻以降に変更されていない場合に限りです。

## Rust

### SDK for Rust

#### Note

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

```
use aws_sdk_s3::{
    error::SdkError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client,
};
use s3_code_examples::error::S3ExampleError;
use tracing::{error, warn};

const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
```

```
let bucket_name = format!("if-modified-since-{{uuid}}");
client
    .create_bucket()
    .bucket(bucket_name.clone())
    .send()
    .await?;

// Create a new object in the bucket whose name is `KEY` and whose
// contents are `BODY`.
let put_object_output = client
    .put_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .body(ByteStream::from_static(BODY.as_bytes()))
    .send()
    .await;

// If the `PutObject` succeeded, get the eTag string from it. Otherwise,
// report an error and return an empty string.
let e_tag_1 = match put_object_output {
    Ok(put_object) => put_object.e_tag.unwrap(),
    Err(err) => {
        error!("{err:?}");
        String::new()
    }
};

// Request the object's headers.
let head_object_output = client
    .head_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .send()
    .await;

// If the `HeadObject` request succeeded, create a tuple containing the
// values of the headers `last-modified` and `etag`. If the request
// failed, return the error in a tuple instead.
let (last_modified, e_tag_2) = match head_object_output {
    Ok(head_object) => (
        Ok(head_object.last_modified().cloned().unwrap()),
        head_object.e_tag.unwrap(),
    ),
    Err(err) => (Err(err), String::new()),
};
```

```
};

warn!("last modified: {last_modified:?}");
assert_eq!(
    e_tag_1, e_tag_2,
    "PutObject and first GetObject had differing eTags"
);

println!("First value of last_modified: {last_modified:?}");
println!("First tag: {}\n", e_tag_1);

// Send a second `HeadObject` request. This time, the `if_modified_since`
// option is specified, giving the `last_modified` value returned by the
// first call to `HeadObject`.
//
// Since the object hasn't been changed, and there are no other objects in
// the bucket, there should be no matching objects.

let head_object_output = client
    .head_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .if_modified_since(last_modified.unwrap())
    .send()
    .await;

// If the `HeadObject` request succeeded, the result is a tuple containing
// the `last_modified` and `e_tag_1` properties. This is _not_ the expected
// result.
//
// The _expected_ result of the second call to `HeadObject` is an
// `SdkError::ServiceError` containing the HTTP error response. If that's
// the case and the HTTP status is 304 (not modified), the output is a
// tuple containing the values of the HTTP `last-modified` and `etag`
// headers.
//
// If any other HTTP error occurred, the error is returned as an
// `SdkError::ServiceError`.

let (last_modified, e_tag_2) = match head_object_output {
    Ok(head_object) => (
        Ok(head_object.last_modified().cloned().unwrap()),
        head_object.e_tag.unwrap(),
    ),
```

```

Err(err) => match err {
  SdkError::ServiceError(err) => {
    // Get the raw HTTP response. If its status is 304, the
    // object has not changed. This is the expected code path.
    let http = err.raw();
    match http.status().as_u16() {
      // If the HTTP status is 304: Not Modified, return a
      // tuple containing the values of the HTTP
      // `last-modified` and `etag` headers.
      304 => (
        Ok(DateTime::from_str(
          http.headers().get("last-modified").unwrap(),
          DateTimeFormat::HttpDate,
        )
          .unwrap()),
        http.headers().get("etag").map(|t| t.into()).unwrap(),
      ),
      // Any other HTTP status code is returned as an
      // `SdkError::ServiceError`.
      _ => (Err(SdkError::ServiceError(err)), String::new()),
    }
  }
  // Any other kind of error is returned in a tuple containing the
  // error and an empty string.
  _ => (Err(err), String::new()),
},
};

warn!("last modified: {last_modified:?}");
assert_eq!(
  e_tag_1, e_tag_2,
  "PutObject and second HeadObject had different eTags"
);

println!("Second value of last modified: {last_modified:?}");
println!("Second tag: {}", e_tag_2);

// Clean up by deleting the object and the bucket.
client
  .delete_object()
  .bucket(bucket_name.as_str())
  .key(KEY)
  .send()
  .await?;

```



```
client
    .delete_bucket()
    .bucket(bucket_name.as_str())
    .send()
    .await?;

Ok(())
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 オブジェクトの暗号化の使用を開始する

次のコード例は、Amazon S3 オブジェクトの暗号化を開始する方法を示しています。

.NET

SDK for .NET

### Note

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

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
```

```
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class SSEClientEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "exampleobject.txt";
        string copyTargetKeyName = "examplecopy.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();

        try
        {
            // Create an encryption key.
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);

            // Upload the object.
            PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

            // Download the object and verify that its contents match what
you uploaded.
            await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

            // Get object metadata and verify that the object uses AES-256
encryption.
            await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

            // Copy both the source and target objects using server-side
encryption with
            // an encryption key.
            await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
        }
    }
}
```

```

        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }

    /// <summary>
    /// Uploads an object to an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// PutObjectAsync.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
    /// object will be uploaded.</param>
    /// <param name="keyName">The name of the object to upload to the Amazon
S3
    /// bucket.</param>
    /// <param name="base64Key">The encryption key.</param>
    /// <returns>The PutObjectRequest object for use by
DownloadObjectAsync.</returns>
    public static async Task<PutObjectRequest> UploadObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string base64Key)
    {
        PutObjectRequest putObjectRequest = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
            ContentBody = "sample text",
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };
        PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
        return putObjectRequest;
    }

    /// <summary>
    /// Downloads an encrypted object from an Amazon S3 bucket.
    /// </summary>

```

```
call    /// <param name="client">The initialized Amazon S3 client object used to
object  /// GetObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// is located.</param>
        /// <param name="keyName">The name of the Amazon S3 object to download.</
param>   param>
        /// <param name="base64Key">The encryption key used to encrypt the
        /// object.</param>
        /// <param name="putObjectRequest">The PutObjectRequest used to upload
        /// the object.</param>
        public static async Task DownloadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key,
            PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // Provide encryption information for the object stored in Amazon
S3.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
                using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
                {
                    string content = reader.ReadToEnd();
                    if (string.Compare(putObjectRequest.ContentBody, content) == 0)
                    {
                        Console.WriteLine("Object content is same as we uploaded");
                    }
                    else
                    {
                        Console.WriteLine("Error...Object content is not same.");
                    }
                }
            }
        }
    }
}
```

```

        }

        if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
        {
            Console.WriteLine("Object encryption method is AES256, same
as we set");
        }
        else
        {
            Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
        }
    }
}

/// <summary>
/// Retrieves the metadata associated with an Amazon S3 object.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used
/// to call GetObjectMetadataAsync.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket containing
the
/// object for which we want to retrieve metadata.</param>
/// <param name="keyName">The name of the object for which we wish to
/// retrieve the metadata.</param>
/// <param name="base64Key">The encryption key associated with the
/// object.</param>
public static async Task GetObjectMetadataAsync(
    IAmazonS3 client,
    string bucketName,
    string keyName,
    string base64Key)
{
    GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
    {
        BucketName = bucketName,
        Key = keyName,

        // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
    }
}

```

```

        ServerSideEncryptionCustomerProvidedKey = base64Key,
    };

    GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
    Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
}

/// <summary>
/// Copies an encrypted object from one Amazon S3 bucket to another.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// CopyObjectAsync.</param>
/// <param name="bucketName">The Amazon S3 bucket containing the object
/// to copy.</param>
/// <param name="keyName">The name of the object to copy.</param>
/// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
/// will be copied.</param>
/// <param name="aesEncryption">The encryption type to use.</param>
/// <param name="base64Key">The encryption key to use.</param>
public static async Task CopyObjectAsync(
    IAmazonS3 client,
    string bucketName,
    string keyName,
    string copyTargetKeyName,
    Aes aesEncryption,
    string base64Key)
{
    aesEncryption.GenerateKey();
    string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

    CopyObjectRequest copyRequest = new CopyObjectRequest
    {
        SourceBucket = bucketName,
        SourceKey = keyName,
        DestinationBucket = bucketName,
        DestinationKey = copyTargetKeyName,

        // Information about the source object's encryption.
        CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
    }

```

```
CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

// Information about the target object's encryption.
ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
};
await client.CopyObjectAsync(copyRequest);
}
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
- [CopyObject](#)
- [GetObject](#)
- [GetObjectMetadata](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 オブジェクトのタグの使用を開始する

次のコード例は、Amazon S3 オブジェクトのタグの使用を開始する方法を示しています。

.NET

SDK for .NET

### Note

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

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
```

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
/// Service (Amazon S3) objects.
/// </summary>
public class ObjectTag
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "newobject.txt";
        string filePath = @"*** file path ***";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        var client = new AmazonS3Client(bucketRegion);
        await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
    }

    /// <summary>
    /// This method uploads an object with tags. It then shows the tag
    /// values, changes the tags, and shows the new tags.
    /// </summary>
    /// <param name="client">The Initialized Amazon S3 client object used
    /// to call the methods to create and change an objects tags.</param>
    /// <param name="bucketName">A string representing the name of the
    /// bucket where the object will be stored.</param>
    /// <param name="keyName">A string representing the key name of the
    /// object to be tagged.</param>
    /// <param name="filePath">The directory location and file name of the
    /// object to be uploaded to the Amazon S3 bucket.</param>
    public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
    {
        try
        {
            // Create an object with tags.
            var putRequest = new PutObjectRequest
            {
                BucketName = bucketName,
```



```
        Key = keyName,
        FilePath = filePath,
        TagSet = new List<Tag>
        {
            new Tag { Key = "Keyx1", Value = "Value1" },
            new Tag { Key = "Keyx2", Value = "Value2" },
        },
    };

    PutObjectResponse response = await
client.PutObjectAsync(putRequest);

    // Now retrieve the new object's tags.
    GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
    {
        BucketName = bucketName,
        Key = keyName,
    };

    GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

    // Display the tag values.
    objectTags.Tagging
        .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

    Tagging newTagSet = new Tagging()
    {
        TagSet = new List<Tag>
        {
            new Tag { Key = "Key3", Value = "Value3" },
            new Tag { Key = "Key4", Value = "Value4" },
        },
    };

    PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
    {
        BucketName = bucketName,
        Key = keyName,
        Tagging = newTagSet,
    };
};
```

```
        PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

        // Retrieve the tags again and show the values.
        GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
        {
            BucketName = bucketName,
            Key = keyName,
        };
        GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

        objectTags2.Tagging
            .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine(
            $"Error: '{ex.Message}'");
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## ビッグデータ処理クラスターの使用開始

次のコード例は、以下の操作方法を示しています。

- EC2 キーペアを作成する
- ストレージをセットアップしてアプリケーションを準備する

- リソースをクリーンアップする

## Bash

AWS CLIBash スクリプトを使用した

### Note

GitHub には、その他のリソースもあります。[デベロッパーチュートリアル](#)のサンプルレポジトリで用例一覧を検索し、設定と実行の方法を確認してください。

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}
```

```
# Function to clean up resources
cleanup() {
    echo ""
    echo "=====
    echo "CLEANUP IN PROGRESS"
    echo "=====
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null ||
true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ];
then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi
```

```
# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
"Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-
gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \
```

```

"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
|| true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to
    find the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as
    's3://emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-
    tutorial-bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
    Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

```

```
# Create a DataFrame of the top 10 restaurants with the most Red
violations
top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
total_red_violations
FROM restaurant_violations
WHERE violation_type = 'RED'
GROUP BY name
ORDER BY total_red_violations DESC LIMIT 10""")

# Write the results to the specified output URI
top_red_violation_restaurants.write.option("header",
"true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
"Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
then
```

```
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample
data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
    handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output
text 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi
```



```
# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
  --release-label emr-6.10.0 \
  --applications Name=Spark \
  --ec2-attributes KeyName="$KEY_NAME" \
  --instance-type m5.xlarge \
  --instance-count 3 \
  --use-default-roles \
  --log-uri "s3://$BUCKET_NAME/logs/" \
  --ebs-root-volume-size 100 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
  --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
  handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
  CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
  CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:' \
  [[:space:]]*"^[^"]*" | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then
  handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
  "Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
```

```

echo "Waiting for cluster to reach WAITING state..."
WAIT_COUNT=0
MAX_WAIT=120
while [ "$CLUSTER_STATE" != "WAITING" ]; do
    if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
        handle_error "Cluster did not reach WAITING state within timeout
period"
    fi
    sleep 30
    CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --
query "Cluster.Status.State" --output text)
    echo "Current cluster state: $CLUSTER_STATE"

    # Check for error states
    if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
"TERMINATED" ]]; then
        handle_error "Cluster entered error state: $CLUSTER_STATE"
    fi
    WAIT_COUNT=$((WAIT_COUNT + 1))
done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
    --cluster-id "$CLUSTER_ID" \
    --steps Type=Spark,Name="Health Violations
Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq >> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

```

```
if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
    handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id
"$STEP_ID" --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output
files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv"
| head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv"
| head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi
fi
```

```
aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
  handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./}
${KEY_NAME}.pem)"

# Display summary of created resources
echo ""
echo "=====
echo "RESOURCES CREATED"
echo "=====
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup

echo "Script completed successfully."
```

- API の詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。
  - [AddSteps](#)
  - [Cp](#)
  - [CreateCluste:](#)
  - [CreateDefaultRoles](#)
  - [:CreateKeyPair](#)
  - [DescribeCluster](#)
  - [DescribeKeyPairs](#)

- [DescribeStep](#)
- [Ls](#)
- [Mb](#)
- [Rb](#)
- [Rm](#)
- [Ssh](#)
- [TerminateClusters](#)
- [待ちます。](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## 設定管理の使用開始

次のコード例は、以下の操作方法を示しています。

- Amazon S3 バケットを作成する
- Amazon SNS トピックを作成する
- Config 用の IAM ロールを作成する
- Config 設定レコーダーを設定する
- Config 配信チャネルを設定する
- 設定レコーダーを起動する
- Config の設定を確認する

### Bash

AWS CLIBash スクリプトを使用した

#### Note

GitHub には、その他のリソースもあります。[デベロッパーチュートリアル](#)のサンプルレポジトリで用例一覧を検索し、設定と実行の方法を確認してください。

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi
}
```

```
fi

# Check if we created a new delivery channel before trying to delete it
if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
"true" ]; then
    echo "Deleting delivery channel..."
    aws configservice delete-delivery-channel --delivery-channel-name
"$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
fi

if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
"true" ]; then
    echo "Deleting configuration recorder..."
    aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
fi

if [ -n "$ROLE_NAME" ]; then
    if [ -n "$POLICY_NAME" ]; then
        echo "Detaching custom policy from role..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" 2>/dev/null || true
    fi

    if [ -n "$MANAGED_POLICY_ARN" ]; then
        echo "Detaching managed policy from role..."
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN" 2>/dev/null || true
    fi

    echo "Deleting IAM role..."
    aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
fi

if [ -n "$SNS_TOPIC_ARN" ]; then
    echo "Deleting SNS topic..."
    aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
fi

if [ -n "$S3_BUCKET_NAME" ]; then
    echo "Emptying S3 bucket..."
    aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

    echo "Deleting S3 bucket..."
```

```

        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "=====
    echo "CREATED RESOURCES"
    echo "=====
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "=====
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket

```



```
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1" # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets"
```

```
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-`${RANDOM_ID}`"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
  Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-`${RANDOM_ID}`"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
  Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
```

```
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
```

```
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-records 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
    "name": "${CONFIG_RECORDER_NAME}",
    "roleARN": "${ROLE_ARN}",
    "recordingMode": {
        "recordingFrequency": "CONTINUOUS"
    }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
    "allSupported": true,
    "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
```

```

RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
    "name": "${DELIVERY_CHANNEL_NAME}",
    "s3BucketName": "${S3_BUCKET_NAME}",
    "snsTopicARN": "${SNS_TOPIC_ARN}",
    "configSnapshotDeliveryProperties": {
        "deliveryFrequency": "Six_Hours"
    }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"

```

```
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
    "name": "${DELIVERY_CHANNEL_NAME}",
    "s3BucketName": "${S3_BUCKET_NAME}",
    "snsTopicARN": "${SNS_TOPIC_ARN}",
    "configSnapshotDeliveryProperties": {
        "deliveryFrequency": "Six_Hours"
    }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
```

```
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "======"
echo "CLEANUP CONFIRMATION"
echo "======"
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- APIの詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [CreateBucket](#)
  - [CreateRole](#)

- [CreateTopic](#)
- [DeleteBucket](#)
- [DeleteConfigurationRecorder](#)
- [DeleteDeliveryChannel](#)
- [DeleteRole](#)
- [DeleteRolePolicy](#)
- [DeleteTopic](#)
- [DescribeConfigurationRecorderStatus](#)
- [DescribeConfigurationRecorders](#)
- [DescribeDeliveryChannels](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [PutConfigurationRecorder](#)
- [PutDeliveryChannel](#)
- [PutPublicAccessBlock](#)
- [PutRolePolicy](#)
- [Rm](#)
- [StartConfigurationRecorder](#)
- [StopConfigurationRecorder](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## ドキュメントテキスト抽出の使用開始

次のコード例は、以下の操作方法を示しています。

- S3 バケットを作成する
- S3 にドキュメントをアップロードする
- リソースをクリーンアップする



## Bash

### AWS CLIBash スクリプトを使用した

#### Note

GitHub には、その他のリソースもあります。[デベロッパーチュートリアル](#)のサンプルレポジトリで用例一覧を検索し、設定と実行の方法を確認してください。

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====="
echo "Amazon Textract Getting Started Tutorial"
echo "====="
echo "This script will guide you through using Amazon Textract to analyze
document text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*\1=***REDACTED***/'g'
        cleanup_on_error
        exit 1
    fi
}
```

```
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to
delete document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ];
then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
    exit 1
fi

AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
```

```
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization
\).*\/1=***REDACTED***/'
    exit 1
fi

# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
    default region."
    exit 1
fi

# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-${RANDOM_ID}"
fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
```

```

    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
{"SSEAlgorithm": "AES256"}]}'} 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text
50,50 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUgAAAAEAAAABCAyAAAAfFcSJAAAADU1EQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."

```

```
UPLOAD_OUTPUT=$(aws s3 cp ".$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256
2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp ".$DOCUMENT_NAME "s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
  "S3Object": {
    "Bucket": "BUCKET_PLACEHOLDER",
    "Name": "DOCUMENT_PLACEHOLDER"
  }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES","FORMS","SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
  echo "ERROR: Document analysis failed"
  echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*\1=***REDACTED***/'
  exit 1
fi
```

```
# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "====="
echo "Analysis Summary"
echo "====="
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk
'{print $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l ||
echo 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l ||
echo 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l ||
echo 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l ||
echo 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"'
| wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' |
wc -l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""
```

```

# Cleanup confirmation
echo ""
echo "=====
echo "RESOURCES CREATED"
echo "=====
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-
results.json) has been kept."

echo ""
echo "=====
echo "Tutorial complete!"
echo "=====
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"

```

```
echo ""  
  
trap - EXIT
```

- API の詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。
  - [AnalyzeDocument](#)
  - [Cp](#)
  - [Help](#)
  - [Mb](#)
  - [Rb](#)
  - [Rm](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## 機械学習特徴量ストアの使用開始

次のコード例は、以下の操作方法を示しています。

- IAM 許可をセットアップする
- SageMaker 実行ロールを作成する
- 特徴量グループを作成する
- リソースをクリーンアップする

### Bash

AWS CLIBash スクリプトを使用した

#### Note

GitHub には、その他のリソースもあります。[デベロッパーチュートリアル](#)の[サンプル](#)レポジトリで用例一覧を検索し、設定と実行の方法を確認してください。



```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
```

```

        --output text)
    echo "Current status: ${status}"

    if [ "$status" = "Failed" ]; then
        handle_error "Feature group ${feature_group_name} creation failed"
    fi
done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
"$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)

```

```
        echo "Unknown resource type: $resource_type"
        ;;
    esac
done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
    Key=tutorial,Value=sagemaker-featurestore 2>&1
```

```

# Attach necessary policies
echo "Attaching policies to role..." >&2

# SageMaker execution policy
local policy1_result=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

if echo "$policy1_result" | grep -i "error" > /dev/null; then
    handle_error "Failed to attach SageMaker policy: $policy1_result"
fi

# S3 access policy
local policy2_result=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

if echo "$policy2_result" | grep -i "error" > /dev/null; then
    handle_error "Failed to attach S3 policy: $policy2_result"
fi

# Get account ID for role ARN
local account_id=$(aws sts get-caller-identity --query Account --output text)
local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

echo "Role ARN: $role_arn" >&2
echo "Waiting 10 seconds for role to propagate..." >&2
sleep 10

# Return only the role ARN to stdout
echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

```

```
# Validate the role ARN
ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\\///')
ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
    echo "Creating a new role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
```

```

    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
            --region "$REGION" 2>&1)
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
    fi
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
    'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
    "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources

```

```

        exit 1
    fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-`${RANDOM_ID}`"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://${S3_BUCKET_NAME}/${PREFIX}"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

```

```

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-`${RANDOM_ID}`"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
  --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
  --record-identifier-feature-name "customer_id" \
  --event-time-feature-name "EventTime" \
  --feature-definitions '[
    {"FeatureName": "customer_id", "FeatureType": "Integral"},
    {"FeatureName": "order_id", "FeatureType": "String"},
    {"FeatureName": "order_date", "FeatureType": "String"},
    {"FeatureName": "product", "FeatureType": "String"},
    {"FeatureName": "quantity", "FeatureType": "Integral"},
    {"FeatureName": "amount", "FeatureType": "Fractional"},
    {"FeatureName": "EventTime", "FeatureType": "Fractional"}
  ]' \
  --online-store-config '{"EnableOnlineStore": true}' \
  --offline-store-config '{
    "S3StorageConfig": {
      "S3Uri": "s3://`${S3_BUCKET_NAME}`/`${PREFIX}`"
    },
    "DisableGlueTableCreation": false
  }' \
  --role-arn "$ROLE_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to create orders feature group: $ORDERS_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

```



```
# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "573291"},
    {"FeatureName": "name", "ValueAsString": "John Doe"},
    {"FeatureName": "age", "ValueAsString": "35"},
    {"FeatureName": "address", "ValueAsString": "123 Main St"},
    {"FeatureName": "membership_type", "ValueAsString": "premium"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "109382"},
    {"FeatureName": "name", "ValueAsString": "Jane Smith"},
    {"FeatureName": "age", "ValueAsString": "28"},
    {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
    {"FeatureName": "membership_type", "ValueAsString": "standard"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
```

```

--feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
--record '[
    {"FeatureName": "customer_id", "ValueAsString": "573291"},
    {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
    {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
    {"FeatureName": "product", "ValueAsString": "Laptop"},
    {"FeatureName": "quantity", "ValueAsString": "1"},
    {"FeatureName": "amount", "ValueAsString": "1299.99"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"

```

```
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
| grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi
```

```

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "======"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "======"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "======"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "======"
    echo "CLEANUP CONFIRMATION"
    echo "======"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else

```

```
    echo "Skipping cleanup. Resources will remain in your account."
    echo "To clean up later, delete the following resources:"
    echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
$ORDERS_FEATURE_GROUP_NAME"
    echo "- S3 Bucket: $S3_BUCKET_NAME"
    if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
        echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
'IAMRole:[^[:space:]]*' | cut -d: -f2)"
    fi
    echo ""
    echo "Estimated ongoing cost: ~$0.01 per month for online store"
fi
fi

echo "Script completed at $(date)"
```

- API の詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。
  - [AttachRolePolicy](#)
  - [:CreateBucket](#)
  - [CreateFeatureGroup](#)
  - [CreateRole](#)
  - [:DeleteBucket](#)
  - [DeleteFeatureGroup](#)
  - [DeleteRole](#)
  - [DescribeFeatureGroup](#)
  - [DetachRolePolicy](#)
  - [GetCallerIdentity](#)
  - [GetRole](#)
  - [ListFeatureGroups](#)
  - [PutPublicAccessBlock](#)
  - [Rm](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## オブジェクトストレージの使用開始

次のコード例は、以下の操作方法を示しています。

- 最初の S3 バケットを作成する
- オブジェクトのアップロード
- バージョニングの有効化
- デフォルトの暗号化を設定する
- バケットにタグを追加する
- オブジェクトとバージョンを一覧表示する
- リソースをクリーンアップする

### Bash

AWS CLIBash スクリプトを使用した

#### Note

GitHub には、その他のリソースもあります。[デベロッパーチュートリアル](#)のサンプルレポジトリで用例一覧を検索し、設定と実行の方法を確認してください。

```
#!/bin/bash
# S3 Getting Started - Create a bucket, upload and download objects, copy to a
# folder prefix, enable versioning, configure encryption and public access
# blocking, tag the bucket, list objects and versions, and clean up.

set -eE
set -o pipefail

# =====
# Prerequisites check
# =====

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
"$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Run 'aws configure' or set
AWS_DEFAULT_REGION."
```

```

        exit 1
    fi

    # Verify AWS credentials are configured
    if ! aws sts get-caller-identity &>/dev/null; then
        echo "ERROR: AWS credentials not configured or invalid. Run 'aws configure'."
        exit 1
    fi

    # =====
    # Setup: logging, temp directory, resource tracking
    # =====

    UNIQUE_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
    # Check for shared prereq bucket
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || true)
    if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
        BUCKET_NAME="$PREREQ_BUCKET"
        BUCKET_IS_SHARED=true
        echo "Using shared bucket: $BUCKET_NAME"
    else
        BUCKET_IS_SHARED=false
        BUCKET_NAME="s3api-`${UNIQUE_ID}`"
    fi

    TEMP_DIR=$(mktemp -d)
    trap 'rm -rf "$TEMP_DIR"' EXIT
    LOG_FILE="${TEMP_DIR}/s3-gettingstarted.log"
    CREATED_RESOURCES=()

    exec > >(tee -a "$LOG_FILE") 2>&1

    echo "======"
    echo "S3 Getting Started"
    echo "======"
    echo "Bucket name: ${BUCKET_NAME}"
    echo "Temp directory: ${TEMP_DIR}"
    echo "Log file: ${LOG_FILE}"
    echo ""

    # =====

```

```

# Helper functions
# =====

get_region() {
    echo "${AWS_REGION:-${AWS_DEFAULT_REGION:-${CONFIGURED_REGION}}}"
}

delete_object_versions() {
    local bucket=$1
    local query=$2

    local versions
    versions=$(aws s3api list-object-versions \
        --bucket "$bucket" \
        --query "$query" \
        --output json 2>&1) || return 0

    if [ -z "$versions" ] || [ "$versions" = "null" ] || [ "$versions" = "[]" ];
    then
        return 0
    fi

    echo "$versions" | jq -r '.[ ] | "\(.Key)\t\(.VersionId)"" 2>/dev/null | while
IFS=$'\t' read -r key version_id; do
    if [ -n "$key" ] && [ "$key" != "null" ]; then
        aws s3api delete-object --bucket "$bucket" --key "$key" --version-id
"$version_id" >/dev/null 2>&1 || true
    fi
done

    return 0
}

# =====
# Error handling and cleanup functions
# =====

cleanup() {
    echo ""
    echo "=====
echo "CLEANUP"
echo "=====

    if [ "$BUCKET_IS_SHARED" = "false" ]; then

```



```

        echo "Deleting all object versions in bucket..."

        delete_object_versions "$BUCKET_NAME" "Versions[.
{Key:Key,VersionId:VersionId}" || true

        delete_object_versions "$BUCKET_NAME" "DeleteMarkers[.
{Key:Key,VersionId:VersionId}" || true

        echo "Deleting bucket: ${BUCKET_NAME}"
        if ! aws s3api delete-bucket --bucket "$BUCKET_NAME" 2>/dev/null; then
            echo "WARNING: Failed to delete bucket ${BUCKET_NAME}"
        fi

        # Clean up logs bucket
        LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
        if aws s3api head-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null; then
            echo "Deleting log bucket: ${LOG_TARGET_BUCKET}"
            if ! aws s3api delete-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/
null; then
                echo "WARNING: Failed to delete bucket ${LOG_TARGET_BUCKET}"
            fi
        fi
    else
        echo "Keeping shared bucket: ${BUCKET_NAME}"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    local line_number=$1
    echo ""
    echo "======"
    echo "ERROR on line ${line_number}"
    echo "======"
    echo ""
    echo "Resources created before error:"
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        for RESOURCE in "${CREATED_RESOURCES[@]}"; do
            echo "  - ${RESOURCE}"
        done
    else
        echo "  (none)"
    fi
}

```

```

    fi
    echo ""
    echo "Attempting cleanup..."
    cleanup
    exit 1
}

trap 'handle_error "$LINENO"' ERR

# =====
# Step 1: Create a bucket
# =====

echo "Step 1: Creating bucket ${BUCKET_NAME}..."
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        if ! aws s3api create-bucket --bucket "$BUCKET_NAME" >/dev/null 2>&1;
        then
            echo "ERROR: Failed to create bucket $BUCKET_NAME"
            exit 1
        fi
    else
        if ! aws s3api create-bucket \
            --bucket "$BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
        2>&1; then
            echo "ERROR: Failed to create bucket $BUCKET_NAME in region $REGION"
            exit 1
        fi
    fi
    CREATED_RESOURCES+=("s3:bucket:${BUCKET_NAME}")
    echo "Bucket created."

    if ! aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {

```

```

        "Key": "tutorial",
        "Value": "s3-gettingstarted"
    }
]
}' >/dev/null 2>&1; then
echo "WARNING: Failed to tag bucket"
fi
fi
echo ""

# =====
# Step 2: Upload a sample text file
# =====

echo "Step 2: Uploading a sample text file..."

SAMPLE_FILE="${TEMP_DIR}/sample.txt"
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is a sample file for the getting started tutorial.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted" >/dev/null 2>&1; then
echo "ERROR: Failed to upload sample.txt"
exit 1
fi
echo "File uploaded."
echo ""

# =====
# Step 3: Download the object
# =====

echo "Step 3: Downloading the object..."

DOWNLOAD_FILE="${TEMP_DIR}/downloaded-sample.txt"
if ! aws s3api get-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    "$DOWNLOAD_FILE" >/dev/null 2>&1; then

```

```
    echo "ERROR: Failed to download sample.txt"
    exit 1
fi
echo "Downloaded to: ${DOWNLOAD_FILE}"
echo "Contents:"
cat "$DOWNLOAD_FILE"
echo ""

# =====
# Step 4: Copy the object to a folder prefix
# =====

echo "Step 4: Copying object to a folder prefix..."

if ! aws s3api copy-object \
    --bucket "$BUCKET_NAME" \
    --copy-source "${BUCKET_NAME}/sample.txt" \
    --key "backup/sample.txt" \
    --server-side-encryption AES256 \
    --metadata-directive COPY >/dev/null 2>&1; then
    echo "ERROR: Failed to copy object to backup/sample.txt"
    exit 1
fi
echo "Object copied to backup/sample.txt."
echo ""

# =====
# Step 5: Enable versioning and upload a second version
# =====

echo "Step 5: Enabling versioning..."

if ! aws s3api put-bucket-versioning \
    --bucket "$BUCKET_NAME" \
    --versioning-configuration Status=Enabled >/dev/null 2>&1; then
    echo "ERROR: Failed to enable versioning"
    exit 1
fi
echo "Versioning enabled."

echo "Uploading a second version of sample.txt..."
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is version 2 of the sample file.
EOF
```

```
if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted,version=2" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload second version of sample.txt"
    exit 1
fi
echo "Second version uploaded."
echo ""

# =====
# Step 6: Configure SSE-S3 encryption
# =====

echo "Step 6: Configuring SSE-S3 default encryption..."

if ! aws s3api put-bucket-encryption \
    --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": true
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to configure SSE-S3 encryption"
    exit 1
fi
echo "SSE-S3 encryption configured."
echo ""

# =====
# Step 7: Block all public access
# =====

echo "Step 7: Blocking all public access..."

if ! aws s3api put-public-access-block \
```

```

--bucket "$BUCKET_NAME" \
--public-access-block-configuration '{
    "BlockPublicAcls": true,
    "IgnorePublicAcls": true,
    "BlockPublicPolicy": true,
    "RestrictPublicBuckets": true
}' >/dev/null 2>&1; then
echo "ERROR: Failed to block public access"
exit 1
fi
echo "Public access blocked."
echo ""

# =====
# Step 8: Configure bucket logging
# =====

echo "Step 8: Configuring bucket logging..."

LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        aws s3api create-bucket --bucket "$LOG_TARGET_BUCKET" >/dev/null 2>&1 ||
true
    else
        aws s3api create-bucket \
            --bucket "$LOG_TARGET_BUCKET" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
2>&1 || true
    fi

    if ! aws s3api put-bucket-tagging \
        --bucket "$LOG_TARGET_BUCKET" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "ERROR: Failed to configure bucket logging"
        exit 1
    fi
fi

```

```
        }
    ]
}' >/dev/null 2>&1; then
echo "WARNING: Failed to tag log bucket"
fi

aws s3api put-bucket-acl --bucket "$LOG_TARGET_BUCKET" --acl log-delivery-
write 2>/dev/null || true

if ! aws s3api put-bucket-logging \
    --bucket "$BUCKET_NAME" \
    --bucket-logging-status '{
        "LoggingEnabled": {
            "TargetBucket": "'$LOG_TARGET_BUCKET'",
            "TargetPrefix": "logs/"
        }
    }' >/dev/null 2>&1; then
echo "WARNING: Failed to configure bucket logging"
else
echo "Bucket logging configured."
fi
else
echo "Skipping logging configuration for shared bucket."
fi
echo ""

# =====
# Step 9: Tag the bucket
# =====

echo "Step 9: Tagging the bucket..."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \
    --tagging '{
        "TagSet": [
            {
                "Key": "project",
                "Value": "doc-smith"
            },
            {
                "Key": "tutorial",
                "Value": "s3-gettingstarted"
            },
        ],
    }'
```

```
        {
            "Key": "Environment",
            "Value": "Tutorial"
        },
        {
            "Key": "Project",
            "Value": "S3-GettingStarted"
        },
        {
            "Key": "ManagedBy",
            "Value": "Bash-Tutorial"
        }
    ]
}' >/dev/null 2>&1; then
    echo "ERROR: Failed to tag bucket"
    exit 1
fi
echo "Bucket tagged."

echo "Verifying tags..."
if ! aws s3api get-bucket-tagging --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to retrieve bucket tags"
fi
echo ""

# =====
# Step 10: List objects and versions
# =====

echo "Step 10: Listing objects..."

if ! aws s3api list-objects-v2 --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list objects"
fi
echo ""

echo "Listing object versions..."

if ! aws s3api list-object-versions --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list object versions"
fi
echo ""

# =====
```



```
# Step 11: Cleanup
# =====

echo ""
echo "=====
echo "TUTORIAL COMPLETE"
echo "=====
echo ""
echo "Resources created:"
if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
    for RESOURCE in "${CREATED_RESOURCES[@}"; do
        echo "  - ${RESOURCE}"
    done
else
    echo "  (none)"
fi
echo ""
echo "=====
echo "CLEANUP"
echo "=====
echo "Cleaning up all created resources..."
cleanup

echo ""
echo "Done."
```

- API の詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [HeadObject](#)
  - [ListObjectVersions](#)
  - [ListObjectsV2](#)
  - [PutBucketEncryption](#)
  - [PutBucketTagging](#)
  - [PutBucketVersioning](#)

- [PutObject](#)
- [PutPublicAccessBlock](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## クエリ分析の使用開始

次のコード例は、以下の操作方法を示しています。

- クエリ結果用の S3 バケットを作成する
- データベースの作成
- テーブルを作成する
- クエリを実行する
- 名前付きクエリを作成して使用する
- リソースをクリーンアップする

## Bash

AWS CLIBash スクリプトを使用した

### Note

GitHub には、その他のリソースもあります。[デベロッパーチュートリアル](#)の[サンプル](#)レポジトリで用例一覧を検索し、設定と実行の方法を確認してください。

```
#!/bin/bash

# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
```

```
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi

    echo "Exiting..."
    exit 1
}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] ||
    [ ${#bucket_name} -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
}
```

```

    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z_][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
    handle_error "Failed to generate valid random ID"
fi

# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
    text 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --
output text 2>/dev/null)
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

```

```
if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi

# Security: Validate region format - expanded regex for newer regions
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~ ^[a-z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION"
2>&1)
    if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
    fi

    aws s3api put-bucket-tagging \
        --bucket "$S3_BUCKET" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-athena-gs}]'
```

```

# Security: Enable S3 bucket encryption with KMS validation
echo "Enabling default encryption on S3 bucket..."
if ! aws s3api put-bucket-encryption \
    --bucket "$S3_BUCKET" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' 2>&1; then
    echo "Warning: Could not enable encryption on bucket"
fi

# Security: Block public access
echo "Blocking public access to S3 bucket..."
if ! aws s3api put-public-access-block \
    --bucket "$S3_BUCKET" \
    --public-access-block-configuration \

"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1; then
    echo "Warning: Could not block public access on bucket"
fi

# Security: Enable versioning for data protection
echo "Enabling versioning on S3 bucket..."
if ! aws s3api put-bucket-versioning \
    --bucket "$S3_BUCKET" \
    --versioning-configuration Status=Enabled 2>&1; then
    echo "Warning: Could not enable versioning on bucket"
fi

echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then

```

```
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

# Verify the database was created
echo "Verifying database creation..."
LIST_DB_RESULT=$(aws athena list-databases --catalog-name AwsDataCatalog --region
"$AWS_REGION" 2>&1)
if echo "$LIST_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list databases: $LIST_DB_RESULT"
fi
echo "$LIST_DB_RESULT"
```

```

# Step 2: Create a table
echo "Step 2: Creating Athena table: $TABLE_NAME"
# Replace the region placeholder in the S3 location
CREATE_TABLE_QUERY="CREATE EXTERNAL TABLE IF NOT EXISTS $DATABASE_NAME.
$TABLE_NAME (
    \`Date\` DATE,
    Time STRING,
    Location STRING,
    Bytes INT,
    RequestIP STRING,
    Method STRING,
    Host STRING,
    Uri STRING,
    Status INT,
    Referrer STRING,
    os STRING,
    Browser STRING,
    BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
    \`input.regex\` = \"^(!#)([ ]+)\s+([ ]+)\s+([ ]+)\s+([ ]+)\s+
+([ ]+)\s+([ ]+)\s+([ ]+)\s+([ ]+)\s+([ ]+)\s+([ ]+)\s+([ ]+)\s+
\\\\([ ]+\\\\\\\\([ ]+\\\\\\\\;)+).*\\\\\\\\%20([ ]+\\\\\\\\/)+[\\\\\\\\/](.*)$\"
) LOCATION 's3://athena-examples-us-east-1/cloudfront/plaintext/';"

CREATE_TABLE_RESULT=$(aws athena start-query-execution \
    --query-string "$CREATE_TABLE_QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_TABLE_RESULT" | grep -qi "error\\|failed"; then
    handle_error "Failed to create table: $CREATE_TABLE_RESULT"
fi

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' |
    cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
echo "Table creation query ID: $QUERY_ID"

```



```
# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
```

```

--result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
--region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
|| echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
"$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"

```

```

fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi

DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/
dev/null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi

```

```

echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
|| echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then

```

```
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "======"
echo "RESOURCES CREATED"
echo "======"
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "======"

# Auto-confirm cleanup
echo ""
echo "======"
echo "CLEANUP CONFIRMATION"
```

```

echo "=====
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
"$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
        echo "Named query deleted successfully."
    fi

    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
        --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*'
| cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Table dropped successfully."
                    break

```

```

        elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
            echo "Warning: Table deletion failed with status:
$QUERY_STATUS"
            break
        fi
        echo "Table deletion in progress, status: $QUERY_STATUS"
        sleep 2
        ((ELAPSED+=2))
    done
fi
fi

# Drop database
echo "Dropping database: $DATABASE_NAME"
DROP_DB_RESULT=$(aws athena start-query-execution \
    --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
    echo "Warning: Failed to drop database: $DROP_DB_RESULT"
else
    QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' |
cut -d'"' -f4)
    if [ -n "$QUERY_ID" ]; then
        echo "Waiting for database deletion to complete..."

        ELAPSED=0
        while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
            QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
            if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                echo "Database dropped successfully."
                break
            elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
                echo "Warning: Database deletion failed with status:
$QUERY_STATUS"
                break
            fi
        done
    fi
fi

```

```
        echo "Database deletion in progress, status: $QUERY_STATUS"
        sleep 2
        ((ELAPSED+=2))
    done
fi
fi

# Empty and delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Emptying S3 bucket: $S3_BUCKET"
    EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
    if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
    else
        echo "S3 bucket emptied successfully."
    fi

    echo "Deleting S3 bucket: $S3_BUCKET"
    DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
    if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
    else
        echo "S3 bucket deleted successfully."
    fi
else
    echo "Skipping S3 bucket deletion (shared resource)"
fi

# Security: Remove downloaded query results
if [ -f "./query-results.csv" ]; then
    if command -v shred &>/dev/null; then
        shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
    else
        rm -f "./query-results.csv"
    fi
    echo "Query results file securely removed."
fi

echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```



- API の詳細については、AWS CLI コマンドリファレンスの以下のトピックを参照してください。
  - [Cp](#)
  - [CreateNamedQuery](#)
  - [DeleteNamedQuery](#)
  - [GetNamedQuery](#)
  - [GetQueryExecution](#)
  - [GetQueryResults](#)
  - [ListDatabases](#)
  - [ListNamedQueries](#)
  - [ListTableMetadata](#)
  - [Mb](#)
  - [Rb](#)
  - [Rm](#)
  - [StartQueryExecution](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 オブジェクトロック機能を実行する

次のコード例は、S3 オブジェクトロック機能を使用する方法を示しています。

.NET

SDK for .NET

### Note

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

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

        This .NET example performs the following tasks:
        1. Create test Amazon Simple Storage Service (S3) buckets with different
        lock policies.
        2. Upload sample objects to each bucket.
        3. Set some Legal Hold and Retention Periods on objects and buckets.
        4. Investigate lock policies by viewing settings or attempting to delete
        or overwrite objects.
        5. Clean up objects and buckets.
    */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
    private static string _resourcePrefix = null!;
    private static string noLockBucketName = null!;
    private static string lockEnabledBucketName = null!;
    private static string retentionAfterCreationBucketName = null!;
    private static List<string> bucketNames = new List<string>();
    private static List<string> fileNames = new List<string>();

    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<IAmazonS3>()
        .AddTransient<S3ActionsWrapper>()
    )
    .Build();

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load settings from .json file.
    .AddJsonFile("settings.local.json",
        true) // Optionally, load local settings.
    .Build();

ConfigurationSetup();

ServicesSetup(host);

try
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
    Console.WriteLine(new string('-', 80));
    await Setup(true);

    await DemoActionChoices();

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Cleaning up resources.");
    Console.WriteLine(new string('-', 80));
    await Cleanup(true);

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Amazon S3 Object Locking Scenario is complete.");
    Console.WriteLine(new string('-', 80));
}
catch (Exception ex)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"There was a problem: {ex.Message}");
}
```

```
        await Cleanup(true);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
}

/// <summary>
/// Any setup operations needed.
/// </summary>
public static void ConfigurationSetup()
{
    _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

    noLockBucketName = _resourcePrefix + "-no-lock";
    lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
    retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-creation";

    bucketNames.Add(noLockBucketName);
    bucketNames.Add(lockEnabledBucketName);
    bucketNames.Add(retentionAfterCreationBucketName);
}

// <summary>
// Deploy necessary resources for the scenario.
// </summary>
// <param name="interactive">True to run as interactive.</param>
// <returns>True if successful.</returns>
public static async Task<bool> Setup(bool interactive)
{
    Console.WriteLine(
        "\nFor this scenario, we will use the AWS SDK for .NET to create
several S3\n" +
        "buckets and files to demonstrate working with S3 locking features.
\n");
}
```

```
Console.WriteLine(new string('-', 80));
Console.WriteLine("Press Enter when you are ready to start.");
if (interactive)
    Console.ReadLine();

Console.WriteLine("\nS3 buckets can be created either with or without
object lock enabled.");
await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName,
false);
await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
true);
await
_s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
false);

Console.WriteLine("Press Enter to continue.");
if (interactive)
    Console.ReadLine();

Console.WriteLine("\nA bucket can be configured to use object locking
with a default retention period.");
await
_s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
true,
    ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

Console.WriteLine("Press Enter to continue.");
if (interactive)
    Console.ReadLine();

Console.WriteLine("\nObject lock policies can also be added to existing
buckets.");
await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);

Console.WriteLine("Press Enter to continue.");
if (interactive)
    Console.ReadLine();

// Upload some files to the buckets.
Console.WriteLine("\nNow let's add some test files:");
var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
int fileCount = 2;
// Create the file if it does not already exist.
if (!File.Exists(fileName))
```

```
{
    await using StreamWriter sw = File.CreateText(fileName);
    await sw.WriteLineAsync(
        "This is a sample file for uploading to a bucket.");
}

foreach (var bucketName in bucketNames)
{
    for (int i = 0; i < fileCount; i++)
    {
        var numberedFileName = Path.GetFileNameWithoutExtension(fileName)
+ i + Path.GetExtension(fileName);
        fileNames.Add(numberedFileName);
        await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
    }
}
Console.WriteLine("Press Enter to continue.");
if (interactive)
    Console.ReadLine();

if (!interactive)
    return true;
Console.WriteLine("\nNow we can set some object lock policies on
individual files:");
foreach (var bucketName in bucketNames)
{
    for (int i = 0; i < fileNames.Count; i++)
    {
        // No modifications to the objects in the first bucket.
        if (bucketName != bucketNames[0])
        {
            var exampleFileName = fileNames[i];
            switch (i)
            {
                case 0:
                {
                    var question =
                        $"Would you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                    if (GetYesNoResponse(question))
                    {
                        // Set a legal hold.
```

```

        await
        _s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
        ObjectLockLegalHoldStatus.On);

        }
        break;
    }
    case 1:
    {
        var question =
            $"\\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
            "\\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
        if (GetYesNoResponse(question))
        {
            // Set a Governance mode retention period for
1 day.

            await
            _s3ActionsWrapper.ModifyObjectRetentionPeriod(
                bucketName, exampleFileName,
                ObjectLockRetentionMode.Governance,
                DateTime.UtcNow.AddDays(1));
        }
        break;
    }
}

}

}

}

Console.WriteLine(new string('-', 80));
return true;
}

// <summary>
/// List all of the current buckets and objects.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>The list of buckets and objects.</returns>
public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
{
    var allObjects = new List<S3ObjectVersion>();

```

```
        foreach (var bucketName in bucketNames)
        {
            var objectsInBucket = await
                _s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
            foreach (var objectKey in objectsInBucket.Versions)
            {
                allObjects.Add(objectKey);
            }
        }

        if (interactive)
        {
            Console.WriteLine("\nCurrent buckets and objects:\n");
            int i = 0;
            foreach (var bucketObject in allObjects)
            {
                i++;
                Console.WriteLine(
                    $"{i}: {bucketObject.Key} \n\tBucket: {bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
            }
        }

        return allObjects;
    }

    /// <summary>
    /// Present the user with the demo action choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<bool> DemoActionChoices()
    {
        var choices = new string[]{
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the scenario."};

        var choice = 0;
        // Keep asking the user until they choose to move on.
        while (choice != 6)
```



```
{
    Console.WriteLine(new string('-', 80));
    choice = GetChoiceResponse(
        "\nExplore the S3 locking features by selecting one of the
following choices:"
        , choices);
    Console.WriteLine(new string('-', 80));
    switch (choice)
    {
        case 0:
        {
            await ListBucketsAndObjects(true);
            break;
        }
        case 1:
        {
            Console.WriteLine("\nEnter the number of the object to
delete:");

            var allFiles = await ListBucketsAndObjects(true);
            var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
            await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
            break;
        }
        case 2:
        {
            Console.WriteLine("\nEnter the number of the object to
delete:");

            var allFiles = await ListBucketsAndObjects(true);
            var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
            await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
            break;
        }
        case 3:
        {
            var allFiles = await ListBucketsAndObjects(true);
            Console.WriteLine("\nEnter the number of the object to
overwrite:");
```

```

        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        // Create the file if it does not already exist.
        if (!File.Exists(allFiles[fileChoice].Key))
        {
            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a
bucket.");
        }
        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
        break;
    }
    case 4:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
        break;
    }
    case 5:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object to
view:");
        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
        break;
    }
    }
}

```

```
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
        {
            // Remove all locks and delete all buckets and objects.
            var allFiles = await ListBucketsAndObjects(false);
            foreach (var fileInfo in allFiles)
            {
                // Check for a legal hold.
                var legalHold = await
                _s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
                if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
                {
                    await
                    _s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
                    ObjectLockLegalHoldStatus.Off);
                }

                // Check for a retention period.
                var retention = await
                _s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
                var hasRetentionPeriod = retention?.Mode ==
                ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
                DateTime.UtcNow.Date;
                await
                _s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
                hasRetentionPeriod, fileInfo.VersionId);
            }

            foreach (var bucketName in bucketNames)
            {
                await _s3ActionsWrapper.DeleteBucketByName(bucketName);
            }
        }
    }
}
```

```

    }
    else
    {
        Console.WriteLine(
            "Ok, we'll leave the resources intact.\n" +
            "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
        );
    }

    Console.WriteLine(new string('-', 80));
    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>
/// Helper method to get a choice response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <param name="choices">The choices to print on the console.</param>
/// <returns>The index of the selected choice</returns>
private static int GetChoiceResponse(string? question, string[] choices)
{
    if (question != null)
    {
        Console.WriteLine(question);

        for (int i = 0; i < choices.Length; i++)

```

```
        {
            Console.WriteLine($"{t{i + 1}. {choices[i]}");
        }
    }

    var choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > choices.Length)
    {
        var choice = Console.ReadLine();
        Int32.TryParse(choice, out choiceNumber);
    }

    return choiceNumber - 1;
}
}
```

### S3 関数のラッパークラス。

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }
}
```

```
/// <summary>
/// Create a new Amazon S3 bucket with object lock actions.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
{
    Console.WriteLine($"\\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
            ObjectLockEnabledForBucket = enableObjectLock,
        };

        var response = await _amazonS3.PutBucketAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Enable object lock on an existing bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to modify.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableObjectLockOnBucket(string bucketName)
{
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
```

```

        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
            },
        };

        var response = await
        _amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"{bucketName}\tAdded an object lock policy to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Set or modify a retention period on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="retention">The retention mode.</param>
/// <param name="retainUntilDate">The date retention expires.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
    string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
{
    try

```

```

    {
        var request = new PutObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            Retention = new ObjectLockRetention()
            {
                Mode = retention,
                RetainUntilDate = retainUntilDate
            }
        };

        var response = await _amazonS3.PutObjectRetentionAsync(request);
        Console.WriteLine($"\\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Set or modify a retention period on an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to modify.</param>
/// <param name="retention">The retention mode.</param>
/// <param name="retainUntilDate">The date for retention until.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
{
    var enabledString = enableObjectLock ? "Enabled" : "Disabled";
    var timeDifference = retainUntilDate.Subtract(DateTime.Now);
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,

```



```

        VersioningConfig = new S3BucketVersioningConfig()
        {
            EnableMfaDelete = false,
            Status = VersionStatus.Enabled
        }
    });

    var request = new PutObjectLockConfigurationRequest()
    {
        BucketName = bucketName,
        ObjectLockConfiguration = new ObjectLockConfiguration()
        {
            ObjectLockEnabled = new ObjectLockEnabled(enabledString),
            Rule = new ObjectLockRule()
            {
                DefaultRetention = new DefaultRetention()
                {
                    Mode = retention,
                    Days = timeDifference.Days // Can be specified in
days or years but not both.
                }
            }
        }
    };

    var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
    Console.WriteLine($"{Environment.NewLine}Added a default retention to bucket
{bucketName}.");
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"{Environment.NewLine}Error modifying object lock: '{ex.Message}'");
    return false;
}
}

/// <summary>
/// Get the retention period for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object retention details.</returns>

```

```

public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectRetentionAsync(request);
        Console.WriteLine($"Object retention for {objectKey} in
{bucketName}: " +
            $"{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}");
        return response.Retention;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to fetch object lock retention:
'{ex.Message}'");
        return new ObjectLockRetention();
    }
}

/// <summary>
/// Set or modify a legal hold on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="holdStatus">The On or Off status for the legal hold.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectLegalHold(string bucketName,
    string objectKey, ObjectLockLegalHoldStatus holdStatus)
{
    try
    {
        var request = new PutObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            LegalHold = new ObjectLockLegalHold()
        }
    }
}

```

```

        Status = holdStatus
    }
};

var response = await _amazonS3.PutObjectLegalHoldAsync(request);
Console.WriteLine($"\\tModified legal hold for {objectKey} in
{bucketName}.");
return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tError modifying legal hold: '{ex.Message}'");
    return false;
}
}

/// <summary>
/// Get the legal hold details for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object legal hold details.</returns>
public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectLegalHoldAsync(request);
        Console.WriteLine($"\\tObject legal hold for {objectKey} in
{bucketName}: " +
            $"\\n\\tStatus: {response.LegalHold.Status}");
        return response.LegalHold;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch legal hold: '{ex.Message}'");
        return new ObjectLockLegalHold();
    }
}

```

```

    }

    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
    GetBucketObjectLockConfiguration(string bucketName)
    {
        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };

            var response = await
            _amazonS3.GetObjectLockConfigurationAsync(request);
            Console.WriteLine($"\\tBucket object lock config for {bucketName} in
{bucketName}: " +
                $"\\n\\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                $"\\n\\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }

    /// <summary>
    /// Upload a file from the local computer to an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.</returns>

```

```
public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
{
    var request = new PutObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
        FilePath = filePath,
        ChecksumAlgorithm = ChecksumAlgorithm.SHA256
    };

    var response = await _amazonS3.PutObjectAsync(request);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{bucketName}\tSuccessfully uploaded {objectName} to
{bucketName}.");
        return true;
    }
    else
    {
        Console.WriteLine($"{bucketName}\tCould not upload {objectName} to
{bucketName}.");
        return false;
    }
}

/// <summary>
/// List bucket objects and versions.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <returns>The list of objects and versions.</returns>
public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
bucketName)
{
    var request = new ListVersionsRequest()
    {
        BucketName = bucketName
    };

    var response = await _amazonS3.ListVersionsAsync(request);
    return response;
}

/// <summary>
```

```

    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey, bool hasRetention, string? versionId = null)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Unable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>

```

```
/// <returns>True if successful.</returns>
public async Task<bool> DeleteBucketByName(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest() { BucketName = bucketName, };
        var response = await _amazonS3.DeleteBucketAsync(request);
        Console.WriteLine($"\\tDelete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
            ex.Message);
        return false;
    }
}
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
  - [GetObjectLegalHold](#)
  - [GetObjectLockConfiguration](#)
  - [GetObjectRetention](#)
  - [PutObjectLegalHold](#)
  - [PutObjectLockConfiguration](#)
  - [PutObjectRetention](#)

Go

SDK for Go V2

 Note

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

Amazon S3 オブジェクトロック機能を示すインタラクティブなシナリオを実行します。

```
import (
    "context"
    "fmt"
    "log"
    "strings"

    "s3_object_lock/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
    questioner demotools.IQuestioner
    resources  Resources
    s3Actions  *actions.S3Actions
    sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner
    demotools.IQuestioner) ObjectLockScenario {
    scenario := ObjectLockScenario{
        questioner: questioner,
        resources:  Resources{},
        s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
        sdkConfig:  sdkConfig,
    }
    scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
    scenario.resources.init(scenario.s3Actions, questioner)
    return scenario
}

type nameLocked struct {
    name  string
    locked bool
}
```



```
var createInfo = []nameLocked{
    {"standard-bucket", false},
    {"lock-bucket", true},
    {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
    log.Println("Let's create some S3 buckets to use for this workflow.")
    success := false
    for !success {
        prefix := scenario.questioner.Ask(
            "This example creates three buckets. Enter a prefix to name your buckets\n(remember bucket names must be globally unique):")

        for _, info := range createInfo {
            log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
            bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
                fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
            if err != nil {
                switch err.(type) {
                    case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
                        log.Printf("Couldn't create bucket %s.\n", bucketName)
                    default:
                        panic(err)
                }
                break
            }
            scenario.resources.demoBuckets[info.name] = &DemoBucket{
                name:      bucketName,
                objectKeys: []string{},
            }
            log.Printf("Created bucket %s.\n", bucketName)
        }

        if len(scenario.resources.demoBuckets) < len(createInfo) {
            scenario.resources.deleteBuckets(ctx)
        } else {
            success = true
        }
    }

    log.Println("S3 buckets created.")
}
```

```
log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
    log.Println("\nA bucket can be configured to use object locking.")
    scenario.questioner.Ask("Press Enter to continue.")

    var err error
    bucket := scenario.resources.demoBuckets["retention-bucket"]
    err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
        default:
            panic(err)
        }
    } else {
        log.Printf("Object locking enabled on bucket %s.", bucket.name)
    }

    log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a
// bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
context.Context) {
    log.Println("\nA bucket can be configured to use object locking with a default
retention period.")

    bucket := scenario.resources.demoBuckets["retention-bucket"]
    retentionPeriod := scenario.questioner.AskInt("Enter the default retention
period in days: ")
    err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
types.ObjectLockRetentionModeGovernance)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("Couldn't configure a default retention period on bucket %s.\n",
bucket.name)
        default:
```

```
        panic(err)
    }
} else {
    log.Printf("Default retention policy set on bucket %s with %d day retention
period.", bucket.name, retentionPeriod)
    bucket.retentionEnabled = true
}

log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
    log.Println("Uploading test objects to S3 buckets.")

    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        for i := 0; i < 2; i++ {
            key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
fmt.Sprintf("example-%d", i),
            fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
            if err != nil {
                switch err.(type) {
                case *types.NoSuchBucket:
                    log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
                default:
                    panic(err)
                }
            } else {
                log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
                bucket.objectKeys = append(bucket.objectKeys, key)
            }
        }
    }

    scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
    log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
context.Context) {
    log.Println("Now let's set object lock configurations on individual objects.")
}
```

```

buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
scenario.resources.demoBuckets["retention-bucket"]}
for _, bucket := range buckets {
    for index, objKey := range bucket.objectKeys {
        switch index {
        case 0:
            if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
to %s in %s (y/n)? ", objKey, bucket.name), "y") {
                err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
types.ObjectLockLegalHoldStatusOn)
                if err != nil {
                    switch err.(type) {
                    case *types.NoSuchKey:
                        log.Printf("Couldn't set legal hold on %s.\n", objKey)
                    default:
                        panic(err)
                    }
                } else {
                    log.Printf("Legal hold set on %s.\n", objKey)
                }
            }
        case 1:
            q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
%s in %s?\n"+
                "Reminder: Only a user with the s3:BypassGovernanceRetention permission is
able to delete this object\n"+
                "or its bucket until the retention period has expired. (y/n) ", objKey,
bucket.name)
            if scenario.questioner.AskBool(q, "y") {
                err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
types.ObjectLockRetentionModeGovernance, 1)
                if err != nil {
                    switch err.(type) {
                    case *types.NoSuchKey:
                        log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
bucket.name)
                    default:
                        panic(err)
                    }
                } else {
                    log.Printf("Retention period set to 1 for %s.", objKey)
                    bucket.retentionEnabled = true
                }
            }
        }
    }
}

```

```

    }
    }
    }
}
log.Println(strings.Repeat("-", 88))
}

const (
    ListAll = iota
    DeleteObject
    DeleteRetentionObject
    OverwriteObject
    ViewRetention
    ViewLegalHold
    Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
// object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
    log.Println("Now you can interact with the objects to explore the object lock
    configurations.")
    interactiveChoices := []string{
        "List all objects and buckets.",
        "Attempt to delete an object.",
        "Attempt to delete an object with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the retention settings for an object.",
        "View the legal hold settings for an object.",
        "Finish the workflow."}

    choice := ListAll
    for choice != Finish {
        objList := scenario.GetAllObjects(ctx)
        objChoices := scenario.makeObjectChoiceList(objList)
        choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
        interactiveChoices)
        switch choice {
        case ListAll:
            log.Println("The current objects in the example buckets are:")
            for _, objChoice := range objChoices {
                log.Println("\t", objChoice)
            }
        case DeleteObject, DeleteRetentionObject:

```

```
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
delete:\n", objChoices)
    obj := objList[objChoice]
    deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
obj.versionId, choice == DeleteRetentionObject)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Println("Nothing to delete.")
        default:
            panic(err)
        }
    } else if deleted {
        log.Printf("Object %s deleted.\n", obj.key)
    }
    case OverwriteObject:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
        obj := objList[objChoice]
        _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Println("Couldn't upload to nonexistent bucket.")
            default:
                panic(err)
            }
        } else {
            log.Printf("Uploaded new content to object %s.\n", obj.key)
        }
    case ViewRetention:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
        obj := objList[objChoice]
        retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
obj.key)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Can't get retention configuration for %s.\n", obj.key)
            default:
                panic(err)
            }
        }
```

```

    } else if retention != nil {
        log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
            retention.Mode, retention.RetainUntilDate)
    } else {
        log.Printf("Object %s does not have object retention configured.\n", obj.key)
    }
    case ViewLegalHold:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
            view:\n", objChoices)
        obj := objList[objChoice]
        legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
            obj.key, obj.versionId)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
            default:
                panic(err)
            }
        } else if legalHold != nil {
            log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
        } else {
            log.Printf("Object %s does not have legal hold configured.", obj.key)
        }
    case Finish:
        log.Println("Let's clean up.")
    }
    log.Println(strings.Repeat("-", 88))
}

type BucketKeyVersionId struct {
    bucket    string
    key       string
    versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
// them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
    []BucketKeyVersionId {
    var objectList []BucketKeyVersionId
    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]

```

```

versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
        log.Printf("Couldn't get object versions for %s.\n", bucket.name)
    default:
        panic(err)
    }
} else {
    for _, version := range versions {
        objectList = append(objectList,
            BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
                *version.VersionId})
    }
}
return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
// are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
[]BucketKeyVersionId) []string {
    choices := make([]string, len(bucketObjects))
    for i := 0; i < len(bucketObjects); i++ {
        choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
            bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
    }
    return choices
}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := scenario.questioner.(*demotools.MockQuestioner)
            if isMock || scenario.questioner.AskBool("Do you want to see the full error
message (y/n)?", "y") {
                log.Println(r)
            }
            scenario.resources.Cleanup(ctx)
        }
    }
}

```



```
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
log.Println(strings.Repeat("-", 88))

scenario.CreateBuckets(ctx)
scenario.EnableLockOnBucket(ctx)
scenario.SetDefaultRetentionPolicy(ctx)
scenario.UploadTestObjects(ctx)
scenario.SetObjectLockConfigurations(ctx)
scenario.InteractWithObjects(ctx)

scenario.resources.Cleanup(ctx)

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

この例で使用されている S3 アクションをラップする構造体を定義します。

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client *s3.Client
```

```
S3Manager *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking
// enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    }

    if enableObjectLock {
        input.ObjectLockEnabledForBucket = aws.Bool(true)
    }

    _, err := actor.S3Client.CreateBucket(ctx, input)
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", bucket)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", bucket)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
        }
    }

    return bucket, err
}
```

```
// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
    var status *types.ObjectLockLegalHoldStatus
    input := &s3.GetObjectLegalHoldInput{
        Bucket:    aws.String(bucket),
        Key:        aws.String(key),
        VersionId: aws.String(versionId),
    }

    output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
    if err != nil {
        var noSuchKeyErr *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noSuchKeyErr) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noSuchKeyErr
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "NoSuchObjectLockConfiguration":
                log.Printf("Object %s does not have an object lock configuration.\n", key)
                err = nil
            case "InvalidRequest":
                log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
                err = nil
            }
        }
    } else {
        status = &output.LegalHold.Status
    }

    return status, err
}

// GetObjectLockConfiguration retrieves the object lock configuration for an S3
    bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
    string) (*types.ObjectLockConfiguration, error) {
    var lockConfig *types.ObjectLockConfiguration
    input := &s3.GetObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
```

```

    }

    output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
"ObjectLockConfigurationNotFoundError" {
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    } else {
        lockConfig = output.ObjectLockConfiguration
    }

    return lockConfig, err
}

// GetObjectRetention retrieves the object retention configuration for an S3
// object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
string) (*types.ObjectLockRetention, error) {
    var retention *types.ObjectLockRetention
    input := &s3.GetObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }

    output, err := actor.S3Client.GetObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "NoSuchObjectLockConfiguration":
                err = nil
            }
        }
    }
}

```

```
    case "InvalidRequest":
        log.Printf("Bucket %s does not have locking enabled.", bucket)
        err = nil
    }
}
} else {
    retention = output.Retention
}

return retention, err
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }

    _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

```
// ModifyDefaultBucketRetention modifies the default retention period of an
// existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }

    _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}

// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
    string) error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
            MFADelete: types.MFADeleteDisabled,
            Status:    types.BucketVersioningStatusEnabled,
        },
    }

    _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
```

```

if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
    return err
}

input := &s3.PutObjectLockConfigurationInput{
    Bucket: aws.String(bucket),
    ObjectLockConfiguration: &types.ObjectLockConfiguration{
        ObjectLockEnabled: types.ObjectLockEnabledEnabled,
    },
}
_, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
}

return err
}

// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
error {
    input := &s3.PutObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        Retention: &types.ObjectLockRetention{
            Mode:          retentionMode,
            RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
        },
        BypassGovernanceRetention: aws.Bool(true),
    }
    _, err := actor.S3Client.PutObjectRetention(ctx, input)

```

```
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
string, contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:      aws.String(bucket),
        Key:         aws.String(key),
        Body:        bytes.NewReader([]byte(contents)),
        ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
    }
    output, err := actor.S3Manager.Upload(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    } else {
        err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
            Bucket: aws.String(bucket),
            Key:    aws.String(key),
        }, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
bucket)
        } else {
            outKey = *output.Key
        }
    }
    return outKey, err
}
```



```
// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
    var versions []types.ObjectVersion
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
    for versionPaginator.HasMorePages() {
        output, err = versionPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
            break
        } else {
            versions = append(versions, output.Versions...)
        }
    }
    return versions, err
}
```

```
// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
```

```

if err != nil {
    var noKey *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in %s.\n", key, bucket)
        err = noKey
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "AccessDenied":
            log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
            err = nil
        case "InvalidArgument":
            if bypassGovernance {
                log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
                err = nil
            }
        }
    } else {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
        } else {
            deleted = true
        }
    }
    return deleted, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),

```

```
Delete: &types.Delete{
    Objects: objects,
    Quiet:   aws.Bool(true),
},
}
if bypassGovernance {
    input.BypassGovernanceRetention = aws.Bool(true)
}
delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
if err != nil || len(delOut.Errors) > 0 {
    log.Printf("Error deleting objects from bucket %s.\n", bucket)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    } else if len(delOut.Errors) > 0 {
        for _, outErr := range delOut.Errors {
            log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
        }
        err = fmt.Errorf("%s", *delOut.Errors[0].Message)
    }
} else {
    for _, delObjs := range delOut.Deleted {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                *delObjs.Key)
        } else {
            log.Printf("Deleted %s.\n", *delObjs.Key)
        }
    }
}
return err
}
```

リソースをクリーンアップします。

```
import (
    "context"
    "log"
    "s3_object_lock/actions"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
    name            string
    retentionEnabled bool
    objectKeys      []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
// and handles
// cleanup when the scenario finishes.
type Resources struct {
    demoBuckets map[string]*DemoBucket

    s3Actions  *actions.S3Actions
    questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
    demotools.IQuestioner) {
    resources.s3Actions = s3Actions
    resources.questioner = questioner
    resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
        }
    }()
}
```

```

    log.Println("Use the AWS Management Console to remove any remaining resources
" +
    "that were created for this scenario.")
}
}()

wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
resources that were created "+
"during this demo (y/n)?", "y")
if !wantDelete {
    log.Println("Be sure to remove resources when you're done with them to avoid
unexpected charges!")
    return
}

log.Println("Removing objects from S3 buckets and deleting buckets...")
resources.deleteBuckets(ctx)
//resources.deleteRetentionObjects(resources.retentionBucket,
resources.retentionObjects)

log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
    for _, info := range createInfo {
        bucket := resources.demoBuckets[info.name]
        resources.deleteObjects(ctx, bucket)
        _, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
            Bucket: aws.String(bucket.name),
        })
        if err != nil {
            panic(err)
        }
    }
    for _, info := range createInfo {
        bucket := resources.demoBuckets[info.name]
        err := s3.NewBucketNotExistsWaiter(resources.s3Actions.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket.name)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
bucket.name)
        } else {

```

```

    log.Printf("Deleted %s.\n", bucket.name)
}
}
resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket
    *DemoBucket) {
    lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
        bucket.name)
    if err != nil {
        panic(err)
    }
    versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("No objects to get from %s.\n", bucket.name)
        default:
            panic(err)
        }
    }
    delObjects := make([]types.ObjectIdentifier, len(versions))
    for i, version := range versions {
        if lockConfig != nil && lockConfig.ObjectLockEnabled ==
            types.ObjectLockEnabledEnabled {
            status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
                *version.Key, *version.VersionId)
            if err != nil {
                switch err.(type) {
                case *types.NoSuchKey:
                    log.Printf("Couldn't determine legal hold status for %s in %s.\n",
                        *version.Key, bucket.name)
                default:
                    panic(err)
                }
            } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
                err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
                    *version.VersionId, types.ObjectLockLegalHoldStatusOff)
                if err != nil {
                    switch err.(type) {
                    case *types.NoSuchKey:

```

```
    log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
bucket.name)
    default:
        panic(err)
    }
}
}
}
del0bjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.Delete0bjects(ctx, bucket.name, del0bjects,
bucket.retentionEnabled)
if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
        log.Println("Nothing to delete.")
    default:
        panic(err)
    }
}
}
```

- API の詳細については、「AWS SDK for Go API リファレンス」の以下のトピックを参照してください。
  - [GetObjectLegalHold](#)
  - [GetObjectLockConfiguration](#)
  - [GetObjectRetention](#)
  - [PutObjectLegalHold](#)
  - [PutObjectLockConfiguration](#)
  - [PutObjectRetention](#)

## Java

### SDK for Java 2.x

#### Note

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

Amazon S3 オブジェクトロック機能を示すインタラクティブなシナリオを実行します。

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
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/setup.html

This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
    policies.
    2. Upload sample objects to each bucket.
    3. Set some Legal Hold and Retention Periods on objects and buckets.
    4. Investigate lock policies by viewing settings or attempting to delete or
    overwrite objects.
    5. Clean up objects and buckets.
*/
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    static String bucketName;
    static S3LockActions s3LockActions;
```



```

private static final List<String> bucketNames = new ArrayList<>();
private static final List<String> fileNames = new ArrayList<>();

public static void main(String[] args) {
    final String usage = ""
        Usage:
            <bucketName> \s

        Where:
            bucketName - The Amazon S3 bucket name.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }
    s3LockActions = new S3LockActions();
    bucketName = args[0];
    Scanner scanner = new Scanner(System.in);

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();
    configurationSetup();
    System.out.println(DASHES);

    System.out.println(DASHES);
    setup();
    System.out.println("Setup is complete. Press Enter to continue...");
    scanner.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Lets present the user with choices.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();
    demoActionChoices() ;
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Would you like to clean up the resources? (y/n)");
    String delAns = scanner.nextLine().trim();

```

```
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by
selecting one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to
delete:");

                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
```

```
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        String version = allFiles.get(fileChoice).getVersion();
        s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
    }

    case 2 -> {
        System.out.println("Enter the number of the object to
delete:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        String version = allFiles.get(fileChoice).getVersion();
        s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
    }

    case 3 -> {
        System.out.println("Enter the number of the object to
overwrite:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();

        // Attempt to overwrite the file.
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
            writer.write("This is a modified text.");
        }
    }
}
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
        s3LockActions.uploadFile(bucketName, objectKey, objectKey);
    }

    case 4 -> {
        System.out.println("Enter the number of the object to
overwrite:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectRetention(bucketName, objectKey);
    }

    case 5 -> {
        System.out.println("Enter the number of the object to
view:");
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectLegalHold(bucketName, objectKey);
        s3LockActions.getBucketObjectLockConfiguration(bucketName);
    }

    case 6 -> {
        System.out.println("Exiting the workflow...");
        return;
    }

    default -> {
        System.out.println("Invalid choice. Please select again.");
    }
}
```

```

    }
}

// Clean up the resources from the scenario.
private static void cleanup() {
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
    for (S3InfoObject fileInfo : allFiles) {
        String bucketName = fileInfo.getBucketName();
        String key = fileInfo.getKeyName();
        String version = fileInfo.getVersion();
        if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
            ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
            if (legalHold != null) {
                String holdStatus = legalHold.status().name();
                System.out.println(holdStatus);
                if (holdStatus.compareTo("ON") == 0) {
                    s3LockActions.modifyObjectLegalHold(bucketName, key,
false);
                }
            }
            // Check for a retention period.
            ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
            boolean hasRetentionPeriod ;
            hasRetentionPeriod = retention != null;
            s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

        } else {
            System.out.println(bucketName +" objects do not have a legal
lock");
            s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
        }
    }
}

// Delete the buckets.
System.out.println("Delete "+bucketName);
for (String bucket : bucketNames){
    s3LockActions.deleteBucketByName(bucket);
}

```

```
}

private static void setup() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("""
        For this workflow, we will use the AWS SDK for Java to create
several S3
        buckets and files to demonstrate working with S3 locking
features.
        """);

    System.out.println("S3 buckets can be created either with or without
object lock enabled.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();

    // Create three S3 buckets.
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
    s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Bucket "+bucketNames.get(2) +" will be configured to
use object locking with a default retention period.");
    s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
    s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    // Upload some files to the buckets.
    System.out.println("Now let's add some test files:");
    String fileName = "exampleFile.txt";
    int fileCount = 2;
    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
        writer.write("This is a sample file for uploading to a bucket.");

    } catch (IOException e) {
```

```

        e.printStackTrace();
    }

    for (String bucketName : bucketNames){
        for (int i = 0; i < fileCount; i++) {
            // Get the file name without extension.
            String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
            int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
            if (extensionIndex > 0) {
                fileNameWithoutExtension =
fileNameWithoutExtension.substring(0, extensionIndex);
            }

            // Create the numbered file names.
            String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
            fileNames.add(numberedFileName);
            s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
        }
    }

    String question = null;
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    System.out.println("Now we can set some object lock policies on
individual files:");
    for (String bucketName : bucketNames) {
        for (int i = 0; i < fileNames.size(); i++){

            // No modifications to the objects in the first bucket.
            if (!bucketName.equals(bucketNames.get(0))) {
                String exampleFileName = fileNames.get(i);
                switch (i) {
                    case 0 -> {
                        question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                        System.out.println(question);
                        String ans = scanner.nextLine().trim();
                        if (ans.equalsIgnoreCase("y")) {
                            System.out.println("**** You have selected to put
a legal hold " + exampleFileName);

                            // Set a legal hold.

```

```

        s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
    }
}
case 1 -> {
    ""
    Would you like to add a 1 day Governance
retention period to %s in %s (y/n)?
    Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
    "".formatted(exampleFileName, bucketName);
    System.out.println(question);
    String ans2 = scanner.nextLine().trim();
    if (ans2.equalsIgnoreCase("y")) {

s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
    }
    }
    }
    }
}

// Get file extension.
private static String getFileExtension(String fileName) {
    int dotIndex = fileName.lastIndexOf('.');
    if (dotIndex > 0) {
        return fileName.substring(dotIndex);
    }
    return "";
}

public static void configurationSetup() {
    String noLockBucketName = bucketName + "-no-lock";
    String lockEnabledBucketName = bucketName + "-lock-enabled";
    String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
    bucketNames.add(noLockBucketName);
    bucketNames.add(lockEnabledBucketName);
    bucketNames.add(retentionAfterCreationBucketName);
}

```



```
public static int getChoiceResponse(String question, String[] choices) {
    Scanner scanner = new Scanner(System.in);
    if (question != null) {
        System.out.println(question);
        for (int i = 0; i < choices.length; i++) {
            System.out.println("\t" + (i + 1) + ". " + choices[i]);
        }
    }

    int choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > choices.length) {
        String choice = scanner.nextLine();
        try {
            choiceNumber = Integer.parseInt(choice);
        } catch (NumberFormatException e) {
            System.out.println("Invalid choice. Please enter a valid
number.");
        }
    }

    return choiceNumber - 1;
}
```

## S3 関数のラッパークラス。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
```

```
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
    software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
    {
```

```
// Calculate the instant one day from now.
Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

// Convert the Instant to a ZonedDateTime object with a specific time
zone.
ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

// Define a formatter for human-readable output.
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

// Format the ZonedDateTime object to a human-readable date string.
String humanReadableDate = formatter.format(zonedDateTime);

// Print the formatted date string.
System.out.println("Formatted Date: " + humanReadableDate);
ObjectLockRetention retention = ObjectLockRetention.builder()
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .retainUntilDate(futureInstant)
    .build();

PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .retention(retention)
    .build();

getClient().putObjectRetention(retentionRequest);
System.out.println("Set retention for "+objectKey+" in "+bucketName+"
until "+ humanReadableDate +".");
}

// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();
```

```
        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
        ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
    }

    return null;
}

// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}

public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
    AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
    return bucketNames.stream()
        .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
        .map(version -> {
            S3InfoObject s3InfoObject = new S3InfoObject();
            s3InfoObject.setBucketName(bucketName);
            s3InfoObject.setVersion(version.versionId());
```

```

        s3InfoObject.setKeyName(version.key());
        return s3InfoObject;
    )))
    .peek(s3InfoObject -> {
        int i = counter.incrementAndGet(); // Increment and get the
updated value.
        if (interactive) {
            System.out.println(i + ": " + s3InfoObject.getKeyName());
            System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
            System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
        }
    })
    .collect(Collectors.toList());
}

public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
    ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
        .bucket(bucketName)
        .build();

    return getClient().listObjectVersions(versionsRequest);
}

// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .mfaDelete(MFADelete.DISABLED)
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention retention = DefaultRetention.builder()
        .days(1)

```

```
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(rention)
        .build();

    ObjectLockConfiguration objectLockConfiguration =
    ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .rule(lockRule)
        .build();

    PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
    PutObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .objectLockConfiguration(objectLockConfiguration)
        .build();

    getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
    System.out.println("Added a default retention to bucket "+bucketName
    +".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
            VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
            PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
            PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
```

```
        .objectLockConfiguration(ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .build())
        .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on
"+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
        }
    }

    public void uploadFile(String bucketName, String objectName, String filePath)
    {
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectName)
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)
            .build();

        PutObjectResponse response = getClient().putObject(request, file);
        if (response != null) {
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
        } else {
            System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
        }
    }

    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
```

```
        .status(ObjectLockLegalHoldStatus.OFF)
        .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
}

// Delete an object from a specific bucket.
public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
    try {
        DeleteObjectRequest objectRequest;
        if (hasRetention) {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
                .bypassGovernanceRetention(true)
                .build();
        } else {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
                .build();
        }

        getClient().deleteObject(objectRequest) ;
        System.out.println("The object was successfully deleted");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```



```
// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("Object retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

public void deleteBucketByName(String bucketName) {
    try {
        DeleteBucketRequest request = DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        getClient().deleteBucket(request);
        System.out.println(bucketName +" was deleted.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .build();
```

```
GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
System.out.println("Bucket object lock config for "+bucketName+": ");
System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().objectLockEnabled());
System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [GetObjectLegalHold](#)
  - [GetObjectLockConfiguration](#)
  - [GetObjectRetention](#)
  - [PutObjectLegalHold](#)
  - [PutObjectLockConfiguration](#)
  - [PutObjectRetention](#)

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

シナリオのエントリーポイント (index.js)。これにより、すべてのステップがオーケストレーションされます。GitHub にアクセスして、Scenario、ScenarioInput、ScenarioOutput、ScenarioAction の実装の詳細を確認します。

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
```

```
    loadState,
    saveState,
  } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "../welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
  updateRetention,
  updateRetentionAction,
} from "../setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Object Locking - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
```

```
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
    ],
    initialState,
),
demo: new scenarios.Scenario(
    "S3 Object Locking - Demo",
    [loadState, replAction(scenarios, client)],
    initialState,
),
clean: new scenarios.Scenario(
    "S3 Object Locking - Destroy",
    [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
    ],
    initialState,
),
};
```

```

};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
  });
}

```

ウェルカムメッセージをコンソールに出力します (welcome.steps.js)。

```

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "Welcome to the Amazon Simple Storage Service (S3) Object Locking Feature Scenario. For this workflow, we will use the AWS SDK for JavaScript to create several S3 buckets and files to demonstrate working with S3 locking features.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */

```

```
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

バケット、オブジェクト、ファイル設定をデプロイします (setup.steps.js)。

```
import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
  GetBucketVersioningCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
```

```
*/
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );

/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
      ${state.bucketPrefix}-no-lock with object lock False.
      ${state.bucketPrefix}-lock-enabled with object lock True.
      ${state.bucketPrefix}-retention-after-creation with object lock False.` ,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const noLockBucketName = `${state.bucketPrefix}-no-lock`;
    const lockEnabledBucketName = `${state.bucketPrefix}-lock-enabled`;
    const retentionBucketName = `${state.bucketPrefix}-retention-after-creation`;

    try {
      await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
      await waitUntilBucketExists({ client }, { Bucket: noLockBucketName });
      await client.send(
        new CreateBucketCommand({
```

```

        Bucket: lockEnabledBucketName,
        ObjectLockEnabledForBucket: true,
    )),
    );
    await waitUntilBucketExists(
        { client },
        { Bucket: lockEnabledBucketName },
    );
    await client.send(
        new CreateBucketCommand({ Bucket: retentionBucketName }),
    );
    await waitUntilBucketExists({ client }, { Bucket: retentionBucketName });

    state.noLockBucketName = noLockBucketName;
    state.lockEnabledBucketName = lockEnabledBucketName;
    state.retentionBucketName = retentionBucketName;
} catch (caught) {
    if (
        caught instanceof BucketAlreadyExists ||
        caught instanceof BucketAlreadyOwnedByYou
    ) {
        console.error(`${caught.name}: ${caught.message}`);
        state.earlyExit = true;
    } else {
        throw caught;
    }
}
});

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
    new scenarios.ScenarioOutput(
        "populateBuckets",
        (state) => `The following test files will be created:
            file0.txt in ${state.bucketPrefix}-no-lock.
            file1.txt in ${state.bucketPrefix}-no-lock.
            file0.txt in ${state.bucketPrefix}-lock-enabled.
            file1.txt in ${state.bucketPrefix}-lock-enabled.
            file0.txt in ${state.bucketPrefix}-retention-after-creation.
            file1.txt in ${state.bucketPrefix}-retention-after-creation.`
        ,
        { preformatted: true },
    );

```



```
/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
```

```

        Bucket: state.lockEnabledBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
    )),
);
await client.send(
    new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
    )),
);
await client.send(
    new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
    )),
);
} catch (caught) {
    if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while uploading object.  ${caught.name}:
${caught.message}`,
        );
    } else {
        throw caught;
    }
}
});

/**
 * @param {Scenarios} scenarios
 */
const updateRetention = (scenarios) =>
    new scenarios.ScenarioOutput(
        "updateRetention",
        (state) => `A bucket can be configured to use object locking with a default
retention period.
A default retention period will be configured for ${state.bucketPrefix}-
retention-after-creation.`
    );

```

```
        { preformatted: true },
    );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmUpdateRetention",
        "Configure default retention period?",
        { type: "confirm" },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) => {
    new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
        await client.send(
            new PutBucketVersioningCommand({
                Bucket: state.retentionBucketName,
                VersioningConfiguration: {
                    MFADelete: MFADeleteStatus.Disabled,
                    Status: BucketVersioningStatus.Enabled,
                },
            }),
        );

        const getBucketVersioning = new GetBucketVersioningCommand({
            Bucket: state.retentionBucketName,
        });

        await retry({ intervalInMs: 500, maxRetries: 10 }, async () => {
            const { Status } = await client.send(getBucketVersioning);
            if (Status !== "Enabled") {
                throw new Error("Bucket versioning is not enabled.");
            }
        });
    });

    await client.send(
        new PutObjectLockConfigurationCommand({
            Bucket: state.retentionBucketName,
            ObjectLockConfiguration: {
```

```

        ObjectLockEnabled: "Enabled",
        Rule: {
            DefaultRetention: {
                Mode: "GOVERNANCE",
                Years: 1,
            },
        },
    },
    }),
);
});

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
    new scenarios.ScenarioOutput(
        "updateLockPolicy",
        (state) => `Object lock policies can also be added to existing buckets.
An object lock policy will be added to ${state.bucketPrefix}-lock-enabled.` ,
        { preformatted: true },
    );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateLockPolicy = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmUpdateLockPolicy",
        "Add object lock policy?",
        { type: "confirm" },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateLockPolicyAction = (scenarios, client) =>
    new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
        await client.send(
            new PutObjectLockConfigurationCommand({
                Bucket: state.lockEnabledBucketName,
                ObjectLockConfiguration: {
                    ObjectLockEnabled: "Enabled",

```

```
    },
  )),
);
});

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileEnabled",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
    ${state.lockEnabledBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileEnabledAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        })),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
  );
```

```

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
      ${state.lockEnabledBucketName}?
      Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
      able to delete this file or its bucket until the retention period has expired.`
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
        })
      );
      console.log(
        `Set retention for file1.txt in ${state.lockEnabledBucketName} until
        ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
  );

```

```
/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileRetention",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
${state.retentionBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileRetentionAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
```

```

    new scenarios.ScenarioInput(
      "confirmSetRetentionPeriodFileRetention",
      (state) =>
        `Would you like to add a 1 day Governance retention period to file1.txt in
        ${state.retentionBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
able to delete this file or its bucket until the retention period has expired.`
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.retentionBucketName} until
        ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
  );

export {
  getBucketPrefix,
  createBuckets,

```



```
confirmCreateBuckets,  
createBucketsAction,  
populateBuckets,  
confirmPopulateBuckets,  
populateBucketsAction,  
updateRetention,  
confirmUpdateRetention,  
updateRetentionAction,  
updateLockPolicy,  
confirmUpdateLockPolicy,  
updateLockPolicyAction,  
confirmSetLegalHoldFileEnabled,  
setLegalHoldFileEnabledAction,  
confirmSetRetentionPeriodFileEnabled,  
setRetentionPeriodFileEnabledAction,  
confirmSetLegalHoldFileRetention,  
setLegalHoldFileRetentionAction,  
confirmSetRetentionPeriodFileRetention,  
setRetentionPeriodFileRetentionAction,  
};
```

バケット内のファイルを表示および削除します (repl.steps.js)。

```
import {  
  ChecksumAlgorithm,  
  DeleteObjectCommand,  
  GetObjectLegalHoldCommand,  
  GetObjectLockConfigurationCommand,  
  GetObjectRetentionCommand,  
  ListObjectVersionsCommand,  
  PutObjectCommand,  
} from "@aws-sdk/client-s3";  
  
/**  
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios  
 */  
  
/**  
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client  
 */  
  
const choices = {
```

```
EXIT: 0,
LIST_ALL_FILES: 1,
DELETE_FILE: 2,
DELETE_FILE_WITH_RETENTION: 3,
OVERWRITE_FILE: 4,
VIEW_RETENTION_SETTINGS: 5,
VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    "Explore the S3 locking features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
          name: "View the object and bucket retention settings for a file.",
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
```

```

/** @type {{bucket: string, key: string, version: string}[]} */
const files = [];
for (const bucket of buckets) {
  const objectsResponse = await client.send(
    new ListObjectVersionsCommand({ Bucket: bucket }),
  );
  for (const version of objectsResponse.Versions || []) {
    const { Key, VersionId } = version;
    files.push({ bucket, key: Key, version: VersionId });
  }
}

return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.noLockBucketName,
        state.lockEnabledBucketName,
        state.retentionBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
              file.version
            })`,
            value: index,
          })),
        },
      );

      const { replChoice } = state;

```

```
switch (replChoice) {
  case choices.LIST_ALL_FILES: {
    const files = await getAllFiles(client, [
      state.noLockBucketName,
      state.lockEnabledBucketName,
      state.retentionBucketName,
    ]);
    state.replOutput = files
      .map(
        (file) =>
          `${file.bucket}: ${file.key} (version: ${file.version})`,
      )
      .join("\n");
    break;
  }
  case choices.DELETE_FILE: {
    /** @type {number} */
    const fileToDelete = await fileInput.handle(state);
    const selectedFile = files[fileToDelete];
    try {
      await client.send(
        new DeleteObjectCommand({
          Bucket: selectedFile.bucket,
          Key: selectedFile.key,
          VersionId: selectedFile.version,
        }),
      );
      state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
    } catch (err) {
      state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
    }
    break;
  }
  case choices.DELETE_FILE_WITH_RETENTION: {
    /** @type {number} */
    const fileToDelete = await fileInput.handle(state);
    const selectedFile = files[fileToDelete];
    try {
      await client.send(
        new DeleteObjectCommand({
          Bucket: selectedFile.bucket,
```

```
        Key: selectedFile.key,
        VersionId: selectedFile.version,
        BypassGovernanceRetention: true,
      )),
    );
    state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
  } catch (err) {
    state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
  }
  break;
}
case choices.OVERWRITE_FILE: {
  /** @type {number} */
  const fileToOverwrite = await fileInput.handle(state);
  const selectedFile = files[fileToOverwrite];
  try {
    await client.send(
      new PutObjectCommand({
        Bucket: selectedFile.bucket,
        Key: selectedFile.key,
        Body: "New content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      )),
    );
    state.replOutput = `Overwrote ${selectedFile.key} in
${selectedFile.bucket}.`;
  } catch (err) {
    state.replOutput = `Unable to overwrite object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
  }
  break;
}
case choices.VIEW_RETENTION_SETTINGS: {
  /** @type {number} */
  const fileToView = await fileInput.handle(state);
  const selectedFile = files[fileToView];
  try {
    const retention = await client.send(
      new GetObjectRetentionCommand({
        Bucket: selectedFile.bucket,
        Key: selectedFile.key,
        VersionId: selectedFile.version,
```

```

    }},
  );
  const bucketConfig = await client.send(
    new GetObjectLockConfigurationCommand({
      Bucket: selectedFile.bucket,
    }),
  );
  state.replOutput = `Object retention for ${selectedFile.key}
in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
  } catch (err) {
    state.replOutput = `Unable to fetch object lock retention:
'${err.message}'`;
  }
  break;
}
case choices.VIEW_LEGAL_HOLD_SETTINGS: {
  /** @type {number} */
  const fileToView = await fileInput.handle(state);
  const selectedFile = files[fileToView];
  try {
    const legalHold = await client.send(
      new GetObjectLegalHoldCommand({
        Bucket: selectedFile.bucket,
        Key: selectedFile.key,
        VersionId: selectedFile.version,
      }),
    );
    state.replOutput = `Object legal hold for ${selectedFile.key} in
${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
  } catch (err) {
    state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
  }
  break;
}
default:
  throw new Error(`Invalid replChoice: ${replChoice}`);
}
},
{

```

```

    whileConfig: {
      whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
      input: replInput(scenarios),
      output: new scenarios.ScenarioOutput(
        "REPL output",
        (state) => state.replOutput,
        { preformatted: true },
      ),
    },
  },
},
);

export { replInput, replAction, choices };

```

作成されたすべてのリソースを破棄します (clean.steps.js)。

```

import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
  GetObjectLegalHoldCommand,
  GetObjectRetentionCommand,
  PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**

```

```
* @param {Scenarios} scenarios
* @param {S3Client} client
*/
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;

    const buckets = [
      noLockBucketName,
      lockEnabledBucketName,
      retentionBucketName,
    ];

    for (const bucket of buckets) {
      /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
      let objectsResponse;

      try {
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Object's bucket has already been deleted.");
          continue;
        }
        throw e;
      }

      for (const version of objectsResponse.Versions || []) {
        const { Key, VersionId } = version;

        try {
          const legalHold = await client.send(
            new GetObjectLegalHoldCommand({
              Bucket: bucket,
              Key,
              VersionId,
            }),
          );
        }
      }
    }
  });
```



```
    if (legalHold.LegalHold?.Status === "ON") {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: bucket,
          Key,
          VersionId,
          LegalHold: {
            Status: "OFF",
          },
        }),
      );
    }
  } catch (err) {
    console.log(
      `Unable to fetch legal hold for ${Key} in ${bucket}:` +
      `${err.message}`,
    );
  }

  try {
    const retention = await client.send(
      new GetObjectRetentionCommand({
        Bucket: bucket,
        Key,
        VersionId,
      }),
    );

    if (retention.Retention?.Mode === "GOVERNANCE") {
      await client.send(
        new DeleteObjectCommand({
          Bucket: bucket,
          Key,
          VersionId,
          BypassGovernanceRetention: true,
        }),
      );
    }
  } catch (err) {
    console.log(
      `Unable to fetch object lock retention for ${Key} in ${bucket}:` +
      `${err.message}`,
    );
  }
}
```

```
        await client.send(
            new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
            }),
        );
    }

    await client.send(new DeleteBucketCommand({ Bucket: bucket }));
    console.log(`Delete for ${bucket} complete.`);
}

export { confirmCleanup, cleanupAction };
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の以下のトピックを参照してください。
  - [GetObjectLegalHold](#)
  - [GetObjectLockConfiguration](#)
  - [GetObjectRetention](#)
  - [PutObjectLegalHold](#)
  - [PutObjectLockConfiguration](#)
  - [PutObjectRetention](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 の条件付きリクエストを行う

次のコード例は、Amazon S3 リクエストに前提条件を追加する方法を示しています。

## .NET

### SDK for .NET

#### Note

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

Amazon S3 の条件付きリクエスト機能を示すインタラクティブなシナリオを実行します。

```
using Amazon.S3;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ConditionalRequestsScenario;

public static class S3ConditionalRequestsScenario
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

        This example demonstrates the use of conditional requests for S3 operations.
        You can use conditional requests to add preconditions to S3 read requests to
        return or copy
        an object based on its Entity tag (ETag), or last modified date.
        You can use a conditional write requests to prevent overwrites by ensuring
        there is no existing object with the same key.
    */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
    public static string _resourcePrefix = null!;
    public static string _sourceBucketName = null!;
    public static string _destinationBucketName = null!;
    public static string _sampleObjectKey = null!;
```

```
public static string _sampleObjectEtag = null!;  
public static bool _interactive = true;  
  
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<IAmazonS3>()  
                .AddTransient<S3ActionsWrapper>()  
        )  
        .Build();  
  
    _configuration = new ConfigurationBuilder()  
        .SetBasePath(Directory.GetCurrentDirectory())  
        .AddJsonFile("settings.json") // Load settings from .json file.  
        .AddJsonFile("settings.local.json",  
            true) // Optionally, load local settings.  
        .Build();  
  
    ServicesSetup(host);  
  
    try  
    {  
        Console.WriteLine(new string('-', 80));  
        Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)  
Conditional Requests Feature Scenario.");  
        Console.WriteLine(new string('-', 80));  
        ConfigurationSetup();  
        _sampleObjectEtag = await Setup(_sourceBucketName,  
_destinationBucketName, _sampleObjectKey);  
  
        await DisplayDemoChoices(_sourceBucketName, _destinationBucketName,  
_sampleObjectKey, _sampleObjectEtag, 0);  
  
        Console.WriteLine(new string('-', 80));  
        Console.WriteLine("Cleaning up resources.");  
    }  
}
```

```

        Console.WriteLine(new string('-', 80));
        await Cleanup(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Amazon S3 Conditional Requests Feature Scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"There was a problem: {ex.Message}");
        await CleanupScenario(_sourceBucketName, _destinationBucketName);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
}

/// <summary>
/// Any setup operations needed.
/// </summary>
public static void ConfigurationSetup()
{
    _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

    _sourceBucketName = _resourcePrefix + "-source";
    _destinationBucketName = _resourcePrefix + "-dest";
    _sampleObjectKey = _resourcePrefix + "-sample-object.txt";
}

/// <summary>
/// Sets up the scenario by creating a source and destination bucket, and
uploading a test file to the source bucket.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>

```

```

    /// <param name="objectKey">The name of the test file to add to the source
    bucket.</param>
    /// <returns>The ETag of the uploaded test file.</returns>
    public static async Task<string> Setup(string sourceBucket, string
    destBucket, string objectKey)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create
    several S3\n" +
            "buckets and files to demonstrate working with S3 conditional
    requests.\n" +
            "This example demonstrates the use of conditional requests for S3
    operations.\r\n" +
            "You can use conditional requests to add preconditions to S3 read
    requests to return or copy\r\n" +
            "an object based on its Entity tag (ETag), or last modified date. \r
    \n" +
            "You can use a conditional write requests to prevent overwrites by
    ensuring \r\n" +
            "there is no existing object with the same key. \r\n\r\n" +
            "This example will allow you to perform conditional reads\r\n" +
            "and writes that will succeed or fail based on your selected options.
    \r\n\r\n" +
            "Sample buckets and a sample object will be created as part of the
    example.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (_interactive)
            Console.ReadLine();

        await _s3ActionsWrapper.CreateBucketWithName(sourceBucket);
        await _s3ActionsWrapper.CreateBucketWithName(destBucket);

        var eTag = await _s3ActionsWrapper.PutObjectConditional(objectKey,
        sourceBucket,
            "Test file content.");

        return eTag;
    }

    /// <summary>
    /// Cleans up the scenario by deleting the source and destination buckets.
    /// </summary>

```

```
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
public static async Task CleanupScenario(string sourceBucket, string
destBucket)
{
    await _s3ActionsWrapper.CleanupBucketByName(sourceBucket);
    await _s3ActionsWrapper.CleanupBucketByName(destBucket);
}

/// <summary>
/// Displays a list of the objects in the test buckets.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
public static async Task DisplayBuckets(string sourceBucket, string
destBucket)
{
    await _s3ActionsWrapper.ListBucketContentsByName(sourceBucket);
    await _s3ActionsWrapper.ListBucketContentsByName(destBucket);
}

/// <summary>
/// Displays the menu of conditional request options for the user.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
/// <param name="objectKey">The key of the test object in the source
bucket.</param>
/// <param name="etag">The ETag of the test object in the source bucket.</
param>
public static async Task DisplayDemoChoices(string sourceBucket, string
destBucket, string objectKey, string etag, int defaultChoice)
{
    var actions = new[]
    {
        "Print a list of bucket items.",
        "Perform a conditional read.",
        "Perform a conditional copy.",
        "Perform a conditional write.",
        "Clean up and exit."
    };

    var conditions = new[]
    {
```

```

        "If-Match: using the object's ETag. This condition should succeed.",
        "If-None-Match: using the object's ETag. This condition should
fail.",
        "If-Modified-Since: using yesterday's date. This condition should
succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
fail."
    };

    var conditionTypes = new[]
    {
        S3ConditionType.IfMatch,
        S3ConditionType.IfNoneMatch,
        S3ConditionType.IfModifiedSince,
        S3ConditionType.IfUnmodifiedSince,
    };

    var yesterdayDate = DateTime.UtcNow.AddDays(-1);

    int choice;
    while ((choice = GetChoiceResponse("\nExplore the S3 conditional request
features by selecting one of the following choices:", actions, defaultChoice)) != 4)
    {
        switch (choice)
        {
            case 0:
                Console.WriteLine("Listing the objects and buckets.");
                await DisplayBuckets(sourceBucket, destBucket);
                break;
            case 1:
                int conditionTypeIndex = GetChoiceResponse("Perform a
conditional read:", conditions, 1);
                if (conditionTypeIndex == 0 || conditionTypeIndex == 1)
                {
                    await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], null, _sampleObjectEtag);
                }
                else if (conditionTypeIndex == 2 || conditionTypeIndex == 3)
                {
                    await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], yesterdayDate);
                }
                break;
        }
    }

```



```
        case 2:
            int copyConditionTypeIndex = GetChoiceResponse("Perform a
conditional copy:", conditions, 1);
            string destKey = GetStringResponse("Enter an object key:",
"sampleObjectKey");
            if (copyConditionTypeIndex == 0 || copyConditionTypeIndex ==
1)
            {
                await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex], null,
etag);
            }
            else if (copyConditionTypeIndex == 2 ||
copyConditionTypeIndex == 3)
            {
                await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex],
yesterdayDate);
            }
            break;
        case 3:
            Console.WriteLine("Perform a conditional write using
IfNoneMatch condition on the object key.");
            Console.WriteLine("If the key is a duplicate, the write will
fail.");
            string newObjectKey = GetStringResponse("Enter an object
key:", "newObjectKey");
            await _s3ActionsWrapper.PutObjectConditional(newObjectKey,
sourceBucket, "Conditional write example data.");
            break;
    }

    if (!_interactive)
    {
        break;
    }

    Console.WriteLine("Proceeding to cleanup.");
}

// <summary>
/// Clean up the resources from the scenario.
/// </summary>
```

```
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Cleanup(bool interactive)
{
    Console.WriteLine(new string('-', 80));

    if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
    {
        await _s3ActionsWrapper.CleanUpBucketByName(_sourceBucketName);
        await _s3ActionsWrapper.CleanUpBucketByName(_destinationBucketName);

    }
    else
    {
        Console.WriteLine(
            "Ok, we'll leave the resources intact.\n" +
            "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
        );
    }

    Console.WriteLine(new string('-', 80));
    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>
/// Helper method to get a choice response from the user.
/// </summary>
```

```
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices, int defaultChoice)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"{i + 1}. {choices[i]}");
            }
        }

        if (!_interactive)
            return defaultChoice;

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }

    /// <summary>
    /// Get a string response from the user.
    /// </summary>
    /// <param name="question">The question to print.</param>
    /// <param name="defaultAnswer">A default answer to use when not
    interactive.</param>
    /// <returns>The string response.</returns>
    public static string GetStringResponse(string? question, string defaultAnswer)
    {
        string? answer = "";
        if (!_interactive)
        {
            do
```

```
        {
            Console.WriteLine(question);
            answer = Console.ReadLine();
        } while (string.IsNullOrEmpty(answer));
    }
    else
    {
        answer = defaultAnswer;
    }

    return answer;
}
}
```

S3 関数のラッパークラス。

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Logging;

namespace S3ConditionalRequestsScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;
    private readonly ILogger<S3ActionsWrapper> _logger;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    /// <param name="logger">The class logger.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, ILogger<S3ActionsWrapper> logger)
    {
        _amazonS3 = amazonS3;
        _logger = logger;
    }
}
```

```
/// <summary>
/// Retrieves an object from Amazon S3 with a conditional request.
/// </summary>
/// <param name="objectKey">The key of the object to retrieve.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="conditionType">The type of condition: 'IfMatch',
'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional read is successful, False otherwise.</
returns>
public async Task<bool> GetObjectConditional(string objectKey, string
sourceBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
        var getObjectRequest = new GetObjectRequest
        {
            BucketName = sourceBucket,
            Key = objectKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                getObjectRequest.EtagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                getObjectRequest.EtagToNotMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfModifiedSince:
                getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            case S3ConditionType.IfUnmodifiedSince:
                getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            default:
```

```

        throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
    }

    var response = await _amazonS3.GetObjectAsync(getObjectRequest);
    var sampleBytes = new byte[20];
    await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
    _logger.LogInformation($"Conditional read
successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
    return true;
}
catch (AmazonS3Exception e)
{
    if (e.ErrorCode == "PreconditionFailed")
    {
        _logger.LogError("Conditional read failed: Precondition failed");
    }
    else if (e.ErrorCode == "NotModified")
    {
        _logger.LogError("Conditional read failed: Object not modified");
    }
    else
    {
        _logger.LogError($"Unexpected error: {e.ErrorCode}");
        throw;
    }
    return false;
}
}

/// <summary>
/// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
/// </summary>
/// <param name="objectKey">The key of the object to upload.</param>
/// <param name="bucket">The source bucket of the object.</param>
/// <param name="content">The content to upload as a string.</param>
/// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
{
    try

```

```

    {
        var putObjectRequest = new PutObjectRequest
        {
            BucketName = bucket,
            Key = objectKey,
            ContentBody = content,
            IfNoneMatch = "*"
        };

        var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
        _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
        return putResult.ETag;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional write failed: Precondition
failed");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return string.Empty;
    }
}

/// <summary>
/// Copies an object from one Amazon S3 bucket to another with a conditional
request.
/// </summary>
/// <param name="sourceKey">The key of the source object to copy.</param>
/// <param name="destKey">The key of the destination object.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="destBucket">The destination bucket of the object.</param>
/// <param name="conditionType">The type of condition to apply, e.g.
'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>

```

```
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional copy is successful, False otherwise.</
returns>
public async Task<bool> CopyObjectConditional(string sourceKey, string
destKey, string sourceBucket, string destBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
        var copyObjectRequest = new CopyObjectRequest
        {
            DestinationBucket = destBucket,
            DestinationKey = destKey,
            SourceBucket = sourceBucket,
            SourceKey = sourceKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                copyObjectRequest.ETagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfModifiedSince:
                copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            case S3ConditionType.IfUnmodifiedSince:
                copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            default:
                throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
        }

        await _amazonS3.CopyObjectAsync(copyObjectRequest);
        _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
        return true;
    }
}
```



```
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional copy failed: Precondition failed");
        }
        else if (e.ErrorCode == "304")
        {
            _logger.LogError("Conditional copy failed: Object not modified");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

/// <summary>
/// Create a new Amazon S3 bucket with a specified name and check that the
bucket is ready.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateBucketWithName(string bucketName)
{
    Console.WriteLine($"\\tCreating bucket {bucketName}.");
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true
        };

        await _amazonS3.PutBucketAsync(request);
        var bucketReady = false;
        var retries = 5;
        while (!bucketReady && retries > 0)
        {
            Thread.Sleep(5000);
        }
    }
}
```

```
        bucketReady = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_amazonS3, bucketName);
        retries--;
    }

    return bucketReady;
}
catch (BucketAlreadyExistsException ex)
{
    Console.WriteLine($"Bucket already exists: '{ex.Message}'");
    return true;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error creating bucket: '{ex.Message}'");
    return false;
}
}

/// <summary>
/// Cleans up objects and deletes the bucket by name.
/// </summary>
/// <param name="bucketName">The name of the bucket.</param>
/// <returns>Async task.</returns>
public async Task CleanupBucketByName(string bucketName)
{
    try
    {
        var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
        foreach (var obj in listObjectsResponse.S3Objects)
        {
            await _amazonS3.DeleteObjectAsync(new DeleteObjectRequest
{ BucketName = bucketName, Key = obj.Key });
        }
        await _amazonS3.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        Console.WriteLine($"Cleaned up bucket: {bucketName}.");
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "NoSuchBucket")
        {

```

```
        Console.WriteLine($"Bucket {bucketName} does not exist, skipping
cleanup.");
    }
    else
    {
        Console.WriteLine($"Error deleting bucket: {e.ErrorCode}");
        throw;
    }
}

}

/// <summary>
/// List the contents of the bucket with their ETag.
/// </summary>
/// <param name="bucketName">The name of the bucket.</param>
/// <returns>Async task.</returns>
public async Task<List<S3Object>> ListBucketContentsByName(string bucketName)
{
    var results = new List<S3Object>();
    try
    {
        Console.WriteLine($"\\t Items in bucket {bucketName}");
        var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
        if (listObjectsResponse.S3Objects.Count == 0)
        {
            Console.WriteLine($"\\t\\tNo objects found.");
        }
        else
        {
            foreach (var obj in listObjectsResponse.S3Objects)
            {
                Console.WriteLine($"\\t\\t object: {obj.Key} ETag {obj.ETag}");
            }
        }
        results = listObjectsResponse.S3Objects;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "NoSuchBucket")
        {
            _logger.LogError($"Bucket {bucketName} does not exist.");
        }
    }
}
```

```

        else
        {
            _logger.LogError($"Error listing bucket and objects:
{e.ErrorCode}");
            throw;
        }
    }

    return results;
}

/// <summary>
/// Delete an object from a specific bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey)
{
    try
    {
        var request = new DeleteObjectRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };
        await _amazonS3.DeleteObjectAsync(request);
        Console.WriteLine($"Deleted {objectKey} in {bucketName}.");
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
        return false;
    }
}

/// <summary>
/// Delete a specific bucket by deleting the objects and then the bucket
itself.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>

```

```
/// <param name="objectKey">The key of the object to delete.</param>
/// <param name="versionId">Optional versionId.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CleanUpBucketByName(string bucketName)
{
    try
    {
        var allFiles = await ListBucketContentsByName(bucketName);

        foreach (var fileInfo in allFiles)
        {
            await DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key);
        }

        var request = new DeleteBucketRequest() { BucketName = bucketName, };
        var response = await _amazonS3.DeleteBucketAsync(request);
        Console.WriteLine($"\\tDelete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
            ex.Message);
        return false;
    }
}
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [GetObject](#)
  - [PutObject](#)

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

ワークフローのエントリーポイント (index.js)。これにより、すべてのステップがオーケストレーションされます。GitHub にアクセスして、Scenario、ScenarioInput、ScenarioOutput、ScenarioAction の実装の詳細を確認します。

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "../welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
} from "../setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
```

```
"S3 Conditional Requests - Deploy",
[
  welcome(scenarios),
  welcomeContinue(scenarios),
  exitOnFalse(scenarios, "welcomeContinue"),
  getBucketPrefix(scenarios),
  createBuckets(scenarios),
  confirmCreateBuckets(scenarios),
  exitOnFalse(scenarios, "confirmCreateBuckets"),
  createBucketsAction(scenarios, client),
  populateBuckets(scenarios),
  confirmPopulateBuckets(scenarios),
  exitOnFalse(scenarios, "confirmPopulateBuckets"),
  populateBucketsAction(scenarios, client),
  saveState,
],
initialState,
),
demo: new scenarios.Scenario(
  "S3 Conditional Requests - Demo",
  [loadState, welcome(scenarios), replAction(scenarios, client)],
  initialState,
),
clean: new scenarios.Scenario(
  "S3 Conditional Requests - Destroy",
  [
    loadState,
    confirmCleanup(scenarios),
    exitOnFalse(scenarios, "confirmCleanup"),
    cleanupAction(scenarios, client),
  ],
  initialState,
),
};
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
```

```

Scenarios.parseScenarioArgs(objectLockingScenarios, {
  name: "Amazon S3 object locking workflow",
  description:
    "Work with Amazon Simple Storage Service (Amazon S3) object locking
    features.",
  synopsis:
    "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
  });
}

```

ウェルカムメッセージをコンソールに出力します (welcome.steps.js)。

```

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "This example demonstrates the use of conditional requests for S3
    operations." +
    " You can use conditional requests to add preconditions to S3 read requests
    to return " +
    "or copy an object based on its Entity tag (ETag), or last modified
    date.You can use " +
    "a conditional write requests to prevent overwrites by ensuring there is no
    existing " +
    "object with the same key.\n" +
    "This example will enable you to perform conditional reads and writes that
    will succeed " +
    "or fail based on your selected options.\n" +
    "Sample buckets and a sample object will be created as part of the example.
    \n" +
    "Some steps require a key name prefix to be defined by the user. Before you
    begin, you can " +
    "optionally edit this prefix in ./object_name.json. If you do so, please
    reload the scenario before you begin.",
    { header: true },
  )

```



```

    );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };

```

バケットとオブジェクトをデプロイします (setup.steps.js)。

```

import {
  ChecksumAlgorithm,
  CreateBucketCommand,
  PutObjectCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );

```

```
);  
/**  
 * @param {Scenarios} scenarios  
 */  
const createBuckets = (scenarios) =>  
  new scenarios.ScenarioOutput(  
    "createBuckets",  
    (state) => `The following buckets will be created:  
      ${state.bucketPrefix}-source-bucket.  
      ${state.bucketPrefix}-destination-bucket.`,  
    { preformatted: true },  
  );  
  
/**  
 * @param {Scenarios} scenarios  
 */  
const confirmCreateBuckets = (scenarios) =>  
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {  
    type: "confirm",  
  });  
  
/**  
 * @param {Scenarios} scenarios  
 * @param {S3Client} client  
 */  
const createBucketsAction = (scenarios, client) =>  
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {  
    const sourceBucketName = `${state.bucketPrefix}-source-bucket`;  
    const destinationBucketName = `${state.bucketPrefix}-destination-bucket`;  
  
    try {  
      await client.send(  
        new CreateBucketCommand({  
          Bucket: sourceBucketName,  
        } ),  
      );  
      await waitUntilBucketExists({ client }, { Bucket: sourceBucketName });  
      await client.send(  
        new CreateBucketCommand({  
          Bucket: destinationBucketName,  
        } ),  
      );  
      await waitUntilBucketExists(  
        { client },
```

```

        { Bucket: destinationBucketName },
    );

    state.sourceBucketName = sourceBucketName;
    state.destinationBucketName = destinationBucketName;
} catch (caught) {
    if (
        caught instanceof BucketAlreadyExists ||
        caught instanceof BucketAlreadyOwnedByYou
    ) {
        console.error(`${caught.name}: ${caught.message}`);
        state.earlyExit = true;
    } else {
        throw caught;
    }
}
});

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
    new scenarios.ScenarioOutput(
        "populateBuckets",
        (state) => `The following test files will be created:
            file01.txt in ${state.bucketPrefix}-source-bucket.`,
        { preformatted: true },
    );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmPopulateBuckets",
        "Populate the buckets?",
        { type: "confirm" },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>

```

```
new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
  try {
    await client.send(
      new PutObjectCommand({
        Bucket: state.sourceBucketName,
        Key: "file01.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
  } catch (caught) {
    if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while uploading object.  ${caught.name}:
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
});

export {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
};
```

S3 条件付きリクエストを使用してオブジェクトを取得、コピー、配置します (repl.steps.js)。

```
import path from "node:path";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

import {
  ListObjectVersionsCommand,
  GetObjectCommand,
```

```
CopyObjectCommand,
PutObjectCommand,
} from "@aws-sdk/client-s3";
import data from "../object_name.json" assert { type: "json" };
import { readFile } from "node:fs/promises";
import {
  ScenarioInput,
  Scenario,
  ScenarioAction,
  ScenarioOutput,
} from "../../libs/scenario/index.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  CONDITIONAL_READ: 2,
  CONDITIONAL_COPY: 3,
  CONDITIONAL_WRITE: 4,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new ScenarioInput(
    "replChoice",
    "Explore the S3 conditional request features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "Print list of bucket items.", value: choices.LIST_ALL_FILES },
        {
          name: "Perform a conditional read.",
          value: choices.CONDITIONAL_READ,
        },
      ],
    },
  ),
```

```

        {
            name: "Perform a conditional copy. These examples use the key name
prefix defined in ./object_name.json.",
            value: choices.CONDITIONAL_COPY,
        },
        {
            name: "Perform a conditional write. This example use the sample file ./
text02.txt.",
            value: choices.CONDITIONAL_WRITE,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
    ],
},
);

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
    /** @type {{bucket: string, key: string, version: string}[]} */
    const files = [];
    for (const bucket of buckets) {
        const objectsResponse = await client.send(
            new ListObjectVersionsCommand({ Bucket: bucket })),
        );
        for (const version of objectsResponse.Versions || []) {
            const { Key } = version;
            files.push({ bucket, key: Key });
        }
    }
    return files;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 * @param {string} key
 */
const getEtag = async (client, bucket, key) => {
    const objectsResponse = await client.send(
        new GetObjectCommand({
            Bucket: bucket,
            Key: key,

```

```

    }},
  );
  return objectsResponse.ETag;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
export const replAction = (scenarios, client) =>
  new ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.sourceBucketName,
        state.destinationBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file to use:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (Etag: ${
              file.version
            })`,
            value: index,
          })),
        },
      );

      const condReadOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional read action would you like to take?",
        {
          type: "select",
          choices: [
            "If-Match: using the object's ETag. This condition should succeed.",

```

```

        "If-None-Match: using the object's ETag. This condition should
fail.",
        "If-Modified-Since: using yesterday's date. This condition should
succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
    ],
},
);
const condCopyOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional copy action would you like to take?",
    {
        type: "select",
        choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
        ],
    },
);
const condWriteOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional write action would you like to take?",
    {
        type: "select",
        choices: [
            "IfNoneMatch condition on the object key: If the key is a duplicate,
the write will fail.",
        ],
    },
);

const { replChoice } = state;

switch (replChoice) {
    case choices.LIST_ALL_FILES: {
        const files = await getAllFiles(client, [
            state.sourceBucketName,
            state.destinationBucketName,

```



```

    ]);
    state.replOutput = files
      .map(
        (file) => `Items in bucket ${file.bucket}: object: ${file.key} `,
      )
      .join("\n");
    break;
  }
  case choices.CONDITIONAL_READ:
  {
    const selectedCondRead = await condReadOptions.handle(state);
    if (
      selectedCondRead ===
      "If-Match: using the object's ETag. This condition should succeed."
    ) {
      const bucket = state.sourceBucketName;
      const key = "file01.txt";
      const ETag = await getEtag(client, bucket, key);

      try {
        await client.send(
          new GetObjectCommand({
            Bucket: bucket,
            Key: key,
            IfMatch: ETag,
          }),
        );
        state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because ETag provided matches the object's ETag.`;
      } catch (err) {
        state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
      }
      break;
    }
    if (
      selectedCondRead ===
      "If-None-Match: using the object's ETag. This condition should
fail."
    ) {
      const bucket = state.sourceBucketName;
      const key = "file01.txt";
      const ETag = await getEtag(client, bucket, key);

```

```
        try {
            await client.send(
                new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfNoneMatch: ETag,
                }),
            );
            state.replOutput = `${key} in ${state.sourceBucketName} was
returned.`;
        } catch (err) {
            state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
        }
        break;
    }
    if (
        selectedCondRead ===
        "If-Modified-Since: using yesterday's date. This condition should
succeed."
    ) {
        const date = new Date();
        date.setDate(date.getDate() - 1);

        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        try {
            await client.send(
                new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfModifiedSince: date,
                }),
            );
            state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because it has been created or modified in the last 24 hours.`;
        } catch (err) {
            state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
        }
        break;
    }
    if (
        selectedCondRead ===
```

```

        "If-Unmodified-Since: using yesterday's date. This condition should
fail."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";

        const date = new Date();
        date.setDate(date.getDate() - 1);
        try {
            await client.send(
                new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfUnmodifiedSince: date,
                }),
            );
            state.replOutput = `${key} in ${state.sourceBucketName} was
read.`;
        } catch (err) {
            state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
        }
        break;
    }
    break;
case choices.CONDITIONAL_COPY: {
    const selectedCondCopy = await condCopyOptions.handle(state);
    if (
        selectedCondCopy ===
        "If-Match: using the object's ETag. This condition should succeed."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const ETag = await getEtag(client, bucket, key);

        const copySource = `${bucket}/${key}`;
        // Optionally edit the default key name prefix of the copied object
        in ./object_name.json.
        const name = data.name;
        const copiedKey = `${name}${key}`;
        try {
            await client.send(
                new CopyObjectCommand({

```

```

        CopySource: copySource,
        Bucket: state.destinationBucketName,
        Key: copiedKey,
        CopySourceIfMatch: ETag,
    )),
    );
    state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because ETag provided matches the object's
ETag.`;
    } catch (err) {
        state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName}: ${err.message}`;
    }
    break;
}
if (
    selectedCondCopy ===
    "If-None-Match: using the object's ETag. This condition should fail."
) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    const ETag = await getETag(client, bucket, key);
    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;

    try {
        await client.send(
            new CopyObjectCommand({
                CopySource: copySource,
                Bucket: state.destinationBucketName,
                Key: copiedKey,
                CopySourceIfNoneMatch: ETag,
            )),
        );
        state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName}`;
    } catch (err) {
        state.replOutput = `Unable to copy object as ${key} as as
${copiedKey} to bucket ${state.destinationBucketName}: ${err.message}`;
    }
    break;
}

```

```
    }
    if (
      selectedCondCopy ===
        "If-Modified-Since: using yesterday's date. This condition should
succeed."
    ) {
      const bucket = state.sourceBucketName;
      const key = "file01.txt";
      const copySource = `${bucket}/${key}`;
      // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
      const name = data.name;
      const copiedKey = `${name}${key}`;

      const date = new Date();
      date.setDate(date.getDate() - 1);

      try {
        await client.send(
          new CopyObjectCommand({
            CopySource: copySource,
            Bucket: state.destinationBucketName,
            Key: copiedKey,
            CopySourceIfModifiedSince: date,
          }),
        );
        state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because it has been created or modified in the
last 24 hours.`;
      } catch (err) {
        state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName} : ${err.message}`;
      }
      break;
    }
    if (
      selectedCondCopy ===
        "If-Unmodified-Since: using yesterday's date. This condition should
fail."
    ) {
      const bucket = state.sourceBucketName;
      const key = "file01.txt";
      const copySource = `${bucket}/${key}`;
```

```

        // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
        const name = data.name;
        const copiedKey = `${name}${key}`;

        const date = new Date();
        date.setDate(date.getDate() - 1);

        try {
            await client.send(
                new CopyObjectCommand({
                    CopySource: copySource,
                    Bucket: state.destinationBucketName,
                    Key: copiedKey,
                    CopySourceIfUnmodifiedSince: date,
                }),
            );
            state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName} because it has not been created or modified in
the last 24 hours.`;
        } catch (err) {
            state.replOutput = `Unable to copy object ${key} to bucket
${state.destinationBucketName}: ${err.message}`;
        }
        break;
    }
    case choices.CONDITIONAL_WRITE:
    {
        const selectedCondWrite = await condWriteOptions.handle(state);
        if (
            selectedCondWrite ===
            "IfNoneMatch condition on the object key: If the key is a
duplicate, the write will fail."
        ) {
            // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
            const key = "text02.txt";
            const __filename = fileURLToPath(import.meta.url);
            const __dirname = dirname(__filename);
            const filePath = path.join(__dirname, "text02.txt");
            try {
                await client.send(
                    new PutObjectCommand({

```

```

        Bucket: `${state.destinationBucketName}`,
        Key: `${key}`,
        Body: await readFile(filePath),
        IfNoneMatch: "*",
      )),
    );
    state.replOutput = `${key} uploaded to bucket
${state.destinationBucketName} because the key is not a duplicate.`;
  } catch (err) {
    state.replOutput = `Unable to upload object to bucket
${state.destinationBucketName}:${err.message}`;
  }
  break;
}
}
break;

default:
  throw new Error(`Invalid replChoice: ${replChoice}`);
},
{
  whileConfig: {
    whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
    input: replInput(scenarios),
    output: new ScenarioOutput("REPL output", (state) => state.replOutput, {
      preformatted: true,
    }),
  },
},
);

export { replInput, choices };

```

作成されたすべてのリソースを破棄します (clean.steps.js)。

```

import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
} from "@aws-sdk/client-s3";

```

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { sourceBucketName, destinationBucketName } = state;
    const buckets = [sourceBucketName, destinationBucketName].filter((b) => b);

    for (const bucket of buckets) {
      try {
        let objectsResponse;
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
        for (const version of objectsResponse.Versions || []) {
          const { Key, VersionId } = version;
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
              }),
            );
          } catch (err) {
```



```
        console.log(`An error occurred: ${err.message} `);
    }
}
} catch (e) {
    if (e instanceof Error && e.name === "NoSuchBucket") {
        console.log("Objects and buckets have already been deleted.");
        continue;
    }
    throw e;
}

await client.send(new DeleteBucketCommand({ Bucket: bucket }));
console.log(`Delete for ${bucket} complete.`);
}
});

export { confirmCleanup, cleanupAction };
```

- API の詳細については、「AWS SDK for JavaScript API リファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [GetObject](#)
  - [PutObject](#)

## Python

### SDK for Python (Boto3)

#### Note

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

Amazon S3 の条件付きリクエストを示すインタラクティブなシナリオを実行します。

```
"""
Purpose
```

Shows how to use AWS SDK for Python (Boto3) to get started using conditional requests for Amazon Simple Storage Service (Amazon S3).

```
"""

import logging
import random
import sys
import datetime

import boto3
from botocore.exceptions import ClientError

from s3_conditional_requests import S3ConditionalRequests

# Add relative path to include demo_tools in this code example without need for
# setup.
sys.path.append("../..")
import demo_tools.question as q # noqa

# Constants
FILE_CONTENT = "This is a test file for S3 conditional requests."
RANDOM_SUFFIX = str(random.randint(100, 999))

logger = logging.getLogger(__name__)

class ConditionalRequestsScenario:
    """Runs a scenario that shows how to use S3 Conditional Requests."""

    def __init__(self, conditional_requests, s3_client):
        """
        :param conditional_requests: An object that wraps S3 conditional request
        actions.
        :param s3_client: A Boto3 S3 client for setup and cleanup operations.
        """
        self.conditional_requests = conditional_requests
        self.s3_client = s3_client

    def setup_scenario(self, source_bucket: str, dest_bucket: str, object_key:
str):
        """
        Sets up the scenario by creating a source and destination bucket.
```

```

    Prompts the user to provide a bucket name prefix.

:param source_bucket: The name of the source bucket.
:param dest_bucket: The name of the destination bucket.
:param object_key: The name of a test file to add to the source bucket.
"""

# Create the buckets.
try:
    self.s3_client.create_bucket(Bucket=source_bucket)
    self.s3_client.create_bucket(Bucket=dest_bucket)
    print(
        f"Created source bucket: {source_bucket} and destination bucket:
{dest_bucket}"
    )
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    logger.error(f"Error creating buckets: {error_code}")
    raise

# Upload test file into the source bucket.
try:
    print(f"Uploading file {object_key} to bucket {source_bucket}")
    response = self.s3_client.put_object(
        Bucket=source_bucket, Key=object_key, Body=FILE_CONTENT
    )
    object_etag = response["ETag"]
    return object_etag

except Exception as e:
    logger.error(
        f"Failed to upload file {object_key} to bucket {source_bucket}:
{e}"
    )

def cleanup_scenario(self, source_bucket: str, dest_bucket: str):
    """
    Cleans up the scenario by deleting the source and destination buckets.

:param source_bucket: The name of the source bucket.
:param dest_bucket: The name of the destination bucket.
"""
    self.cleanup_bucket(source_bucket)

```

```
self.cleanup_bucket(dest_bucket)

def cleanup_bucket(self, bucket_name: str):
    """
    Cleans up the bucket by deleting all objects and then the bucket itself.

    :param bucket_name: The name of the bucket.
    """
    try:
        # Get list of all objects in the bucket.
        list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
        objs = list_response.get("Contents", [])
        for obj in objs:
            key = obj["Key"]
            self.s3_client.delete_object(Bucket=bucket_name, Key=key)
        self.s3_client.delete_bucket(Bucket=bucket_name)
        print(f"Cleaned up bucket: {bucket_name}.")
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "NoSuchBucket":
            logger.info(f"Bucket {bucket_name} does not exist, skipping cleanup.")
        else:
            logger.error(f"Error deleting bucket: {error_code}")
            raise

def display_buckets(self, source_bucket: str, dest_bucket: str):
    """
    Display a list of the objects in the test buckets.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    """
    self.list_bucket_contents(source_bucket)
    self.list_bucket_contents(dest_bucket)

def list_bucket_contents(self, bucket_name):
    """
    Display a list of the objects in the bucket.

    :param bucket_name: The name of the bucket.
    """
    try:
```

```

        # Get list of all objects in the bucket.
        print(f"\t Items in bucket {bucket_name}")
        list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
        objs = list_response.get("Contents", [])
        if not objs:
            print("\t\tNo objects found.")
        for obj in objs:
            key = obj["Key"]
            print(f"\t\t object: {key} ETag {obj['ETag']}")
        return objs
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "NoSuchBucket":
            logger.info(f"Bucket {bucket_name} does not exist.")
        else:
            logger.error(f"Error listing bucket and objects: {error_code}")
            raise

def display_menu(
    self, source_bucket: str, dest_bucket: str, object_key: str, etag: str
):
    """
    Displays the menu of conditional request options for the user.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    :param object_key: The key of the test object in the source bucket.
    :param etag: The etag of the test object in the source bucket.
    """

    actions = [
        "Print list of bucket items.",
        "Perform a conditional read.",
        "Perform a conditional copy.",
        "Perform a conditional write.",
        "Clean up and exit.",
    ]

    conditions = [
        "If-Match: using the object's ETag. This condition should succeed.",
        "If-None-Match: using the object's ETag. This condition should
fail.",

```

```

        "If-Modified-Since: using yesterday's date. This condition should
        succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
        fail.",
    ]

    condition_types = [
        "IfMatch",
        "IfNoneMatch",
        "IfModifiedSince",
        "IfUnmodifiedSince",
    ]
    copy_condition_types = [
        "CopySourceIfMatch",
        "CopySourceIfNoneMatch",
        "CopySourceIfModifiedSince",
        "CopySourceIfUnmodifiedSince",
    ]

    yesterday_date = datetime.datetime.utcnow() - datetime.timedelta(days=1)

    choice = 0
    while choice != 4:
        print("-" * 88)
        print("Choose an action to explore some example conditional
requests.")
        choice = q.choose("Which action would you like to take? ", actions)
        if choice == 0:
            print("Listing the objects and buckets.")
            self.display_buckets(source_bucket, dest_bucket)
        elif choice == 1:
            print("Perform a conditional read.")
            condition_type = q.choose("Enter the condition type : ",
conditions)
            if condition_type == 0 or condition_type == 1:
                self.conditional_requests.get_object_conditional(
                    object_key, source_bucket,
condition_types[condition_type], etag
                )
            elif condition_type == 2 or condition_type == 3:
                self.conditional_requests.get_object_conditional(
                    object_key,
                    source_bucket,
                    condition_types[condition_type],

```

```

        yesterday_date,
    )
    elif choice == 2:
        print("Perform a conditional copy.")
        condition_type = q.choose("Enter the condition type : ",
conditions)
        dest_key = q.ask("Enter an object key: ", q.non_empty)
        if condition_type == 0 or condition_type == 1:
            self.conditional_requests.copy_object_conditional(
                object_key,
                dest_key,
                source_bucket,
                dest_bucket,
                copy_condition_types[condition_type],
                etag,
            )
        elif condition_type == 2 or condition_type == 3:
            self.conditional_requests.copy_object_conditional(
                object_key,
                dest_key,
                copy_condition_types[condition_type],
                yesterday_date,
            )
    elif choice == 3:
        print(
            "Perform a conditional write using IfNoneMatch condition on
the object key."
        )
        print("If the key is a duplicate, the write will fail.")
        object_key = q.ask("Enter an object key: ", q.non_empty)
        self.conditional_requests.put_object_conditional(
            object_key, source_bucket, b"Conditional write example data."
        )
    elif choice == 4:
        print("Proceeding to cleanup.")

def run_scenario(self):
    """
    Runs the interactive scenario.
    """
    print("-" * 88)
    print("Welcome to the Amazon S3 conditional requests example.")
    print("-" * 88)

```

```
print(
    f"""\
```

This example demonstrates the use of conditional requests for S3 operations.

You can use conditional requests to add preconditions to S3 read requests to return or copy

an object based on its Entity tag (ETag), or last modified date.

You can use a conditional write requests to prevent overwrites by ensuring

there is no existing object with the same key.

This example will allow you to perform conditional reads and writes that will succeed or fail based on your selected options.

Sample buckets and a sample object will be created as part of the example.

```
"""
)
```

```
bucket_prefix = q.ask("Enter a bucket name prefix: ", q.non_empty)
source_bucket_name = f"{bucket_prefix}-source-{RANDOM_SUFFIX}"
dest_bucket_name = f"{bucket_prefix}-dest-{RANDOM_SUFFIX}"
object_key = "test-upload-file.txt"
```

```
try:
    etag = self.setup_scenario(source_bucket_name, dest_bucket_name,
object_key)
    self.display_menu(source_bucket_name, dest_bucket_name, object_key,
etag)
finally:
    self.cleanup_scenario(source_bucket_name, dest_bucket_name)
```

```
print("-" * 88)
print("Thanks for watching.")
print("-" * 88)
```

```
if __name__ == "__main__":
    scenario = ConditionalRequestsScenario(
        S3ConditionalRequests.from_client(), boto3.client("s3")
    )
    scenario.run_scenario()
```



条件付きリクエストオペレーションを定義するラッパークラス。

```
import boto3
import logging

from botocore.exceptions import ClientError

# Configure logging
logger = logging.getLogger(__name__)

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.

```

```

        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
        'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
    try:
        response = self.s3.get_object(
            Bucket=source_bucket,
            Key=object_key,
            **{condition_type: condition_value},
        )
        sample_bytes = response["Body"].read(20)
        print(
            f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
        )
        print(f"\t{sample_bytes}")
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional read failed: Precondition failed")
        elif error_code == "304": # Not modified error code.
            print("\tConditional read failed: Object not modified")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise

    def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
    try:
        self.s3.put_object(
            Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
        )
    print(

```

```

        f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
    )
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    if error_code == "PreconditionFailed":
        print("\tConditional write failed: Precondition failed")
    else:
        logger.error(f"Unexpected error: {error_code}")
        raise

def copy_object_conditional(
    self,
    source_key: str,
    dest_key: str,
    source_bucket: str,
    dest_bucket: str,
    condition_type: str,
    condition_value: str,
):
    """
    Copies an object from one Amazon S3 bucket to another with a conditional
    request.

    :param source_key: The key of the source object to copy.
    :param dest_key: The key of the destination object.
    :param source_bucket: The source bucket of the object.
    :param dest_bucket: The destination bucket of the object.
    :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch',
        'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
    :param condition_value: The value to use for the condition.
    """
    try:
        self.s3.copy_object(
            Bucket=dest_bucket,
            Key=dest_key,
            CopySource={"Bucket": source_bucket, "Key": source_key},
            **{condition_type: condition_value},
        )
        print(
            f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."

```

```
)
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    if error_code == "PreconditionFailed":
        print("\tConditional copy failed: Precondition failed")
    elif error_code == "304": # Not modified error code.
        print("\tConditional copy failed: Object not modified")
    else:
        logger.error(f"Unexpected error: {error_code}")
        raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [GetObject](#)
  - [PutObject](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 バケットのアクセスコントロールリスト (ACL) を管理する

次のコード例は、Amazon S3 バケットのアクセスコントロールリスト (ACL) を管理する方法を示しています。

.NET

SDK for .NET

### Note

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

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
/// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
/// access.
/// </summary>
public class ManageACLs
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket1";
        string newBucketName = "amzn-s3-demo-bucket2";
        string keyName = "sample-object.txt";
        string emailAddress = "someone@example.com";

        // If the AWS Region where your bucket is located is different from
        // the Region defined for the default user, pass the Amazon S3
bucket's
        // name to the client constructor. It should look like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
        IAmazonS3 client = new AmazonS3Client();

        await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
    }

    /// <summary>
    /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
    /// information and then adds a new ACL to one of the objects in the
    /// Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// methods to create a bucket, get an ACL, and add a different ACL to
    /// one of the objects.</param>
    /// <param name="bucketName">A string representing the original Amazon S3
    /// bucket name.</param>

```

S3

be

```

    /// <param name="newBucketName">A string representing the name of the
    /// new bucket that will be created.</param>
    /// <param name="keyName">A string representing the key name of an Amazon
    /// object for which we will change the ACL.</param>
    /// <param name="emailAddress">A string representing the email address
    /// belonging to the person to whom access to the Amazon S3 bucket will
    /// be granted.</param>
    public static async Task TestBucketObjectACLsAsync(
        IAmazonS3 client,
        string bucketName,
        string newBucketName,
        string keyName,
        string emailAddress)
    {
        try
        {
            // Create a new Amazon S3 bucket and specify canned ACL.
            var success = await CreateBucketWithCannedACLAsync(client,
newBucketName);

            // Get the ACL on a bucket.
            await GetBucketACLAsync(client, bucketName);

            // Add (replace) the ACL on an object in a bucket.
            await AddACLToExistingObjectAsync(client, bucketName, keyName,
emailAddress);
        }
        catch (AmazonS3Exception amazonS3Exception)
        {
            Console.WriteLine($"Exception: {amazonS3Exception.Message}");
        }
    }

    /// <summary>
    /// Creates a new Amazon S3 bucket with a canned ACL attached.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="newBucketName">A string representing the name of the
    /// new Amazon S3 bucket.</param>
    /// <returns>Returns a boolean value indicating success or failure.</
returns>

```

```
public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
client, string newBucketName)
{
    var request = new PutBucketRequest()
    {
        BucketName = newBucketName,
        BucketRegion = S3Region.EUWest1,

        // Add a canned ACL.
        CannedACL = S3CannedACL.LogDeliveryWrite,
    };

    var response = await client.PutBucketAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Retrieves the ACL associated with the Amazon S3 bucket name in the
/// bucketName parameter.
/// </summary>
/// <param name="client">The initialized client object used to call
/// PutBucketAsync.</param>
/// <param name="bucketName">The Amazon S3 bucket for which we want to
get the
/// ACL list.</param>
/// <returns>Returns an S3AccessControlList returned from the call to
/// GetACLAsync.</returns>
public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
client, string bucketName)
{
    GetACLResponse response = await client.GetACLAsync(new GetACLRequest
    {
        BucketName = bucketName,
    });

    return response.AccessControlList;
}

/// <summary>
/// Adds a new ACL to an existing object in the Amazon S3 bucket.
/// </summary>
```

```
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="bucketName">A string representing the name of the Amazon
S3    /// bucket containing the object to which we want to apply a new ACL.</
param>    /// <param name="keyName">A string representing the name of the object
    /// to which we want to apply the new ACL.</param>
    /// <param name="emailAddress">The email address of the person to whom
    /// we will be applying to whom access will be granted.</param>
    public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
string    string bucketName, string keyName, string emailAddress)
    {
        // Retrieve the ACL for an object.
        GetACLResponse aclResponse = await client.GetACLAsync(new
GetACLRequest    {
            BucketName = bucketName,
            Key = keyName,
        });

        S3AccessControlList acl = aclResponse.AccessControlList;

        // Retrieve the owner.
        Owner owner = acl.Owner;

        // Clear existing grants.
        acl.Grants.Clear();

        // Add a grant to reset the owner's full permission
        // (the previous clear statement removed all permissions).
        var fullControlGrant = new S3Grant
        {
            Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
        };
        acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

        // Specify email to identify grantee for granting permissions.
        var grantUsingEmail = new S3Grant
        {
            Grantee = new S3Grantee { EmailAddress = emailAddress },
            Permission = S3Permission.WRITE_ACP,
        };
    }
}
```



```
// Specify log delivery group as grantee.
var grantLogDeliveryGroup = new S3Grant
{
    Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
    Permission = S3Permission.WRITE,
};

// Create a new ACL.
var newAcl = new S3AccessControlList
{
    Grants = new List<S3Grant> { grantUsingEmail,
grantLogDeliveryGroup },
    Owner = owner,
};

// Set the new ACL. We're throwing away the response here.
_ = await client.PutACLAsync(new PutACLRequest
{
    BucketName = bucketName,
    Key = keyName,
    AccessControlList = newAcl,
});
}

}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
  - [GetBucketAcl](#)
  - [GetObjectAcl](#)
  - [PutBucketAcl](#)
  - [PutObjectAcl](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で、Amazon S3 を使用して大量の Amazon SQS メッセージを管理する

次のコード例は、Amazon SQS 拡張クライアントライブラリを使用して大規模な Amazon SQS メッセージを操作する方法を示しています。

Java

SDK for Java 2.x

### Note

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

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
```

```
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

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

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
        LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
            DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
```

```
this.queueName = "MyQueue-" + UUID.randomUUID();

// Configure the SQS extended client.
final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
    .withPayloadSupportEnabled(s3Client, s3BucketName);

this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
}

public static void main(String[] args) {
    SqsExtendedClientExample example = new SqsExtendedClientExample();
    try {
        example.setup();
        example.sendAndReceiveMessage();
    } finally {
        example.cleanup();
    }
}

/**
 * Send a large message and receive it back.
 *
 * @return The received message
 */
public Message sendAndReceiveMessage() {
    try {
        // Create a large message.
        char[] chars = new char[messageSize];
        Arrays.fill(chars, 'x');
        String largeMessage = new String(chars);

        // Send the message.
        final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody(largeMessage)
            .build();

        sqsExtendedClient.sendMessage(sendMessageRequest);
        logger.info("Sent message of size: {}", largeMessage.length());

        // Receive and return the message.
```

```
final ReceiveMessageResponse receiveMessageResponse =
    sqsExtendedClient.receiveMessage(
        ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

List<Message> messages = receiveMessageResponse.messages();
if (messages.isEmpty()) {
    throw new RuntimeException("No messages received");
}

Message message = messages.getFirst();
logger.info("\nMessage received.");
logger.info("  ID: {}", message.messageId());
logger.info("  Receipt handle: {}", message.receiptHandle());
logger.info("  Message body size: {}", message.body().length());
logger.info("  Message body (first 5 characters): {}",
    message.body().substring(0, 5));

return message;
} catch (RuntimeException e) {
    logger.error("Error during message processing: {}", e.getMessage(),
        e);
    throw e;
}
```

- 詳細については、「[AWS SDK for Java 2.x デベロッパーガイド](#)」を参照してください。
- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [CreateBucket](#)
  - [PutBucketLifecycleConfiguration](#)
  - [ReceiveMessage](#)
  - [SendMessage](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して、バージョン管理されている Amazon S3 オブジェクトを Lambda 関数でバッチで管理する

次のコード例は、バージョン管理されている S3 オブジェクトを Lambda 関数でバッチで管理する方法を示しています。

### Python

#### SDK for Python (Boto3)

AWS Lambda 関数を呼び出して処理を実行するジョブを作成することによって、Amazon Simple Storage Service (Amazon S3) のバージョン管理されているオブジェクトをバッチで操作する方法を示しています。この例では、バージョン対応のバケットを作成し、Lewis Carroll の詩「You Are Old, Father William」からスタンザをアップロードし、Amazon S3 のバッチジョブを使用して、さまざまな方法で詩をひねります。

以下ではその方法を説明しています。

- バージョン管理されたオブジェクトを操作する Lambda 関数を作成します。
- 更新するオブジェクトのマニフェストを作成します。
- Lambda 関数を呼び出してオブジェクトを更新するバッチジョブを作成します。
- Lambda 関数を削除します。
- バージョン管理されているバケットを空にして削除します。

この例は GitHub で最もよく確認できます。完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Amazon S3

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 の URI を解析する

次のコード例は、Amazon S3 の URI を解析してバケット名やオブジェクトキーなどの重要なコンポーネントを抽出する方法を示しています。

## Java

## SDK for Java 2.x

 Note

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

[S3Uri](#) クラスを使用して Amazon S3 の URI を解析します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

/**
 *
 * @param s3Client      - An S3Client through which you acquire an S3Uri
instance.
 * @param s3objectUrl - A complex URL (String) that is used to demonstrate
S3Uri
 *
capabilities.
 */
public static void parseS3UriExample(S3Client s3Client, String s3objectUrl) {
    logger.info(s3objectUrl);
    // Console output:
    // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

    // Create an S3Utilities object using the configuration of the s3Client.
    S3Utilities s3Utilities = s3Client.utilities();

    // From a String URL create a URI object to pass to the parseUri()
method.
    URI uri = URI.create(s3objectUrl);
```

```
S3Uri s3Uri = s3Utilities.parseUri(uri);

// If the URI contains no value for the Region, bucket or key, the SDK
returns
// an empty Optional.
// The SDK returns decoded URI values.

Region region = s3Uri.region().orElse(null);
log("region", region);
// Console output: 'region: us-west-1'.

String bucket = s3Uri.bucket().orElse(null);
log("bucket", bucket);
// Console output: 'bucket: myBucket'.

String key = s3Uri.key().orElse(null);
log("key", key);
// Console output: 'key: resources/doc.txt'.

Boolean isPathStyle = s3Uri.isPathStyle();
log("isPathStyle", isPathStyle);
// Console output: 'isPathStyle: true'.

// If the URI contains no query parameters, the SDK returns an empty map.
Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
log("rawQueryParameters", queryParams);
// Console output: 'rawQueryParameters: {versionId=[abc123],
partNumber=[77,
// 88]]'.

// Retrieve the first or all values for a query parameter as shown in the
// following code.
String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
log("firstMatchingRawQueryParameter-versionId", versionId);
// Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
log("firstMatchingRawQueryParameter-partNumber", partNumber);
// Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
```



```

        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters,
        must be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
        %5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
        query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
        the dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
}

```

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 オブジェクトのマルチパートコピーを実行する

次のコード例は、Amazon S3 オブジェクトのマルチパートコピーを実行する方法を示しています。

## .NET

### SDK for .NET

#### Note

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

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUApiCopyObj
{
    private const string SourceBucket = "amzn-s3-demo-bucket1";
    private const string TargetBucket = "amzn-s3-demo-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        Console.WriteLine("Copying object...");
        await MPCopyObjectAsync(s3Client);
    }

    /// <summary>
    /// This method uses the passed client object to perform a multipart
    /// copy operation.
    /// </summary>
    /// <param name="client">An Amazon S3 client object that will be used
```

```
/// to perform the copy.</param>
public static async Task MPUCopyObjectAsync(AmazonS3Client client)
{
    // Create a list to store the copy part responses.
    var copyResponses = new List<CopyPartResponse>();

    // Setup information required to initiate the multipart upload.
    var initiateRequest = new InitiateMultipartUploadRequest
    {
        BucketName = TargetBucket,
        Key = TargetObjectKey,
    };

    // Initiate the upload.
    InitiateMultipartUploadResponse initResponse =
        await client.InitiateMultipartUploadAsync(initiateRequest);

    // Save the upload ID.
    string uploadId = initResponse.UploadId;

    try
    {
        // Get the size of the object.
        var metadataRequest = new GetObjectMetadataRequest
        {
            BucketName = SourceBucket,
            Key = SourceObjectKey,
        };

        GetObjectMetadataResponse metadataResponse =
            await client.GetObjectMetadataAsync(metadataRequest);
        var objectSize = metadataResponse.ContentLength; // Length in
bytes.

        // Copy the parts.
        var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

        long bytePosition = 0;
        for (int i = 1; bytePosition < objectSize; i++)
        {
            var copyRequest = new CopyPartRequest
            {
                DestinationBucket = TargetBucket,
                DestinationKey = TargetObjectKey,
```

```
        SourceBucket = SourceBucket,
        SourceKey = SourceObjectKey,
        UploadId = uploadId,
        FirstByte = bytePosition,
        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
        PartNumber = i,
    };

    copyResponses.Add(await client.CopyPartAsync(copyRequest));

    bytePosition += partSize;
}

// Set up to complete the copy.
var completeRequest = new CompleteMultipartUploadRequest
{
    BucketName = TargetBucket,
    Key = TargetObjectKey,
    UploadId = initResponse.UploadId,
};
completeRequest.AddPartETags(copyResponses);

// Complete the copy.
CompleteMultipartUploadResponse completeUploadResponse =
    await client.CompleteMultipartUploadAsync(completeRequest);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine($"Error encountered on server.
Message: '{e.Message}' when writing an object");
}
catch (Exception e)
{
    Console.WriteLine($"Unknown encountered on server.
Message: '{e.Message}' when writing an object");
}
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の以下のトピックを参照してください。
  - [CompleteMultipartUpload](#)
  - [CreateMultipartUpload](#)
  - [GetObjectMetadata](#)
  - [UploadPartCopy](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 イベント通知を受信して処理する

次のコード例は、オブジェクト指向の方法で S3 イベント通知を操作する方法を示しています。

Java

SDK for Java 2.x

### Note

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

この例は、Amazon SQS を使用して S3 通知イベントを処理する方法を示しています。

```
/**
 * This method receives S3 event notifications by using an SqsAsyncClient.
 * After the client receives the messages it deserializes the JSON payload
 and logs them. It uses
 * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
 * the JSON payload and access the messages in an object-oriented way.
 *
 * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
 * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
 awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
 API</a>.
```

```

* <p>
* To use S3 event notification serialization/deserialization to objects, add
the following
* dependency to your Maven pom.xml file.
* <dependency>
* <groupId>software.amazon.awssdk</groupId>
* <artifactId>s3-event-notifications</artifactId>
* <version><LATEST></version>
* </dependency>
* <p>
* The S3 event notification API became available with version 2.25.11 of the
Java SDK.
* <p>
* This example shows the use of the API with AWS SQS, but it can be used to
process S3 event notifications
* in AWS SNS or AWS Lambda as well.
* <p>
* Note: The S3EventNotification class does not work with messages routed
through AWS EventBridge.
*/
static void processS3Events(String bucketName, String queueUrl, String
queueArn) {
    try {
        // Configure the bucket to send Object Created and Object Tagging
notifications to an existing SQS queue.
        s3Client.putBucketNotificationConfiguration(b -> b
            .notificationConfiguration(ncb -> ncb
                .queueConfigurations(qcb -> qcb
                    .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                        .queueArn(queueArn)))
                .bucket(bucketName)
            ).join();

        triggerS3EventNotifications(bucketName);
        // Wait for event notifications to propagate.
        Thread.sleep(Duration.ofSeconds(5).toMillis());

        boolean didReceiveMessages = true;
        while (didReceiveMessages) {
            // Display the number of messages that are available in the
queue.

            sqsClient.getQueueAttributes(b -> b
                .queueUrl(queueUrl)

```

```

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                ).thenAccept(attributeResponse ->
                    logger.info("Approximate number of messages in
the queue: {}"),
attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                .join();

// Receive the messages.
ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
    .queueUrl(queueUrl)
    ).get();
logger.info("Count of received messages: {}",
response.messages().size());
didReceiveMessages = !response.messages().isEmpty();

// Create a collection to hold the received message for deletion
// after we log the messages.
HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();

// Process each message.
response.messages().forEach(message -> {
    logger.info("Message id: {}", message.messageId());
    // Deserialize JSON message body to a S3EventNotification
object
    // to access messages in an object-oriented way.
    S3EventNotification event =
S3EventNotification.fromJson(message.body());

    // Log the S3 event notification record details.
    if (event.getRecords() != null) {
        event.getRecords().forEach(record -> {
            String eventName = record.getEventName();
            String key = record.getS3().getObject().getKey();
            logger.info(record.toString());
            logger.info("Event name is {} and key is {}",
eventName, key);
        });
    }
    // Add logged messages to collection for batch deletion.
    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
        .id(message.messageId())
        .receiptHandle(message.receiptHandle())

```

```
        .build());
    });
    // Delete messages.
    if (!messagesToDelete.isEmpty()) {

        sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
            .queueUrl(queueUrl)
            .entries(messagesToDelete)
            .build()
        ).join();
    }
    } // End of while block.
} catch (InterruptedException | ExecutionException e) {
    throw new RuntimeException(e);
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [DeleteMessageBatch](#)
  - [GetQueueAttributes](#)
  - [PutBucketNotificationConfiguration](#)
  - [ReceiveMessage](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して、EXIF とその他の画像情報を保存する

次のコード例は、以下の操作方法を示しています。

- JPG、JPEG、または PNG ファイルから EXIF 情報を取得します。
- Amazon S3 バケットにイメージファイルをアップロードします。
- Amazon Rekognition を使用して、ファイル内の 3 つの上位属性 (ラベル) を特定します。
- EXIF およびラベル情報を、リージョンの Amazon DynamoDB テーブルに追加します。



## Rust

### SDK for Rust

JPG、JPEG、または PNG ファイルから EXIF 情報を取得し、イメージファイルを Amazon S3 バケットにアップロードし、Amazon Rekognition を使用してファイル内の 3 つの上位属性 (Amazon Rekognition のラベル) を特定し、リージョンの Amazon DynamoDB テーブルに EXIF およびラベル情報を追加します。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- DynamoDB
- Amazon Rekognition
- Amazon S3

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### AWS SDK を使用して S3 イベント通知を Amazon EventBridge に送信する

次のコード例は、バケットが EventBridge に S3 イベント通知を送信し、Amazon SNS トピックと Amazon SQS キューに通知をルーティングできるようにする方法を示しています。

## Java

### SDK for Java 2.x

#### Note

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

```
/** This method configures a bucket to send events to AWS EventBridge and  
creates a rule  
* to route the S3 object created events to a topic and a queue.
```

```

*
* @param bucketName Name of existing bucket
* @param topicArn ARN of existing topic to receive S3 event notifications
* @param queueArn ARN of existing queue to receive S3 event notifications
*
* An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
*/
public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
    try {
        // Enable bucket to emit S3 Event notifications to EventBridge.
        s3Client.putBucketNotificationConfiguration(b -> b
            .bucket(bucketName)
            .notificationConfiguration(b1 -> b1
                .eventBridgeConfiguration(
                    SdkBuilder::build)
            ).build()).join();

        // Create an EventBridge rule to route Object Created notifications.
        PutRuleRequest putRuleRequest = PutRuleRequest.builder()
            .name(RULE_NAME)
            .eventPattern("""
                {
                    "source": ["aws.s3"],
                    "detail-type": ["Object Created"],
                    "detail": {
                        "bucket": {
                            "name": ["%s"]
                        }
                    }
                }
            """).formatted(bucketName))
            .build();

        // Add the rule to the default event bus.
        PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
            .whenComplete((r, t) -> {
                if (t != null) {
                    logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                    throw new RuntimeException(t.getCause().getMessage(),
t);

```

```
        }
        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
    }).join();

    // Add the existing SNS topic and SQS queue as targets to the rule.
    eventBridgeClient.putTargets(b -> b
        .eventBusName("default")
        .rule(RULE_NAME)
        .targets(List.of (
            Target.builder()
                .arn(queueArn)
                .id("Queue")
                .build(),
            Target.builder()
                .arn(topicArn)
                .id("Topic")
                .build())
        ))
    ).join();
    return putRuleResponse.ruleArn();
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [PutBucketNotificationConfiguration](#)
  - [PutRule](#)
  - [PutTargets](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 オブジェクトのアップロードまたはダウンロードを追跡する

次のコード例は、Amazon S3 オブジェクトのアップロードまたはダウンロードを追跡する方法を示しています。

Java

SDK for Java 2.x

### Note

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

ファイルのアップロードの進行状況を追跡します。

```
public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    fileUpload.completionFuture().join();
    /*
        The SDK provides a LoggingTransferListener implementation of the
        TransferListener interface.
        You can also implement the interface to provide your own logic.

        Configure log4j2 with settings such as the following.
        <Configuration status="WARN">
            <Appenders>
                <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                    <PatternLayout pattern="%m%n"/>

```

```

        </Console>
    </Appenders>

    <Loggers>
        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
            <AppenderRef ref="AlignedConsoleAppender"/>
        </logger>
    </Loggers>
</Configuration>

```

Log4J2 logs the progress. The following is example output for a 21.3 MB file upload.

```

    Transfer initiated...
    |                               | 0.0%
    |====                         | 21.1%
    |=====                      | 60.5%
    |=====| 100.0%
    Transfer complete!

    */
}

```

ファイルのダウンロードの進捗状況を追跡します。

```

public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,

                                String key, String downloadedFileWithPath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.

        .destination(Paths.get(downloadedFileWithPath))
        .build();

    FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

    CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
    /*

```

The SDK provides a `LoggingTransferListener` implementation of the `TransferListener` interface.

You can also implement the interface to provide your own logic.

Configure log4J2 with settings such as the following.

```
<Configuration status="WARN">
  <Appenders>
    <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
      <PatternLayout pattern="%m%n"/>
    </Console>
  </Appenders>

  <Loggers>
    <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
      <AppenderRef ref="AlignedConsoleAppender"/>
    </logger>
  </Loggers>
</Configuration>
```

Log4J2 logs the progress. The following is example output for a 21.3 MB file download.

```
Transfer initiated...
|=====| 39.4%
|=====| 78.8%
|=====| 100.0%
Transfer complete!

  */
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [GetObject](#)
  - [PutObject](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## S3 Object Lambda でアプリケーションのデータを変換する

次のコード例では、S3 Object Lambda でアプリケーションのデータを変換する方法を示しています。

.NET

### SDK for .NET

オブジェクトがリクエストしたクライアントまたはアプリケーションのニーズに合うように、標準の S3 GET リクエストにカスタムコードを追加し、S3 から取得したリクエストされたオブジェクトを変更する方法を示しています。

完全なソースコードとセットアップおよび実行の手順については、[GitHub](#) で完全な例を参照してください。

この例で使用されているサービス

- Lambda
- Amazon S3

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK による単体テストと統合テストのアプローチ例

次のコード例は、AWS SDK を使用して単体テストと統合テストを作成する際のベストプラクティス手法の例を示しています。

Rust

### SDK for Rust

#### Note

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

Cargo.toml はテストサンプル用です。

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
  "Doug Schwartz <dougsch@amazon.com>",
]
edition = "2021"

[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }

[[bin]]
name = "main"
path = "src/main.rs"
```

オートモックとサービ斯拉ッパーを使ったユニットテストの例。

```
use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};

#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
```



```
pub use S3Impl as S3;

#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
}

#[cfg_attr(test, automock)]
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
```

```

        .await?;

    // Add up the file sizes we got back
    for object in result.contents() {
        total_size_bytes += object.size().unwrap_or(0) as usize;
    }

    // Handle pagination, and break the loop if there are no more pages
    next_token = result.next_continuation_token.clone();
    if next_token.is_none() {
        break;
    }
}
Ok(total_size_bytes)
}

#[cfg(test)]
mod test {
    use super::*;
    use mockall::predicate::eq;

    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
    }
}

```

```

#[tokio::test]
async fn test_multiple_pages() {
    // Create the Mock instance with two pages of objects now
    let mut mock = MockS3Impl::default();
    mock.expect_list_objects()
        .with(eq("test-bucket"), eq("test-prefix"), eq(None))
        .return_once(|_, _, _| {
            Ok(ListObjectsV2Output::builder()
                .set_contents(Some(vec![
                    // Mock content for ListObjectsV2 response
                    s3::types::Object::builder().size(5).build(),
                    s3::types::Object::builder().size(2).build(),
                ]))
                .set_next_continuation_token(Some("next".to_string()))
                .build())
        });
    mock.expect_list_objects()
        .with(
            eq("test-bucket"),
            eq("test-prefix"),
            eq(Some("next".to_string())),
        )
        .return_once(|_, _, _| {
            Ok(ListObjectsV2Output::builder()
                .set_contents(Some(vec![
                    // Mock content for ListObjectsV2 response
                    s3::types::Object::builder().size(3).build(),
                    s3::types::Object::builder().size(9).build(),
                ]))
                .build())
        });

    // Run the code we want to test with it
    let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
        .await
        .unwrap();

    assert_eq!(19, size);
}
}

```

## StaticReplayClient を使用した統合テストの例。

```
use aws_sdk_s3 as s3;

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[allow(dead_code)]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "atestsecretkey",
        Some("atestsessiontoken".to_string()),
    )
}
```

```

        None,
        "",
    )
}

#[cfg(test)]
mod test {
    use super::*;
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
            http::Request::builder()
                .method("GET")
                .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                .body(SdkBody::empty())
                .unwrap(),
            http::Response::builder()
                .status(200)
                .body(SdkBody::from(include_str!("./testing/
response_1.xml")))
                .unwrap(),
        );

        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();
    }
}

```

```

    // Verify we got the correct total size back
    assert_eq!(7, size);
    replay_client.assert_requests_match(&[]);
}

#[tokio::test]
async fn test_multiple_pages() {
    let page_1 = ReplayEvent::new(
        http::Request::builder()
            .method("GET")
            .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
            .body(SdkBody::empty())
            .unwrap(),
        http::Response::builder()
            .status(200)
            .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
            .unwrap(),
    );
    let page_2 = ReplayEvent::new(
        http::Request::builder()
            .method("GET")
            .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
            .body(SdkBody::empty())
            .unwrap(),
        http::Response::builder()
            .status(200)
            .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))
            .unwrap(),
    );
    let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
    let client: s3::Client = s3::Client::from_conf(
        s3::Config::builder()
            .behavior_version(BehaviorVersion::latest())
            .credentials_provider(make_s3_test_credentials())
            .region(s3::config::Region::new("us-east-1"))
            .http_client(replay_client.clone())
            .build(),
    );
};

```

```
// Run the code we want to test with it
let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
    .await
    .unwrap();

assert_eq!(19, size);

replay_client.assert_requests_match(&[]);
}
}
```

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## ローカルディレクトリを Amazon Simple Storage Service (Amazon S3) バケットに、再帰的にアップロードする

以下のコード例では、ローカルディレクトリを再帰的に Amazon Simple Storage Service (Amazon S3) バケットにアップロードする方法を示します。

Java

SDK for Java 2.x

### Note

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

[S3TransferManager](#) を使用して[ローカルディレクトリをアップロード](#)します。[完全なファイルとテスト](#)を表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
```

```
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;

public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
    .source(Paths.get(sourceDirectory))
    .bucket(bucketName)
    .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 との間で大きなファイルをアップロードまたはダウンロードする

次のコード例は、Amazon S3 との間で大きなファイルをアップロードまたはダウンロードする方法を示しています。

詳細については、「[マルチパートアップロードを使用したオブジェクトのアップロード](#)」を参照してください。



## .NET

### SDK for .NET

#### Note

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

Amazon S3 TransferUtility を使用して、S3 バケットとの間でファイルを転送する関数を呼び出します。

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;

// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
```

```
var localPath =
    $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
    \TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
    bucketName, fileToUpload, localPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
    {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
    bucketName, keyPrefix, uploadPath);
if (success)
{
```

```
        Console.WriteLine($"Successfully uploaded the files in {uploadPath} to  
        {bucketName}.");  
        Console.WriteLine($"{bucketName} currently contains the following files:");  
        await DisplayBucketFiles(client, bucketName, keyPrefix);  
        Console.WriteLine();  
    }  
  
    PressEnter();  
  
    // Download a single file from an S3 bucket.  
    DisplayTitle("Download a single file");  
    Console.WriteLine("Now we will download a single file from an S3 bucket.");  
  
    var keyName = _configuration["FileToDownload"];  
  
    Console.WriteLine($"Downloading {keyName} from {bucketName}.");  
  
    success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,  
        keyName, localPath);  
    if (success)  
    {  
        Console.WriteLine($"Successfully downloaded the file, {keyName} from  
        {bucketName}.");  
    }  
  
    PressEnter();  
  
    // Download the contents of a directory from an S3 bucket.  
    DisplayTitle("Download the contents of an S3 bucket");  
    var s3Path = _configuration["S3Path"];  
    var downloadPath = $"{localPath}\\{s3Path}";  
  
    Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");  
    Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");  
    await DisplayBucketFiles(client, bucketName, s3Path);  
    Console.WriteLine();  
  
    success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,  
        bucketName, s3Path, downloadPath);  
    if (success)  
    {  
        Console.WriteLine($"Downloaded the files in {bucketName} to  
        {downloadPath}.");  
        Console.WriteLine($"{downloadPath} now contains the following files:");  
    }
```

```
        DisplayLocalFiles(downloadPath);
    }

    Console.WriteLine("\nThe TransferUtility Basics application has completed.");
    PressEnter();

    // Displays the title for a section of the scenario.
    static void DisplayTitle(string titleText)
    {
        var sepBar = new string('-', Console.WindowWidth);

        Console.WriteLine(sepBar);
        Console.WriteLine(CenterText(titleText));
        Console.WriteLine(sepBar);
    }

    // Displays a description of the actions to be performed by the scenario.
    static void DisplayInstructions()
    {
        var sepBar = new string('-', Console.WindowWidth);

        DisplayTitle("Amazon S3 Transfer Utility Basics");
        Console.WriteLine("This program shows how to use the Amazon S3 Transfer
Utility.");
        Console.WriteLine("It performs the following actions:");
        Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
        Console.WriteLine("\t2. Upload an entire directory from the local computer to
an\n\t S3 bucket.");
        Console.WriteLine("\t3. Download a single object from an S3 bucket.");
        Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
directory.");
        Console.WriteLine($" \n{sepBar}");
    }

    // Pauses the scenario.
    static void PressEnter()
    {
        Console.WriteLine("Press <Enter> to continue.");
        _ = Console.ReadLine();
        Console.WriteLine("\n");
    }

    // Returns the string textToCenter, padded on the left with spaces
    // that center the text on the console display.
```

```
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
    }
}

// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
s3Path)
{
    ListObjectsV2Request request = new()
    {
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
    }
```

```
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}
```

1つのファイルをアップロードします。

```
/// <summary>
/// Uploads a single file from the local computer to an S3 bucket.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The name of the S3 bucket where the file
/// will be stored.</param>
/// <param name="fileName">The name of the file to upload.</param>
/// <param name="localPath">The local path where the file is stored.</
param>
/// <returns>A boolean value indicating the success of the action.</
returns>
public static async Task<bool> UploadSingleFileAsync(
    TransferUtility transferUtil,
    string bucketName,
    string fileName,
    string localPath)
{
    if (File.Exists($"{localPath}\\{fileName}"))
    {
        try
        {
            await transferUtil.UploadAsync(new
TransferUtilityUploadRequest
            {
                BucketName = bucketName,
                Key = fileName,
                FilePath = $"{localPath}\\{fileName}",
            });

            return true;
        }
        catch (AmazonS3Exception s3Ex)
```

```
        {
            Console.WriteLine($"Could not upload {fileName} from
{localPath} because:");
            Console.WriteLine(s3Ex.Message);
            return false;
        }
    }
else
{
    Console.WriteLine($"{{fileName}} does not exist in {localPath}");
    return false;
}
}
```

ローカルディレクトリ全体をアップロードします。

```
/// <summary>
/// Uploads all the files in a local directory to a directory in an S3
/// bucket.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The name of the S3 bucket where the files
/// will be stored.</param>
/// <param name="keyPrefix">The key prefix is the S3 directory where
/// the files will be stored.</param>
/// <param name="localPath">The local directory that contains the files
/// to be uploaded.</param>
/// <returns>A Boolean value representing the success of the action.</
returns>
public static async Task<bool> UploadFullDirectoryAsync(
    TransferUtility transferUtil,
    string bucketName,
    string keyPrefix,
    string localPath)
{
    if (Directory.Exists(localPath))
    {
        try
        {
```

```

        await transferUtil.UploadDirectoryAsync(new
TransferUtilityUploadDirectoryRequest
        {
            BucketName = bucketName,
            KeyPrefix = keyPrefix,
            Directory = localPath,
        });

        return true;
    }
    catch (AmazonS3Exception s3Ex)
    {
        Console.WriteLine($"Can't upload the contents of {localPath}
because:");

        Console.WriteLine(s3Ex?.Message);
        return false;
    }
}
else
{
    Console.WriteLine($"The directory {localPath} does not exist.");
    return false;
}
}

```

1 つのファイルをダウンロードします。

```

/// <summary>
/// Download a single file from an S3 bucket to the local computer.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The name of the S3 bucket containing the
/// file to download.</param>
/// <param name="keyName">The name of the file to download.</param>
/// <param name="localPath">The path on the local computer where the
/// downloaded file will be saved.</param>
/// <returns>A Boolean value indicating the results of the action.</
returns>
public static async Task<bool> DownloadSingleFileAsync(

```



```

TransferUtility transferUtil,
    string bucketName,
    string keyName,
    string localPath)
{
    await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
    {
        BucketName = bucketName,
        Key = keyName,
        FilePath = $"{localPath}\\{keyName}",
    });

    return (File.Exists($"{localPath}\\{keyName}"));
}

```

S3 バケットの内容をダウンロードします。

```

/// <summary>
/// Downloads the contents of a directory in an S3 bucket to a
/// directory on the local computer.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The bucket containing the files to
download.</param>
/// <param name="s3Path">The S3 directory where the files are located.</
param>
/// <param name="localPath">The local path to which the files will be
/// saved.</param>
/// <returns>A Boolean value representing the success of the action.</
returns>
public static async Task<bool> DownloadS3DirectoryAsync(
    TransferUtility transferUtil,
    string bucketName,
    string s3Path,
    string localPath)
{
    int fileCount = 0;

    // If the directory doesn't exist, it will be created.

```

```
        if (Directory.Exists(s3Path))
        {
            var files = Directory.GetFiles(localPath);
            fileCount = files.Length;
        }

        await transferUtil.DownloadDirectoryAsync(new
TransferUtilityDownloadDirectoryRequest
        {
            BucketName = bucketName,
            LocalDirectory = localPath,
            S3Directory = s3Path,
        });

        if (Directory.Exists(localPath))
        {
            var files = Directory.GetFiles(localPath);
            if (files.Length > fileCount)
            {
                return true;
            }

            // No change in the number of files. Assume
            // the download failed.
            return false;
        }

        // The local directory doesn't exist. No files
        // were downloaded.
        return false;
    }
}
```

TransferUtility を使用してアップロードの進行状況を追跡します。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to track the progress of a multipart upload
```

```
/// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
/// upload to an Amazon S3 bucket.
/// </summary>
public class TrackMPUUsingHighLevelAPI
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "sample_pic.png";
        string path = "filepath/directory/";
        string filePath = $"{path}{keyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        await TrackMPUAsync(client, bucketName, filePath, keyName);
    }

    /// <summary>
    /// Starts an Amazon S3 multipart upload and assigns an event handler to
    /// track the progress of the upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// perform the multipart upload.</param>
    /// <param name="bucketName">The name of the bucket to which to upload
    /// the file.</param>
    /// <param name="filePath">The path, including the file name of the
    /// file to be uploaded to the Amazon S3 bucket.</param>
    /// <param name="keyName">The file name to be used in the
    /// destination Amazon S3 bucket.</param>
    public static async Task TrackMPUAsync(
        IAmazonS3 client,
        string bucketName,
        string filePath,
        string keyName)
    {
        try
        {
            var fileTransferUtility = new TransferUtility(client);

            // Use TransferUtilityUploadRequest to configure options.
```

```
// In this example we subscribe to an event.
var uploadRequest =
    new TransferUtilityUploadRequest
    {
        BucketName = bucketName,
        FilePath = filePath,
        Key = keyName,
    };

uploadRequest.UploadProgressEvent +=
    new EventHandler<UploadProgressArgs>(
        UploadRequest_UploadPartProgressEvent);

await fileTransferUtility.UploadAsync(uploadRequest);
Console.WriteLine("Upload completed");
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error:: {ex.Message}");
}
}

/// <summary>
/// Event handler to check the progress of the multipart upload.
/// </summary>
/// <param name="sender">The object that raised the event.</param>
/// <param name="e">The object that contains multipart upload
/// information.</param>
public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
{
    // Process event.
    Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
}
}
```

暗号化したオブジェクトをアップロードします。

```
using System;
using System.Collections.Generic;
using System.IO;
```

```
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
/// perform a multipart upload to an Amazon S3 bucket.
/// </summary>
public class SSECLowLevelMPUcopyObject
{
    public static async Task Main()
    {
        string existingBucketName = "amzn-s3-demo-bucket";
        string sourceKeyName = "sample_file.txt";
        string targetKeyName = "sample_file_copy.txt";
        string filePath = $"sample\\{targetKeyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USEast1.
        IAmazonS3 client = new AmazonS3Client();

        // Create the encryption key.
        var base64Key = CreateEncryptionKey();

        await CreateSampleObjUsingClientEncryptionKeyAsync(
            client,
            existingBucketName,
            sourceKeyName,
            filePath,
            base64Key);
    }

    /// <summary>
    /// Creates the encryption key to use with the multipart upload.
    /// </summary>
    /// <returns>A string containing the base64-encoded key for encrypting
    /// the multipart upload.</returns>
    public static string CreateEncryptionKey()
    {
        Aes aesEncryption = Aes.Create();
        aesEncryption.KeySize = 256;
    }
}
```

```

        aesEncryption.GenerateKey();
        string base64Key = Convert.ToBase64String(aesEncryption.Key);
        return base64Key;
    }

    /// <summary>
    /// Creates and uploads an object using a multipart upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 object used to
    /// initialize and perform the multipart upload.</param>
    /// <param name="existingBucketName">The name of the bucket to which
    /// the object will be uploaded.</param>
    /// <param name="sourceKeyName">The source object name.</param>
    /// <param name="filePath">The location of the source object.</param>
    /// <param name="base64Key">The encryption key to use with the upload.</
param>
    public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
        IAmazonS3 client,
        string existingBucketName,
        string sourceKeyName,
        string filePath,
        string base64Key)
    {
        List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

        InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
        {
            BucketName = existingBucketName,
            Key = sourceKeyName,
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };

        InitiateMultipartUploadResponse initResponse =
            await client.InitiateMultipartUploadAsync(initiateRequest);

        long contentLength = new FileInfo(filePath).Length;
        long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

        try
        {

```

```
        long filePosition = 0;
        for (int i = 1; filePosition < contentLength; i++)
        {
            UploadPartRequest uploadRequest = new UploadPartRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                UploadId = initResponse.UploadId,
                PartNumber = i,
                PartSize = partSize,
                FilePosition = filePosition,
                FilePath = filePath,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            // Upload part and add response to our list.
            uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));

            filePosition += partSize;
        }

        CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
        {
            BucketName = existingBucketName,
            Key = sourceKeyName,
            UploadId = initResponse.UploadId,
        };
        completeRequest.AddPartETags(uploadResponses);

        CompleteMultipartUploadResponse completeUploadResponse =
            await client.CompleteMultipartUploadAsync(completeRequest);
    }
    catch (Exception exception)
    {
        Console.WriteLine($"Exception occurred: {exception.Message}");

        // If there was an error, abort the multipart upload.
        AbortMultipartUploadRequest abortMPURquest = new
AbortMultipartUploadRequest
        {
```

```
        BucketName = existingBucketName,  
        Key = sourceKeyName,  
        UploadId = initResponse.UploadId,  
    };  
  
    await client.AbortMultipartUploadAsync(abortMPURequest);  
}  
}  
}
```

Go

SDK for Go V2

 Note

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

アップロードマネージャーとダウンロードマネージャーを使用してデータを分割し、同時に転送する関数を作成します。

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"
```



```

)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// UploadLargeObject uses an upload manager to upload data to an object in a
// bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
string, objectKey string, largeObject []byte) error {
    largeBuffer := bytes.NewReader(largeObject)
    var partMiBs int64 = 10
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
        u.PartSize = partMiBs * 1024 * 1024
    })
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    largeBuffer,
    })
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
            log.Printf("Error while uploading object to %s. The object is too large.\n"+
                "The maximum size for a multipart upload is 5TB.", bucketName)
        } else {
            log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
                bucketName, objectKey, err)
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                aws.String(objectKey)}, time.Minute)
        if err != nil {

```

```
    log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
}
}

return err
}

// DownloadLargeObject uses a download manager to download an object from a
// bucket.
// The download manager gets the data in parts and writes them to a buffer until
// all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
    string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10
    downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
    {
        d.PartSize = partMiBs * 1024 * 1024
    })
    buffer := manager.NewWriteAtBuffer([]byte{})
    _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    })
    if err != nil {
        log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return buffer.Bytes(), err
}
```

アップロードマネージャーとダウンロードマネージャーをコンテキストで使用方法を示すインタラクティブなシナリオを実行します。

```
import (
    "context"
    "crypto/rand"
    "log"
```

```

"strings"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunLargeObjectScenario is an interactive example that shows you how to use
// Amazon
// Simple Storage Service (Amazon S3) to upload and download large objects.
//
// 1. Create a bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 5. Download a large object by using a download manager.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../../demotools folder of this repo.
func RunLargeObjectScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner) {
defer func() {
if r := recover(); r != nil {
log.Println("Something went wrong with the demo.")
_, isMock := questioner.(*demotools.MockQuestioner)
if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
log.Println(r)
}
}
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 large object demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{s3Client: s3Client}

```

```
bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
    demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
_, _ = rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(ctx, bucketName, largeKey, largeBytes)
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(ctx, bucketName,
largeKey)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))
```

```
if questioner.AskBool("Do you want to delete your bucket and all of its "+
    "contents? (y/n)", "y") {
    log.Println("Deleting object.")
    err = bucketBasics.DeleteObjects(ctx, bucketName, []string{largeKey})
    if err != nil {
        panic(err)
    }
    log.Println("Deleting bucket.")
    err = bucketBasics.DeleteBucket(ctx, bucketName)
    if err != nil {
        panic(err)
    }
} else {
    log.Println("Okay. Don't forget to delete objects from your bucket to avoid
charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

## Java

### SDK for Java 2.x

#### Note

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

S3TransferManager を使用して、S3 バケットとの間でファイルを転送する関数を呼び出します。

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
```

```

        .destination(Paths.get(destinationPathURI))
        .bucket(bucketName)
        .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}

```

ローカルディレクトリ全体をアップロードします。

```

public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());

    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
    completedDirectoryUpload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryUpload.failedTransfers().size();
}

```

1つのファイルをアップロードします。

```

public String uploadFile(S3TransferManager transferManager, String
bucketName,
    String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
}

```

```
CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
return uploadResult.response().eTag();
}
```

コード例では、次のインポートを使用します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

[AWS CRT ベースの S3 クライアント](#) 上にある [S3 Transfer Manager](#) を使用すると、コンテンツのサイズがしきい値を超えたときにマルチパートアップロードを透過的に実行できます。デフォルトのしきい値は 8 MB です。

```
/**
 * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the file path of the file to be uploaded
 */
public void multipartUploadWithTransferManager(String bucketName, String key,
String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

[S3Client API](#) を使用してマルチパートアップロードを実行します。

```
/**
 * Performs a multipart upload to Amazon S3 using the provided S3 client.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the path to the file to be uploaded
 */
public void multipartUploadWithS3Client(String bucketName, String key, String
filePath) {

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
```



```
List<CompletedPart> completedParts = new ArrayList<>();
ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
    long fileSize = file.length();
    long position = 0;
    while (position < fileSize) {
        file.seek(position);
        long read = file.getChannel().read(bb);

        bb.flip(); // Swap position and limit before reading from the
buffer.

        UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .partNumber(partNumber)
            .build();

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNumber)
            .eTag(partResponse.eTag())
            .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    logger.error(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

    .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
```

```
}
```

マルチパートサポートを有効にして [S3AsyncClient API](#) を使用し、マルチパートアップロードを実行します。

```
/**
 * Uploads a file to an S3 bucket using the S3AsyncClient and enabling
 * multipart support.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the local file path of the file to be uploaded
 */
public void multipartUploadWithS3AsyncClient(String bucketName, String key,
String filePath) {
    // Enable multipart support.
    S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
        .multipartEnabled(true)
        .build();

    CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b
-> b
        .bucket(bucketName)
        .key(key),
        Paths.get(filePath));

    response.join();
    logger.info("File uploaded in multiple 8 MiB parts using
S3AsyncClient.");
}
```

## JavaScript

### SDK for JavaScript (v3)

#### Note

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

サイズの大きいファイルをアップロードします。

```
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

import {
  ProgressBar,
  logger,
} from "@aws-doc-sdk-examples/lib/utils/util-log.js";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

/**
 * Create a 25MB file and upload it in parts to the specified
 * Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const str = createString();
  const buffer = Buffer.from(str, "utf8");
  const progressBar = new ProgressBar({
    description: `Uploading "${key}" to "${bucketName}"`,
    barLength: 30,
  });

  try {
    const upload = new Upload({
      client: new S3Client({}),
      params: {
```

```

        Bucket: bucketName,
        Key: key,
        Body: buffer,
    },
});

upload.on("httpUploadProgress", ({ loaded, total }) => {
    progressBar.update({ current: loaded, total });
});

await upload.done();
} catch (caught) {
    if (caught instanceof Error && caught.name === "AbortError") {
        logger.error(`Multipart upload was aborted. ${caught.message}`);
    } else {
        throw caught;
    }
}
};

```

サイズの大きいファイルをダウンロードします。

```

import { fileURLToPath } from "node:url";
import { GetObjectCommand, NoSuchKey, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream, rmSync } from "node:fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
    const command = new GetObjectCommand({
        Bucket: bucket,
        Key: key,
        Range: `bytes=${start}-${end}`,
    });

    return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange

```

```

*/
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: Number.parseInt(start),
    end: Number.parseInt(end),
    length: Number.parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });
    console.log(`Downloaded bytes ${nextRange.start} to ${nextRange.end}`);

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
};

/**
 * Download a large object from and Amazon S3 bucket.
 *
 * When downloading a large file, you might want to break it down into
 * smaller pieces. Amazon S3 accepts a Range header to specify the start
 * and end of the byte range to be downloaded.
 *
 * @param {{ bucketName: string, key: string }}

```

```
*/  
export const main = async ({ bucketName, key }) => {  
  try {  
    await downloadInChunks({  
      bucket: bucketName,  
      key: key,  
    });  
  } catch (caught) {  
    if (caught instanceof NoSuchKey) {  
      console.error(`Failed to download object. No such key "${key}".`);  
      rmSync(key);  
    }  
  }  
};
```

## Python

### SDK for Python (Boto3)

#### Note

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

使用可能な転送マネージャの設定を使用して、ファイルを転送する関数を作成します。コールバッククラスを使用して、ファイル転送中にコールバックの進行状況を書き込みます。

```
import sys  
import threading  
  
import boto3  
from boto3.s3.transfer import TransferConfig  
  
MB = 1024 * 1024  
s3 = boto3.resource("s3")  
  
class TransferCallback:
```

```

"""
Handle callbacks from the transfer manager.

The transfer manager periodically calls the __call__ method throughout
the upload and download process so that it can take action, such as
displaying progress to the user and collecting data about the transfer.
"""

def __init__(self, target_size):
    self._target_size = target_size
    self._total_transferred = 0
    self._lock = threading.Lock()
    self.thread_info = {}

def __call__(self, bytes_transferred):
    """
    The callback method that is called by the transfer manager.

    Display progress during file transfer and collect per-thread transfer
    data. This method can be called by multiple threads, so shared instance
    data is protected by a thread lock.
    """
    thread = threading.current_thread()
    with self._lock:
        self._total_transferred += bytes_transferred
        if thread.ident not in self.thread_info.keys():
            self.thread_info[thread.ident] = bytes_transferred
        else:
            self.thread_info[thread.ident] += bytes_transferred

        target = self._target_size * MB
        sys.stdout.write(
            f"\r{self._total_transferred} of {target} transferred "
            f"({(self._total_transferred / target) * 100:.2f}%)."
        )
        sys.stdout.flush()

def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.

```

```

"""
transfer_callback = TransferCallback(file_size_mb)
s3.Bucket(bucket_name).upload_file(
    local_file_path, object_key, Callback=transfer_callback
)
return transfer_callback.thread_info

def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
    return transfer_callback.thread_info

def upload_with_high_threshold(local_file_path, bucket_name, object_key,
    file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of

```



```
a multipart upload.
"""
transfer_callback = TransferCallback(file_size_mb)
config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
s3.Bucket(bucket_name).upload_file(
    local_file_path, object_key, Config=config, Callback=transfer_callback
)
return transfer_callback.thread_info

def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
Callback=transfer_callback
    )
    return transfer_callback.thread_info

def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
```

```
)
return transfer_callback.thread_info

def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.
```

```
When this kind of encryption is specified, Amazon S3 encrypts the object
at rest and allows downloads only when the expected encryption key is
provided in the download request.
"""

transfer_callback = TransferCallback(file_size_mb)

if sse_key:
    extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
sse_key}
else:
    extra_args = None
s3.Bucket(bucket_name).Object(object_key).download_file(
    download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
)
return transfer_callback.thread_info
```

転送マネージャの機能とレポート結果のデモンストレーションを行います。

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
```

```
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")

class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
files? "
            )
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
"demonstration files? "
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
```

```

        except ParamValidationError as err:
            print(err)
            self.demo_bucket = None
        except ClientError as err:
            print(err)
            print(
                f"Either {self.demo_bucket} doesn't exist or you don't "
                f"have access to it."
            )
            self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
        download_args=None
    ):
        """Run a demonstration.

        Ask the user if they want to run this specific demonstration.
        If they say yes, create a file on the local path, upload it
        using the specified upload function, then download it using the
        specified download function.
        """
        if download_args is None:
            download_args = {}
        if upload_args is None:
            upload_args = {}
        question = question.format(self.file_size_mb)
        answer = input(f"{question} (y/n)")
        if answer.lower() == "y":
            local_file_path, object_key, download_file_path =
self._create_demo_file()

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Uploading", local_file_path, object_key, **upload_args
            )
            start_time = time.perf_counter()
            thread_info = upload_func(
                local_file_path,
                self.demo_bucket,
                object_key,
                self.file_size_mb,

```

```

        **upload_args,
    )
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

    file_transfer.TransferConfig = self._config_wrapper(
        TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
    )
    self._report_transfer_params(
        "Downloading", object_key, download_file_path, **download_args
    )
    start_time = time.perf_counter()
    thread_info = download_func(
        self.demo_bucket,
        object_key,
        download_file_path,
        self.file_size_mb,
        **download_args,
    )
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

def last_name_set(self):
    """Get the name set used for the last demo."""
    return self._chore_list[-1]

def cleanup(self):
    """
    Remove files from the demo folder, and uploaded objects from the
    Amazon S3 bucket.
    """
    print("-" * self._terminal_width)
    for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
        print(f"Removing {local_file_path}")
        try:
            os.remove(local_file_path)
        except FileNotFoundError as err:
            print(err)

        print(f"Removing {downloaded_file_path}")
        try:
            os.remove(downloaded_file_path)
        except FileNotFoundError as err:

```

```

        print(err)

    if self.demo_bucket:
        print(f"Removing {self.demo_bucket}:{s3_object_key}")
        try:

self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
            except ClientError as err:
                print(err)

def _setup_platform_specific(self):
    """Set up platform-specific command used to create a large file."""
    if platform.system() == "Windows":
        self._create_file_cmd = "fsutil file createnew {} {}".format(
            self._size_multiplier, MB)
    elif platform.system() == "Linux" or platform.system() == "Darwin":
        self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={{MB}}
count={{}}"
        self._size_multiplier = 1
    else:
        raise EnvironmentError(
            f"Demo of platform {platform.system()} isn't supported."
        )

def _create_demo_file(self):
    """
    Create a file in the demo folder specified by the user. Store the local
    path, object name, and download path for later cleanup.

    Only the local file is created by this method. The Amazon S3 object and
    download file are created later during the demonstration.

    Returns:
    A tuple that contains the local file path, object name, and download
    file path.
    """
    file_name_template = "TestFile{}-{}.demo"
    local_suffix = "local"
    object_suffix = "s3object"
    download_suffix = "downloaded"
    file_tag = len(self._chore_list) + 1

    local_file_path = os.path.join(
        self.demo_folder, file_name_template.format(file_tag, local_suffix)
    )

```

```

    )

    s3_object_key = file_name_template.format(file_tag, object_suffix)

    downloaded_file_path = os.path.join(
        self.demo_folder, file_name_template.format(file_tag,
download_suffix)
    )

    filled_cmd = self._create_file_cmd.format(
        local_file_path, self.file_size_mb * self._size_multiplier
    )

    print(
        f"Creating file of size {self.file_size_mb} MB "
        f"in {self.demo_folder} by running:"
    )
    print(f"{'':4}{filled_cmd}")
    os.system(filled_cmd)

    chore = (local_file_path, s3_object_key, downloaded_file_path)
    self._chore_list.append(chore)
    return chore

def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
    """Report configuration and extra arguments used for a file transfer."""
    print("-" * self._terminal_width)
    print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
    if kwargs:
        print("With extra args:")
        for arg, value in kwargs.items():
            print(f"{'':4}{arg:<20}: {value}'")

    @staticmethod
    def ask_user(question):
        """
        Ask the user a yes or no question.

        Returns:
        True when the user answers 'y' or 'Y'; otherwise, False.
        """
        answer = input(f"{question} (y/n) ")
        return answer.lower() == "y"

```



```

    @staticmethod
    def _config_wrapper(func, config_attrs):
        def wrapper(*args, **kwargs):
            config = func(*args, **kwargs)
            print("With configuration:")
            for attr in config_attrs:
                print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
            return config

        return wrapper

    @staticmethod
    def _report_transfer_result(thread_info, elapsed):
        """Report the result of a transfer, including per-thread data."""
        print(f"\nUsed {len(thread_info)} threads.")
        for ident, byte_count in thread_info.items():
            print(f'{"":4}Thread {ident} copied {byte_count} bytes.')
        print(f"Your transfer took {elapsed:.2f} seconds.")

def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,

```

```
        file_transfer.download_with_high_threshold,
    )

# Upload with specific chunk size and additional metadata.
# Download with a single thread.
demo_manager.demo(
    "Do you want to upload a {} MB file with a smaller chunk size and "
    "then download the same file using a single thread?",
    file_transfer.upload_with_chunksize_and_meta,
    file_transfer.download_with_single_thread,
    upload_args={
        "metadata": {
            "upload_type": "chunky",
            "favorite_color": "aqua",
            "size": "medium",
        }
    },
)

# Upload using server-side encryption with customer-provided
# encryption keys.
# Generate a 256-bit key from a passphrase.
sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
demo_manager.demo(
    "Do you want to upload and download a {} MB file using "
    "server-side encryption?",
    file_transfer.upload_with_sse,
    file_transfer.download_with_sse,
    upload_args={"sse_key": sse_key},
    download_args={"sse_key": sse_key},
)

# Download without specifying an encryption key to show that the
# encryption key must be included to download an encrypted object.
if demo_manager.ask_user(
    "Do you want to try to download the encrypted "
    "object without sending the required key?"
):
    try:
        _, object_key, download_file_path = demo_manager.last_name_set()
        file_transfer.download_with_default_configuration(
            demo_manager.demo_bucket,
            object_key,
            download_file_path,
```

```
        demo_manager.file_size_mb,
    )
except ClientError as err:
    print(
        "Got expected error when trying to download an encrypted "
        "object without specifying encryption info:"
    )
    print(f"{'':4}{err}")

# Remove all created and downloaded files, remove all objects from
# S3 storage.
if demo_manager.ask_user(
    "Demonstration complete. Do you want to remove local files " "and S3
objects?"
):
    demo_manager.cleanup()

if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

## Rust

### SDK for Rust

#### Note

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

```
use std::fs::File;
```

```

use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
    get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_code_examples::error::S3ExampleError;
use std::process;
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}

async fn run_example() -> Result<(), S3ExampleError> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;

    let key = "sample.txt".to_string();
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client

```

```
.create_multipart_upload()
.bucket(&bucket_name)
.key(&key)
.send()
.await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;

//Create a file of random characters for the upload.
let mut file = File::create(&key).expect("Could not create sample file.");
// Loop until the file is 5 chunks.
while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
    let rand_string: String = thread_rng()
        .sample_iter(&Alphanumeric)
        .take(256)
        .map(char::from)
        .collect();
    let return_string: String = "\n".to_string();
    file.write_all(rand_string.as_ref())
        .expect("Error writing to file.");
    file.write_all(return_string.as_ref())
        .expect("Error writing to file.");
}

let path = Path::new(&key);
let file_size = tokio::fs::metadata(path)
    .await
    .expect("it exists I swear")
    .len();

let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
let mut size_of_last_chunk = file_size % CHUNK_SIZE;
if size_of_last_chunk == 0 {
    size_of_last_chunk = CHUNK_SIZE;
    chunk_count -= 1;
}

if file_size == 0 {
    return Err(S3ExampleError::new("Bad file size."));
}
if chunk_count > MAX_CHUNKS {
    return Err(S3ExampleError::new(
```

```
        "Too many chunks! Try increasing your chunk size.",
    ));
}

let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
            .build(),
    );
}

// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
    CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
```

```
        .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;

let data: GetObjectOutput =
    s3_code_examples::download_object(&client, &bucket_name, &key).await?;
let data_length: u64 = data
    .content_length()
    .unwrap_or_default()
    .try_into()
    .unwrap();
if file.metadata().unwrap().len() == data_length {
    println!("Data lengths match.");
} else {
    println!("The data was not the same size!");
}

s3_code_examples::clear_bucket(&client, &bucket_name)
    .await
    .expect("Error emptying bucket.");
s3_code_examples::delete_bucket(&client, &bucket_name)
    .await
    .expect("Error deleting bucket.");

Ok(())
}
```

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## サイズが不明なストリームを AWS SDK を使用して Amazon S3 オブジェクトにアップロードする

次のコード例は、サイズが不明なストリームを Amazon S3 オブジェクトにアップロードする方法を示しています。

Java

SDK for Java 2.x

### Note

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

[AWS CRT ベースの S3 クライアント](#)を使用します。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsync {
    private static final Logger logger =
        LoggerFactory.getLogger(PutObjectFromStreamAsync.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.
        String key = UUID.randomUUID().toString();
```



```
AsyncExampleUtils.createBucket(bucketName);
try {
    PutObjectFromStreamAsync example = new PutObjectFromStreamAsync();
    S3AsyncClient s3AsyncClientCrt = S3AsyncClient.crtCreate();
    PutObjectResponse putObjectResponse =
example.putObjectFromStreamCrt(s3AsyncClientCrt, bucketName, key);
    logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
    logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
} catch (SdkException e) {
    logger.error(e.getMessage(), e);
} finally {
    AsyncExampleUtils.deleteObject(bucketName, key);
    AsyncExampleUtils.deleteBucket(bucketName);
}
}

/**
 * @param s3CrtAsyncClient - To upload content from a stream of unknown
size, use can the AWS CRT-based S3 client.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
Returns metadata pertaining to the put object operation.
 */
public PutObjectResponse putObjectFromStreamCrt(S3AsyncClient
s3CrtAsyncClient, String bucketName, String key) {

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
    InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());

    // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
    ExecutorService executor = Executors.newSingleThreadExecutor();
    // Specify `null` for the content length when you don't know the content
length.
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
null, executor);
```

```
        CompletableFuture<PutObjectResponse> responseFuture =
            s3CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
response.
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

[マルチパートアップロードが有効になっている標準の非同期 S3 クライアント](#)を使用します。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsyncMp {
    private static final Logger logger =
        LoggerFactory.getLogger(PutObjectFromStreamAsyncMp.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.

        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
```

```

        PutObjectFromStreamAsyncMp example = new
PutObjectFromStreamAsyncMp();
        S3AsyncClient s3AsyncClientMp =
S3AsyncClient.builder().multipartEnabled(true).build();
        PutObjectResponse putObjectResponse =
example.putObjectFromStreamMp(s3AsyncClientMp, bucketName, key);
        logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
    } catch (SdkException e) {
        logger.error(e.getMessage(), e);
    } finally {
        AsyncExampleUtils.deleteObject(bucketName, key);
        AsyncExampleUtils.deleteBucket(bucketName);
    }
}

/**
 * @param s3AsyncClientMp - To upload content from a stream of unknown size,
use can the S3 asynchronous client with multipart enabled.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
Returns metadata pertaining to the put object operation.
 */
public PutObjectResponse putObjectFromStreamMp(S3AsyncClient s3AsyncClientMp,
String bucketName, String key) {

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
    InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());

    // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
    ExecutorService executor = Executors.newSingleThreadExecutor();
    // Specify `null` for the content length when you don't know the content
length.
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
null, executor);

    CompletableFuture<PutObjectResponse> responseFuture =

```

```
s3AsyncClientMp.putObject(r -> r.bucket(bucketName).key(key),
body);

    PutObjectResponse response = responseFuture.join(); // Wait for the
response.
    logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
    executor.shutdown();
    return response;
}
}
```

[Amazon S3 Transfer Manager](#) を使用します。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class UploadStream {
    private static final Logger logger =
        LoggerFactory.getLogger(UploadStream.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            UploadStream example = new UploadStream();
```

```

        CompletedUpload completedUpload =
example.uploadStream(S3TransferManager.create(), bucketName, key);
        logger.info("Object {} etag: {}", key,
completedUpload.response().eTag());
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
    } catch (SdkException e) {
        logger.error(e.getMessage(), e);
    } finally {
        AsyncExampleUtils.deleteObject(bucketName, key);
        AsyncExampleUtils.deleteBucket(bucketName);
    }
}

/**
 * @param transferManager - To upload content from a stream of unknown size,
you can use the S3TransferManager based on the AWS CRT-based S3 client.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
result of the completed upload.
 */
public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
    InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());

    // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
    ExecutorService executor = Executors.newSingleThreadExecutor();
    // Specify `null` for the content length when you don't know the content
length.
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
null, executor);

    Upload upload = transferManager.upload(builder -> builder
        .requestBody(body)
        .putObjectRequest(req -> req.bucket(bucketName).key(key))
        .build());

```

```
CompletedUpload completedUpload = upload.completionFuture().join();
executor.shutdown();
return completedUpload;
}
}
```

## Swift

### SDK for Swift

#### Note

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

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams

/// Upload a file to the specified bucket.
///
/// - Parameters:
///   - bucket: The Amazon S3 bucket name to store the file into.
///   - key: The name (or path) of the file to upload to in the `bucket`.
///   - sourcePath: The pathname on the local filesystem of the file to
///     upload.
func uploadFile(sourcePath: String, bucket: String, key: String?) async
throws {
    let fileURL: URL = URL(fileURLWithPath: sourcePath)
    let fileName: String

    // If no key was provided, use the last component of the filename.
```

```
if key == nil {
    fileName = fileURL.lastPathComponent
} else {
    fileName = key!
}

let s3Client = try await S3Client()

// Create a FileHandle for the source file.

let fileHandle = FileHandle(forReadingAtPath: sourcePath)
guard let fileHandle = fileHandle else {
    throw TransferError.readError
}

// Create a byte stream to retrieve the file's contents. This uses the
// Smithy FileStream and ByteStream types.

let stream = FileStream(fileHandle: fileHandle)
let body = ByteStream.stream(stream)

// Create a `PutObjectInput` with the ByteStream as the body of the
// request's data. The AWS SDK for Swift will handle sending the
// entire file in chunks, regardless of its size.

let putInput = PutObjectInput(
    body: body,
    bucket: bucket,
    key: fileName
)

do {
    _ = try await s3Client.putObject(input: putInput)
} catch {
    throw TransferError.uploadError("Error uploading the file: \(error)")
}

print("File uploaded to \(fileURL.path).")
}
```

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## チェックサムを使用した AWS SDK での Amazon S3 オブジェクトの操作

次のコード例は、チェックサムを使用して Amazon S3 オブジェクトを操作する方法を示しています。

Java

SDK for Java 2.x

### Note

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

コード例では、以下のインポートのサブセットを使用しています。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
```



```
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

[PutObjectRequest をビルドする](#)ときに、putObject メソッドのチェックサムアルゴリズムを指定します。

```
public void putObjectWithChecksum(String bucketName, String key) {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```

[GetObjectRequest をビルドする](#)ときに、getObject メソッドのチェックサムを確認します。

```
public GetObjectResponse getObjectWithChecksum(String bucketName, String key)
{
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

[PutObjectRequest をビルドする](#)ときに、putObject メソッドのチェックサムを事前に計算します。

```
public void putObjectWithPrecalculatedChecksum(String bucketName, String key,
String filePath) {
    String checksum = calculateChecksum(filePath, "SHA-256");

    s3Client.putObject((b -> b
        .bucket(bucketName)
        .key(key)
        .checksumSHA256(checksum)),
        RequestBody.fromFile(Paths.get(filePath)));
}
```

[AWS CRT ベースの S3 クライアント](#) 上にある [S3 Transfer Manager](#) を使用して、コンテンツのサイズがしきい値を超えたときにマルチパートアップロードを透過的に実行します。デフォルトのしきい値は 8 MB です。

SDK で使用するチェックサムアルゴリズムを指定できます。デフォルトでは、SDK は CRC32 アルゴリズムを使用します。

```
public void multipartUploadWithChecksumTm(String bucketName, String key,
String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key)
            .checksumAlgorithm(ChecksumAlgorithm.SHA1))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

[S3Client API](#) または (S3AsyncClient API) を使用してマルチパートアップロードを実行します。追加のチェックサムを指定する場合は、アップロードの開始時に使用するアルゴリズムを指定する必要があります。また、パートリクエストごとにアルゴリズムを指定し、アップロード後にパートごとに計算されたチェックサムを指定する必要があります。

```
public void multipartUploadWithChecksumS3Client(String bucketName, String
key, String filePath) {
    ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .checksumAlgorithm(algorithm) // Checksum specified on each
part.

                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
                RequestBody.fromByteBuffer(bb));

            CompletedPart part = CompletedPart.builder()
                .partNumber(partNumber)
                .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
```

```
        .eTag(partResponse.eTag())
        .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    System.err.println(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [CompleteMultipartUpload](#)
  - [CreateMultipartUpload](#)
  - [UploadPart](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 オブジェクトの整合性機能を実操作する

次のコード例は、S3 オブジェクトの整合性機能を実操作する方法を示しています。

## C++

## SDK for C++

 Note

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

Amazon S3 オブジェクトの整合性機能を示すインタラクティブなシナリオを実行します。

```
//! Routine which runs the S3 object integrity workflow.
/*!
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
    const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
std::endl;
        return false;
    }

    Aws::String bucketName = TEST_BUCKET_PREFIX;
    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
        std::cerr << "Workflow exiting because bucket creation failed." <<
std::endl;
        return false;
    }
}
```

```

    Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
    std::shared_ptr<Aws::S3::S3Client> client =
    Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

    printAsterisksLine();
    std::cout << "Choose from one of the following checksum algorithms."
              << std::endl;

    for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
        std::cout << "  " << hashMethod << " - " <<
    stringForHashMethod(hashMethod)
              << std::endl;
    }

    HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
    DEFAULT,
                                                    SHA256);

    gUseCalculatedChecksum = !askYesNoQuestion(
        "Let the SDK calculate the checksum for you? (y/n) ");

    printAsterisksLine();

    std::cout << "The workflow will now upload a file using PutObject."
              << std::endl;
    std::cout << "Object integrity will be verified using the "
              << stringForHashMethod(chosenHashMethod) << " algorithm."
              << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout
            << "A checksum computed by this workflow will be used for object
integrity verification,"
            << std::endl;
        std::cout << "except for the TransferManager upload." << std::endl;
    } else {
        std::cout
            << "A checksum computed by the SDK will be used for object
integrity verification."
            << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();

```

```
std::shared_ptr<Aws::IOStream> inputData =
    Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                   TEST_FILE,
                                   std::ios_base::in |
                                   std::ios_base::binary);

if (!*inputData) {
    std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

Hasher hasher;
HASH_METHOD putObjectHashMethod = chosenHashMethod;
if (putObjectHashMethod == DEFAULT) {
    putObjectHashMethod = MD5; // MD5 is the default hash method for
PutObject.

    std::cout << "The default checksum algorithm for PutObject is "
                << stringForHashMethod(putObjectHashMethod)
                << std::endl;
}

// Demonstrate in code how the hash is computed.
if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
    std::cerr << "Error calculating hash for file " << TEST_FILE <<
std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}
Aws::String key = stringForHashMethod(putObjectHashMethod);
key += "_";
key += TEST_FILE_KEY;
Aws::String localHash = hasher.getBase64HashString();

// Upload the object with PutObject
if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                       inputData, chosenHashMethod == DEFAULT,
                       *client)) {
    std::cerr << "Error putting file " << TEST_FILE << " to bucket "
                << bucketName << " with key " << key << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}
```

```

    }

    Aws::String retrievedHash;
    if (!retrieveObjectHash(bucketName, key,
                           putObjectHashMethod, retrievedHash,
                           nullptr, *client)) {
        std::cerr << "Error getting file " << TEST_FILE << " from bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    explainPutObjectResults();
    verifyHashingResults(retrievedHash, hasher,
                        "PutObject upload", putObjectHashMethod);

    printAsterisksLine();
    pressEnterToContinue();

    key = "tr_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    introductoryTransferManagerUploadExplanations(key);

    HASH_METHOD transferManagerHashMethod = chosenHashMethod;
    if (transferManagerHashMethod == DEFAULT) {
        transferManagerHashMethod = CRC32; // The default hash method for the
        TransferManager is CRC32.

        std::cout << "The default checksum algorithm for TransferManager is "
                  << stringForHashMethod(transferManagerHashMethod)
                  << std::endl;
    }

    // Upload the large file using the transfer manager.
    if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
                                chosenHashMethod == DEFAULT,
                                client)) {
        std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
        std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }
}

```



```
std::vector<Aws::String> retrievedTransferManagerPartHashes;
Aws::String retrievedTransferManagerFinalHash;

// Retrieve all the hashes for the TransferManager upload.
if (!retrieveObjectHash(bucketName, key,
                        transferManagerHashMethod,
                        retrievedTransferManagerFinalHash,
                        &retrievedTransferManagerPartHashes, *client)) {
    std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

AwsDoc::S3::Hasher locallyCalculatedFinalHash;
std::vector<Aws::String> locallyCalculatedPartHashes;

// Calculate the hashes locally to demonstrate how TransferManager hashes are
computed.
if (!calculatePartHashesForFile(transferManagerHashMethod,
MULTI_PART_TEST_FILE,
                                UPLOAD_BUFFER_SIZE,
                                locallyCalculatedFinalHash,
                                locallyCalculatedPartHashes)) {
    std::cerr << "Exiting because of an error in calculatePartHashesForFile."
<< std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

verifyHashingResults(retrievedTransferManagerFinalHash,
                    locallyCalculatedFinalHash, "TransferManager upload",
                    transferManagerHashMethod,
                    retrievedTransferManagerPartHashes,
                    locallyCalculatedPartHashes);

printAsterisksLine();

key = "mp_";
key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

multiPartUploadExplanations(key, chosenHashMethod);
```

```
pressEnterToContinue();

std::shared_ptr<Aws::IOStream> largeFileInputData =
    Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                  MULTI_PART_TEST_FILE,
                                  std::ios_base::in |
                                  std::ios_base::binary);

if (!largeFileInputData->good()) {
    std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

HASH_METHOD multipartUploadHashMethod = chosenHashMethod;
if (multipartUploadHashMethod == DEFAULT) {
    multipartUploadHashMethod = MD5; // The default hash method for
    multipart uploads is MD5.

    std::cout << "The default checksum algorithm for multipart upload is "
                << stringForHashMethod(putObjectHashMethod)
                << std::endl;
}

AwsDoc::S3::Hasher hashData;
std::vector<Aws::String> partHashes;

if (!doMultipartUpload(bucketName, key,
                       multipartUploadHashMethod,
                       largeFileInputData, chosenHashMethod == DEFAULT,
                       hashData,
                       partHashes,
                       *client)) {
    std::cerr << "Exiting because of an error in doMultipartUpload." <<
    std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

std::cout << "Finished multipart upload of with hash method " <<
           stringForHashMethod(multipartUploadHashMethod) << std::endl;

std::cout << "Now we will retrieve the checksums from the server." <<
std::endl;
```

```

    retrievedHash.clear();
    std::vector<Aws::String> retrievedPartHashes;
    if (!retrieveObjectHash(bucketName, key,
                           multipartUploadHashMethod,
                           retrievedHash, &retrievedPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
multipart." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
                        multipartUploadHashMethod,
                        retrievedPartHashes, partHashes);

    printAsterisksLine();

    if (askYesNoQuestion("Would you like to delete the resources created in this
workflow? (y/n)")) {
        return cleanUp(bucketName, clientConfiguration);
    } else {
        std::cout << "The bucket " << bucketName << " was not deleted." <<
std::endl;
        return true;
    }
}

//! Routine which uploads an object to an S3 bucket with different object
integrity hashing methods.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashData: The hash value that will be associated with the uploaded
object.
    \param hashMethod: The hashing algorithm to use when calculating the hash
value.
    \param body: The data content of the object being uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default hash
method or the one specified in the hashMethod parameter.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

```

```

bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
    &key,
                                const Aws::String &hashData,
                                AwsDoc::S3::HASH_METHOD hashMethod,
                                const std::shared_ptr<Aws::IOStream> &body,
                                bool useDefaultHashMethod,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {
            request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                request.SetContentMD5(hashData);
                break;
            case AwsDoc::S3::SHA1:
                request.SetChecksumSHA1(hashData);
                break;
            case AwsDoc::S3::SHA256:
                request.SetChecksumSHA256(hashData);
                break;
            case AwsDoc::S3::CRC32:
                request.SetChecksumCRC32(hashData);
                break;
            case AwsDoc::S3::CRC32C:
                request.SetChecksumCRC32C(hashData);
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    }
    request.SetBody(body);
    Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
    body->seekg(0, body->beg);
    if (outcome.IsSuccess()) {
        std::cout << "Object successfully uploaded." << std::endl;
    }
}

```

```

    } else {
        std::cerr << "Error uploading object." <<
            outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of
the object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
&key,
                                     AwsDoc::S3::HASH_METHOD hashMethod,
                                     Aws::String &hashData,
                                     std::vector<Aws::String> *partHashes,
                                     const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                outcome.GetError().GetMessage() << std::endl;

```

```

        return false;
    }
} else { // hashMethod != MD5
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
            case AwsDoc::S3::MD5:
                break; // MD5 is not supported.
#pragma clang diagnostic pop
            case AwsDoc::S3::SHA1:
                hashData = result.GetChecksum().GetChecksumSHA1();
                break;
            case AwsDoc::S3::SHA256:
                hashData = result.GetChecksum().GetChecksumSHA256();
                break;
            case AwsDoc::S3::CRC32:
                hashData = result.GetChecksum().GetChecksumCRC32();
                break;
            case AwsDoc::S3::CRC32C:
                hashData = result.GetChecksum().GetChecksumCRC32C();
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    } else {
        std::cerr << "Error retrieving object checksum attributes." <<
outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    if (nullptr != partHashes) {

```

```

        attributes.clear();
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
        request.SetObjectAttributes(attributes);
        outcome = client.GetObjectAttributes(request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
            const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
            for (const Aws::S3::Model::ObjectPart &part: parts) {
                switch (hashMethod) {
                    case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                        break;
                    case AwsDoc::S3::MD5: // MD5 is not supported.
                        break;
                    case AwsDoc::S3::SHA1:
                        partHashes->push_back(part.GetChecksumSHA1());
                        break;
                    case AwsDoc::S3::SHA256:
                        partHashes->push_back(part.GetChecksumSHA256());
                        break;
                    case AwsDoc::S3::CRC32:
                        partHashes->push_back(part.GetChecksumCRC32());
                        break;
                    case AwsDoc::S3::CRC32C:
                        partHashes->push_back(part.GetChecksumCRC32C());
                        break;
                    default:
                        std::cerr << "Unknown hash method." << std::endl;
                        return false;
                }
            }
        } else {
            std::cerr << "Error retrieving object attributes for object
parts." <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
}

return true;
}

```

```

//! Verifies the hashing results between the retrieved and local hashes.
/*!
\param retrievedHash The hash value retrieved from the remote source.
\param localHash The hash value calculated locally.
\param uploadtype The type of upload (e.g., "multipart", "single-part").
\param hashMethod The hashing method used (e.g., MD5, SHA-256).
\param retrievedPartHashes (Optional) The list of hashes for the individual
parts retrieved from the remote source.
\param localPartHashes (Optional) The list of hashes for the individual parts
calculated locally.
*/
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                     const Hasher &localHash,
                                     const Aws::String &uploadtype,
                                     HASH_METHOD hashMethod,
                                     const std::vector<Aws::String>
&retrievedPartHashes,
                                     const std::vector<Aws::String>
&localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash
<< std::endl;
    if (!retrievedPartHashes.empty()) {
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
retrieved."
<< std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << " Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
    if (hashMethod == MD5) {
        hashString = localHash.getHexHashString();
        if (!localPartHashes.empty()) {
            hashString += "-" + std::to_string(localPartHashes.size());
        }
    } else {
        hashString = localHash.getBase64HashString();
    }

    bool allMatch = true;
    if (hashString != retrievedHash) {
        std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
std::endl;
    }
}

```



```

        std::cerr << "Local hash- '" << hashString << "'" << std::endl;
        std::cerr << "Remote hash - '" << retrievedHash << "'" << std::endl;
        allMatch = false;
    }

    if (hashMethod != MD5) {
        if (localPartHashes.size() != retrievedPartHashes.size()) {
            std::cerr << "For " << uploadtype << ", the number of part hashes do
not match" << std::endl;
            std::cerr << "Local number of hashes- '" << localPartHashes.size() <<
""
                << std::endl;
            std::cerr << "Remote number of hashes - '"
                << retrievedPartHashes.size()
                << "'" << std::endl;
        }

        for (int i = 0; i < localPartHashes.size(); ++i) {
            if (localPartHashes[i] != retrievedPartHashes[i]) {
                std::cerr << "For " << uploadtype << ", the part hashes do not
match for part " << i + 1
                    << "." << std::endl;
                std::cerr << "Local hash- '" << localPartHashes[i] << "'"
                    << std::endl;
                std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                    << std::endl;
                allMatch = false;
            }
        }
    }

    if (allMatch) {
        std::cout << "For " << uploadtype << ", locally and remotely calculated
hashes all match!" << std::endl;
    }
}

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
    Aws::Transfer::TransferHandle> &,
                                         const
    Aws::Client::AWSError<Aws::S3::S3Errors> &err) {

```

```

        std::cerr << "Error during transfer: '" << err.GetMessage() << "'" <<
        std::endl;
    }

    static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                              const std::shared_ptr<const
        Aws::Transfer::TransferHandle> &handle) {
        if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
            std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
            std::endl;
        }
    }
}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's
    Transfer Manager.
    /*!
        \param bucket: The name of the S3 bucket where the object will be uploaded.
        \param key: The unique identifier (key) for the object within the S3 bucket.
        \param hashMethod: The hashing algorithm to use when calculating the hash
        value.
        \param useDefaultHashMethod: A flag indicating whether to use the default hash
        method or the one specified in the hashMethod parameter.
        \param client: The S3 client instance used to perform the upload operation.
        \return bool: Function succeeded.
    */
    bool
    AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
        &key,

                                     AwsDoc::S3::HASH_METHOD hashMethod,
                                     bool useDefaultHashMethod,
                                     const std::shared_ptr<Aws::S3::S3Client>

        &client) {
        std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
        Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
            "executor", 25);
        Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
        transfer_config.s3Client = client;
        transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
        if (!useDefaultHashMethod) {
            if (hashMethod == MD5) {
                transfer_config.computeContentMD5 = true;
            } else {
                transfer_config.checksumAlgorithm =
                getChecksumAlgorithmForHashMethod(

```

```

        hashMethod);
    }
}
transfer_config.errorCallback = transferManagerErrorCallback;
transfer_config.transferStatusUpdatedCallback =
transferManagerStatusCallback;

    std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
    Aws::Transfer::TransferManager::Create(
        transfer_config);

    std::cout << "Uploading the file..." << std::endl;
    std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle =
    transfer_manager->UploadFile(MULTI_PART_TEST_FILE,

        bucket, key,

        "text/plain",

        Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success =
        uploadHandle->GetStatus() ==
    Aws::Transfer::TransferStatus::COMPLETED;
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle-
    >GetLastError();
        std::cerr << "File upload failed:  " << err.GetMessage() << std::endl;
    }

    return success;
}

//! Routine which calculates the hash values for each part of a file being
uploaded to an S3 bucket.
/*!
    \param hashMethod: The hashing algorithm to use when calculating the hash
values.
    \param fileName: The path to the file for which the part hashes will be
calculated.
    \param bufferSize: The size of the buffer to use when reading the file.
    \param[out] hashDataResult: The Hasher object that will store the concatenated
hash value.

```

```

    \param[out] partHashes: The vector that will store the calculated hash values
    for each part of the file.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                             const Aws::String &fileName,
                                             size_t bufferSize,
                                             AwsDoc::S3::Hasher &hashDataResult,
                                             std::vector<Aws::String> &partHashes)
{
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
    size_t uploadedBytes = 0;

    while (uploadedBytes < objectSize) {
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(), objectSize -
uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
            Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
            &preallocatedStreamBuf);

        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),

```

```

        hashBuffer.GetUnderlyingData() +
        hashBuffer.GetLength());

        uploadedBytes += bytesToRead;
    }

    return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
&key,

                                Aws::S3::Model::ChecksumAlgorithm
checksumAlgorithm,

                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
        client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}

```

```

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
checksum algorithm,
                                ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
    &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
    &uploadID,
                                                         int partNumber,
    Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                         const Aws::String
    &calculatedHash,
                                                         const
    std::shared_ptr<Aws::IOStream> &body,
                                                         const Aws::S3::S3Client
    &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;

```

```

        case Aws::S3::Model::ChecksumAlgorithm::CRC32:
            request.SetChecksumCRC32(calculatedHash);
            break;
        case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
            request.SetChecksumCRC32C(calculatedHash);
            break;
        case Aws::S3::Model::ChecksumAlgorithm::SHA1:
            request.SetChecksumSHA1(calculatedHash);
            break;
        case Aws::S3::Model::ChecksumAlgorithm::SHA256:
            request.SetChecksumSHA256(calculatedHash);
            break;
    }
}

return client.UploadPart(request);
}

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
                                       const Aws::String &uploadID,
                                       const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
        client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

    }

    return outcome.IsSuccess();
}

//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

    const Aws::String &key,

    const Aws::String &uploadID,

    const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

    const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
        client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

//! Routine which performs a multi-part upload.

```



```

/ * !
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash
value.
    \param ioStream: An IOStream for the data to be uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default
hash method or the one specified in the hashMethod parameter.
    \param[out] hashDataResult: The Hasher object that will store the
concatenated hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
for each part of the file.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                   const Aws::String &key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream>
&ioStream,
                                   bool useDefaultHashMethod,
                                   AwsDoc::S3::Hasher &hashDataResult,
                                   std::vector<Aws::String> &partHashes,
                                   const Aws::S3::S3Client &client) {

    // Get object size.
    ioStream->seekg(0, ioStream->end);
    size_t objectSize = ioStream->tellg();
    ioStream->seekg(0, ioStream->beg);

    Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
    Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {
            checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
        }
    }
    Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
    client);
    if (uploadID.empty()) {
        return false;
    }

    std::vector<unsigned char> totalHashBuffer;
    bool uploadSucceeded = true;

```

```
std::streamsize uploadedBytes = 0;
int partNumber = 1;
Aws::Vector<Aws::S3::Model::CompletedPart> parts;
while (uploadedBytes < objectSize) {
    std::cout << "Uploading part " << partNumber << "." << std::endl;

    std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
    std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),
objectSize - uploadedBytes));
    ioStream->read((char *) buffer.data(), bytesToRead);
    Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),
bytesToRead);
    std::shared_ptr<Aws::IOStream> body =
        Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
            &preallocatedStreamBuf);

    Hasher hasher;
    if (!hasher.calculateObjectHash(*body, hashMethod)) {
        std::cerr << "Error calculating hash." << std::endl;
        uploadSucceeded = false;
        break;
    }

    Aws::String base64HashString = hasher.getBase64HashString();
    partHashes.push_back(base64HashString);

    Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

    totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
                        hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

    Aws::String calculatedHash;
    if (gUseCalculatedChecksum) {
        calculatedHash = base64HashString;
    }
    Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,
```

```

checksumAlgorithm, base64HashString, body,
                                                                    client);

    if (uploadPartOutcome.IsSuccess()) {
        const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
        Aws::S3::Model::CompletedPart completedPart;
        completedPart.SetETag(uploadPartResult.GetETag());
        completedPart.SetPartNumber(partNumber);
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                break; // Do nothing.
            case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                break;
            case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                break;
            case AwsDoc::S3::CRC32:

completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                break;
            case AwsDoc::S3::CRC32C:

completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                break;
            default:
                std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                break;
        }

        parts.push_back(completedPart);
    } else {
        std::cerr << "Error uploading part. " <<
            uploadPartOutcome.GetError().GetMessage() << std::endl;
        uploadSucceeded = false;
        break;
    }

    uploadedBytes += bytesToRead;
    partNumber++;

```

```

    }

    if (!uploadSucceeded) {
        abortMultipartUpload(bucket, key, uploadID, client);
        return false;
    } else {

        Aws::S3::Model::CompleteMultipartUploadOutcome
        completeMultipartUploadOutcome = completeMultipartUpload(bucket,

                                key,

                                uploadID,

                                parts,

                                client);

        if (completeMultipartUploadOutcome.IsSuccess()) {
            std::cout << "Multipart upload completed." << std::endl;
            if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod))
        {
                std::cerr << "Error calculating hash." << std::endl;
                return false;
            }
        } else {
            std::cerr << "Error completing multipart upload." <<
                completeMultipartUploadOutcome.GetError().GetMessage()
                << std::endl;
        }

        return completeMultipartUploadOutcome.IsSuccess();
    }
}

//! Routine which retrieves the string for a HASH_METHOD constant.
/*!
    \param hashMethod: A HASH_METHOD constant.
    \return String: A string description of the hash method.
*/
Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            return "Default";
    }
}

```

```

        case AwsDoc::S3::MD5:
            return "MD5";
        case AwsDoc::S3::SHA1:
            return "SHA1";
        case AwsDoc::S3::SHA256:
            return "SHA256";
        case AwsDoc::S3::CRC32:
            return "CRC32";
        case AwsDoc::S3::CRC32C:
            return "CRC32C";
        default:
            return "Unknown";
    }
}

//! Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
/*!
    \param hashMethod: A HASH_METHOD constant.
    \return ChecksumAlgorithm: The ChecksumAlgorithm enum.
*/
Aws::S3::Model::ChecksumAlgorithm
AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod)
{
    Aws::S3::Model::ChecksumAlgorithm result =
    Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not
valid." << std::endl;
            break; // Default is not supported.
        case AwsDoc::S3::MD5:
            break; // Ignore MD5.
        case AwsDoc::S3::SHA1:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA1;
            break;
        case AwsDoc::S3::SHA256:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA256;
            break;
        case AwsDoc::S3::CRC32:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
            break;
        case AwsDoc::S3::CRC32C:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
            break;
    }
}

```

```

        default:
            std::cerr << "Unknown hash method." << std::endl;
            break;

    }

    return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                        const Aws::S3::S3ClientConfiguration
                        &clientConfiguration) {

    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                                clientConfiguration);
        }
    } else {
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!
    \param bucketName: The name of the S3 bucket to use.
*/
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout
        << "Welcome to the Amazon Simple Storage Service (Amazon S3) object
integrity workflow."
        << std::endl;
    printAsterisksLine();
}

```

```

    std::cout
        << "This workflow demonstrates how Amazon S3 uses checksum values to
verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
        << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
        << "By default it calculates a checksum that is uploaded with an
object.\n"
        << "The default checksum algorithm for PutObject and MultiPart upload
is an MD5 hash.\n"
        << "The default checksum algorithm for TransferManager uploads is a
CRC32 checksum."
        << std::endl;
    std::cout
        << "You can override the default behavior, requiring one of the
following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting
the SDK calculate the value."
        << std::endl;
    std::cout
        << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
        << std::endl;

    std::cout
        << "This workflow will locally compute checksums for files uploaded
to an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
        << "This is done to provide demonstration code for how the checksums
are calculated."
        << std::endl;
    std::cout << "A bucket named '" << bucketName << "' will be created for the
object uploads."
        << std::endl;
}

//! Console interaction which explains the PutObject results.
*/
void AwsDoc::S3::explainPutObjectResults() {

```

```
std::cout << "The upload was successful.\n";
std::cout << "If the checksums had not matched, the upload would have
failed."
    << std::endl;
std::cout
    << "The checksums calculated by the server have been retrieved using
the GetObjectAttributes."
    << std::endl;
std::cout
    << "The locally calculated checksums have been verified against the
retrieved checksums."
    << std::endl;
}

//! Console interaction explaining transfer manager uploads.
/*!
\param objectKey: The key for the object being uploaded.
*/
void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
    const Aws::String &objectKey) {
    std::cout
        << "Now the workflow will demonstrate object integrity for
TransferManager multi-part uploads."
        << std::endl;
    std::cout
        << "The AWS C++ SDK has a TransferManager class which simplifies
multipart uploads."
        << std::endl;
    std::cout
        << "The following code lets the TransferManager handle much of the
checksum configuration."
        << std::endl;

    std::cout << "An object with the key '" << objectKey
        << " will be uploaded by the TransferManager using a "
        << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout << "For TransferManager uploads, this demo always lets the SDK
calculate the hash value."
            << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();
}
```



```

}

//! Console interaction explaining multi-part uploads.
/*!
    \param objectKey: The key for the object being uploaded.
    \param chosenHashMethod: The hash method selected by the user.
*/
void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                              HASH_METHOD chosenHashMethod) {
    std::cout
        << "Now we will provide an in-depth demonstration of multi-part
uploading by calling the multi-part upload APIs directly."
        << std::endl;
    std::cout << "These are the same APIs used by the TransferManager when
uploading large files."
        << std::endl;
    std::cout
        << "In the following code, the checksums are also calculated locally
and then compared."
        << std::endl;
    std::cout
        << "For multi-part uploads, a checksum is uploaded with each part.
The final checksum is a concatenation of"
        << std::endl;
    std::cout << "the checksums for each part." << std::endl;
    std::cout
        << "This is explained in the user guide, https://docs.aws.amazon.com/
AmazonS3/latest/userguide/checking-object-integrity.html,"
        << " in the section \"Using part-level checksums for multipart
uploads\"." << std::endl;

    std::cout << "Starting multipart upload of with hash method " <<
        stringForHashMethod(chosenHashMethod) << " uploading to with object
key\n"
        << "'" << objectKey << "'," << std::endl;
}

//! Create a large file for doing multi-part uploads.
/*!
*/
bool AwsDoc::S3::createLargeFileIfNotExists() {
    // Generate a large file by writing this source file multiple times to a new
    file.

```

```
    if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
        return true;
    }

    std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out
                           | std::ios::binary);

    if (!newFile) {
        std::cerr << "createLargeFileIfNotExists- Error creating file " <<
MULTI_PART_TEST_FILE <<
        std::endl;
        return false;
    }

    std::ifstream input(TEST_FILE, std::ios::in
                        | std::ios::binary);

    if (!input) {
        std::cerr << "Error opening file " << TEST_FILE <<
        std::endl;
        return false;
    }
    std::stringstream buffer;
    buffer << input.rdbuf();

    input.close();

    while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
        buffer.seekg(std::stringstream::beg);
        newFile << buffer.rdbuf();
    }

    newFile.close();

    return true;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - [AbortMultipartUpload](#)

- [CompleteMultipartUpload](#)
- [CreateMultipartUpload](#)
- [DeleteObject](#)
- [GetObjectAttributes](#)
- [PutObject](#)
- [UploadPart](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して、Amazon S3 のバージョン管理されたオブジェクトを操作する  
次のコード例は、以下の操作方法を示しています。

- バージョニングされた S3 バケットを作成します。
- オブジェクトのすべてのバージョンを取得します。
- オブジェクトを以前のバージョンにロールバックします。
- バージョニングされたオブジェクトを削除して復元します。
- オブジェクトのバージョンを完全に削除します。

## Python

### SDK for Python (Boto3)

#### Note

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

S3 アクションをラップする関数を作成します。

```
def create_versioned_bucket(bucket_name, prefix):  
    """  
    Creates an Amazon S3 bucket, enables it for versioning, and configures a  
    lifecycle
```

that expires noncurrent object versions after 7 days.

Adding a lifecycle configuration to a versioned bucket is a best practice. It helps prevent objects in the bucket from accumulating a large number of noncurrent versions, which can slow down request performance.

Usage is shown in the `usage_demo_single_object` function at the end of this module.

```
:param bucket_name: The name of the bucket to create.
:param prefix: Identifies which objects are automatically expired under the
               configured lifecycle rules.
:return: The newly created bucket.
"""
try:
    bucket = s3.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
    logger.info("Created bucket %s.", bucket.name)
except ClientError as error:
    if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
        logger.warning("Bucket %s already exists! Using it.", bucket_name)
        bucket = s3.Bucket(bucket_name)
    else:
        logger.exception("Couldn't create bucket %s.", bucket_name)
        raise

try:
    bucket.Versioning().enable()
    logger.info("Enabled versioning on bucket %s.", bucket.name)
except ClientError:
    logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
    raise

try:
    expiration = 7
    bucket.LifecycleConfiguration().put(
        LifecycleConfiguration={
            "Rules": [
                {
                    "Status": "Enabled",
```

```

        "Prefix": prefix,
        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
    }
]
}
)
logger.info(
    "Configured lifecycle to expire noncurrent versions after %s days "
    "on bucket %s.",
    expiration,
    bucket.name,
)
except ClientError as error:
    logger.warning(
        "Couldn't configure lifecycle on bucket %s because %s. "
        "Continuing anyway.",
        bucket.name,
        error,
    )

return bucket

def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

```

```

logger.debug(
    "Got versions:\n%s",
    "\n".join(
        [
            f"\t{version.version_id}, last modified {version.last_modified}"
            for version in versions
        ]
    ),
)

if version_id in [ver.version_id for ver in versions]:
    print(f"Rolling back to version {version_id}")
    for version in versions:
        if version.version_id != version_id:
            version.delete()
            print(f"Deleted version {version.version_id}")
        else:
            break

    print(f"Active version is now {bucket.Object(object_key).version_id}")
else:
    raise KeyError(
        f"{version_id} was not found in the list of versions for "
        f"{object_key}."
    )

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest
    version
    and the object then presents as not deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.

```

```

"""
# Get the latest version for the object.
response = s3.meta.client.list_object_versions(
    Bucket=bucket.name, Prefix=object_key, MaxKeys=1
)

if "DeleteMarkers" in response:
    latest_version = response["DeleteMarkers"][0]
    if latest_version["IsLatest"]:
        logger.info(
            "Object %s was indeed deleted on %s. Let's revive it.",
            object_key,
            latest_version["LastModified"],
        )
        obj = bucket.Object(object_key)
        obj.Version(latest_version["VersionId"]).delete()
        logger.info(
            "Revived %s, active version is now %s with body '%s'",
            object_key,
            obj.version_id,
            obj.get()["Body"].read(),
        )
    else:
        logger.warning(
            "Delete marker is not the latest version for %s!", object_key
        )
elif "Versions" in response:
    logger.warning("Got an active version for %s, nothing to do.",
object_key)
else:
    logger.error("Couldn't get any version info for %s.", object_key)

def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """

```

```
try:
    bucket.object_versions.filter(Prefix=object_key).delete()
    logger.info("Permanently deleted all versions of object %s.", object_key)
except ClientError:
    logger.exception("Couldn't delete all versions of %s.", object_key)
    raise
```

詩のスタンザをバージョニングされたオブジェクトにアップロードし、一連のアクションを実行します。

```
def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
    obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
```



```
obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
print(
    "The latest version of the stanza is now:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Versions are returned in order, most recent first.
obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
print(
    "The version data of the stanza revisions:",
    *[
        f"    {version.version_id}, last modified {version.last_modified}"
        for version in obj_stanza_1_versions
    ],
    sep="\n",
)

# Rollback two versions.
print("\nRolling back two versions...")
rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
print(
    "The latest version of the stanza:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Delete the stanza
print("\nDeleting the stanza...")
obj_stanza_1.delete()
try:
    obj_stanza_1.get()
except ClientError as error:
    if error.response["Error"]["Code"] == "NoSuchKey":
        print("The stanza is now deleted (as expected).")
    else:
        raise

# Revive the stanza
print("\nRestoring the stanza...")
revive_object(bucket, obj_stanza_1.key)
print(
```

```
        "The stanza is restored! The latest version is again:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
    else:
        print("Something went wrong. The stanza still exists!")

    print(f"\nRemoving {bucket.name}...")
    bucket.delete()
    print(f"{bucket.name} deleted.")
    print("Demo done!")
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の以下のトピックを参照してください。
  - [CreateBucket](#)
  - [DeleteObject](#)
  - [ListObjectVersions](#)
  - [PutBucketLifecycleConfiguration](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## Amazon S3 のサーバーレス例

次のコード例は、AWS SDK で Amazon S3 を使用方法を示します。

例

- [Amazon S3 トリガーから Lambda 関数を呼び出す](#)

## Amazon S3 トリガーから Lambda 関数を呼び出す

次のコード例は、S3 バケットにオブジェクトをアップロードすることによってトリガーされるイベントを受け取る Lambda 関数を実装する方法を示しています。この関数は、イベントパラメータから S3 バケット名とオブジェクトキーを取得し、Amazon S3 API を呼び出してオブジェクトのコンテンツタイプを取得してログに記録します。

.NET

SDK for .NET

### Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

.NET を使用して Lambda で S3 イベントを消費します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
// into a .NET class.
[assembly:
    LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }
    }
}
```

```
internal Function(AmazonS3Client s3Client)
{
    _s3Client = s3Client ?? new AmazonS3Client();
}

public async Task<string> Handler(S3Event evt, ILambdaContext context)
{
    try
    {
        if (evt.Records.Count <= 0)
        {
            context.Logger.LogLine("Empty S3 Event received");
            return string.Empty;
        }

        var bucket = evt.Records[0].S3.Bucket.Name;
        var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

        context.Logger.LogLine($"Request is for {bucket} and {key}");

        var objectResult = await _s3Client.GetObjectAsync(bucket, key);

        context.Logger.LogLine($"Returning {objectResult.Key}");

        return objectResult.Key;
    }
    catch (Exception e)
    {
        context.Logger.LogLine($"Error processing request - {e.Message}");

        return string.Empty;
    }
}
}
```

## Go

## SDK for Go V2

 Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Go を使用して Lambda で S3 イベントを消費します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "log"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Printf("failed to load default config: %s", err)
        return err
    }
    s3Client := s3.NewFromConfig(sdkConfig)

    for _, record := range s3Event.Records {
        bucket := record.S3.Bucket.Name
        key := record.S3.Object.URLDecodedKey
        headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
            Bucket: &bucket,
            Key:    &key,
        })
        if err != nil {
            log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
        }
    }
}
```

```
    return err
}
log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
*headOutput.ContentType)
}

return nil
}

func main() {
    lambda.Start(handler)
}
```

## Java

### SDK for Java 2.x

#### Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Java を使用した Lambda での S3 イベントの消費。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucket)
            .key(key)
            .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

## JavaScript

### SDK for JavaScript (v3)

#### Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

## JavaScript を使用した Lambda での S3 イベントの消費。

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

## TypeScript を使用した Lambda での S3 イベントの消費。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
    // Get the object from the event and show its content type
```



```
const bucket = event.Records[0].s3.bucket.name;
const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
const params = {
  Bucket: bucket,
  Key: key,
};
try {
  const { ContentType } = await s3.send(new HeadObjectCommand(params));
  console.log('CONTENT TYPE:', ContentType);
  return ContentType;
} catch (err) {
  console.log(err);
  const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your bucket is in the same region as this function.`;
  console.log(message);
  throw new Error(message);
}
};
```

## PHP

### SDK for PHP

#### Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を見つけて、設定と実行の方法を確認してください。

PHP を使用して Lambda で S3 イベントの消費。

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

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

```
class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
                $bucket . '. Make sure they exist and your bucket is in the same region as this
                function.' . "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

## Python

### SDK for Python (Boto3)

#### Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Python を使用して Lambda で S3 イベントを消費します。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')

def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
    encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
        your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

## Ruby

### SDK for Ruby

#### Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を見つけて、設定と実行の方法を確認してください。

Ruby を使用して Lambda での S3 イベントの消費。

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
    Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
    and your bucket is in the same region as this function."
    raise e
  end
end
```

## Rust

### SDK for Rust

#### Note

GitHub には、その他のリソースもあります。[サーバーレスサンプル](#)リポジトリで完全な例を検索し、設定および実行の方法を確認してください。

Rust を使用して Lambda で S3 イベントを消費します。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
```

```
tracing::info!(records = ?evt.payload.records.len(), "Received request from
SQS");

if evt.payload.records.len() == 0 {
    tracing::info!("Empty S3 event received");
}

let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
name to exist");
let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
exist");

tracing::info!("Request is for {} and object {}", bucket, key);

let s3_get_object_result = s3_client
    .get_object()
    .bucket(bucket)
    .key(key)
    .send()
    .await;

match s3_get_object_result {
    Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
contains a 'body' property of type ByteStream"),
    Err(_) => tracing::info!("Failure with S3 Get Object request")
}

Ok(())
}
```

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用した Amazon S3 Control のコード例

次のコード例は、AWS Software Development Kit (SDK) で Amazon S3 Control を使用する方法を示しています。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## Amazon S3 Control

- [AWS SDK を使用した Amazon S3 Control の基本的な例](#)
  - [Hello Amazon S3 Control](#)
  - [AWS SDK を使用して Amazon S3 Control の基本を学ぶ](#)
  - [AWS SDK を使用した Amazon S3 Control のアクション](#)
    - [AWS SDK または CLI で CreateJob を使用する](#)
    - [AWS SDK で DeleteJobTagging を使用する](#)
    - [AWS SDK または CLI で DescribeJob を使用する](#)
    - [AWS SDK で GetJobTagging を使用する](#)
    - [AWS SDK で PutJobTagging を使用する](#)
    - [AWS SDK または CLI で UpdateJobPriority を使用する](#)
    - [AWS SDK または CLI で UpdateJobStatus を使用する](#)

## AWS SDK を使用した Amazon S3 Control の基本的な例

次のコード例は、AWS SDK で Amazon S3 Control を使用する基本的な方法を説明しています。

### 例

- [Hello Amazon S3 Control](#)
- [AWS SDK を使用して Amazon S3 Control の基本を学ぶ](#)
- [AWS SDK を使用した Amazon S3 Control のアクション](#)
  - [AWS SDK または CLI で CreateJob を使用する](#)
  - [AWS SDK で DeleteJobTagging を使用する](#)
  - [AWS SDK または CLI で DescribeJob を使用する](#)
  - [AWS SDK で GetJobTagging を使用する](#)
  - [AWS SDK で PutJobTagging を使用する](#)

- [AWS SDK または CLI で UpdateJobPriority を使用する](#)
- [AWS SDK または CLI で UpdateJobStatus を使用する](#)

## Hello Amazon S3 Control

次のコード例は、Amazon S3 Control の使用を開始する方法を示しています。

Java

SDK for Java 2.x

### Note

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

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this example:
 * <p/>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
```



```
* authentication for SDKs and tools, see https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs and Tools Reference Guide.
* <p/>
* You must have a runtime environment configured with the Java SDK.
* See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html
in the Developer Guide if this is not set up.
*/
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;

    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
ex.getMessage());
                    return null;
                })
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
configuration:
     * <ul>
     * <li>Maximum concurrency: 100</li>
     * <li>Connection timeout: 60 seconds</li>
     * <li>Read timeout: 60 seconds</li>
     * <li>Write timeout: 60 seconds</li>
     * <li>API call timeout: 2 minutes</li>
     * <li>API call attempt timeout: 90 seconds</li>
     * <li>Retry policy: 3 retries</li>
     * <li>Region: US_EAST_1</li>
     * </ul>
     */
}
```

```

    * <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
    * </ul>
    *
    * @return the asynchronous S3 Control client instance
    */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    /**
     * Asynchronously lists batch jobs that have completed for the specified
     account.
     *
     * @param accountId the ID of the account to list jobs for
     * @return a CompletableFuture that completes when the job listing operation
     is finished
     */
    public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
        ListJobsRequest jobsRequest = ListJobsRequest.builder()
            .jobStatuses(JobStatus.COMPLETE)
            .accountId(accountId)
            .maxResults(10)

```

```
        .build();

        ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
        return publisher.subscribe(response -> {
            List<JobListDescriptor> jobs = response.jobs();
            for (JobListDescriptor job : jobs) {
                System.out.println("The job id is " + job.jobId());
                System.out.println("The job priority is " + job.priority());
            }
        }).thenAccept(response -> {
            System.out.println("Listing batch jobs completed");
        }).exceptionally(ex -> {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
            throw new RuntimeException(ex);
        });
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[ListJobs](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```
def list_jobs(self, account_id: str) -> None:
    """
    List all batch jobs for the account.

    Args:
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.list_jobs(
```

```
        AccountId=account_id,
        JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
    )
    jobs = response.get('Jobs', [])
    for job in jobs:
        print(f"The job id is {job['JobId']}")
        print(f"The job priority is {job['Priority']}")
except ClientError as e:
    print(f"Error listing jobs: {e}")
    raise
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して Amazon S3 Control の基本を学ぶ

次のコード例は、Amazon S3 Control のコアオペレーションを学ぶ方法を示しています。

### Java

#### SDK for Java 2.x

#### Note

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

コアオペレーションについて学びます。

```
package com.example.s3.batch;

import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.IOException;
import java.util.Map;
```

```
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletionException;

public class S3BatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    private static final String STACK_NAME = "MyS3Stack";
    public static void main(String[] args) throws IOException {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        String uuid = java.util.UUID.randomUUID().toString();
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 Batch basics scenario.");
        System.out.println("""
            S3 Batch operations enables efficient and cost-effective processing
of large-scale
            data stored in Amazon S3. It automatically scales resources to handle
varying workloads
            without the need for manual intervention.

            One of the key features of S3 Batch is its ability to perform tagging
operations on objects stored in
            S3 buckets. Users can leverage S3 Batch to apply, update, or remove
tags on thousands or millions of
            objects in a single operation, streamlining the management and
organization of their data.

            This can be particularly useful for tasks such as cost allocation,
lifecycle management, or
            metadata-driven workflows, where consistent and accurate tagging is
essential.

            S3 Batch's scalability and serverless nature make it an ideal
solution for organizations with
            growing data volumes and complex data management requirements.

            This Java program walks you through Amazon S3 Batch operations.

            Let's get started...

            """);
```

```
        waitForInputToContinue(scanner);
        // Use CloudFormation to stand up the resource required for this
scenario.
        System.out.println("Use CloudFormation to stand up the resource required
for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);

        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
        String iamRoleArn = stackOutputs.get("S3BatchRoleArn");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setup the required bucket for this scenario.");
        waitForInputToContinue(scanner);
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
        System.out.println("This job tags all objects listed in the manifest file
with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
```

```
}

waitForInputToContinue(scanner);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Update an existing S3 Batch Operations job's
priority");
System.out.println("""
    In this step, we modify the job priority value. The higher the
number, the higher the priority.
    So, a job with a priority of `30` would have a higher priority than
a job with
    a priority of `20`. This is a common way to represent the priority
of a task
    or job, with higher numbers indicating a higher priority.

    Ensure that the job status allows for priority updates. Jobs in
certain
    states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
    updated. Only jobs in the Active or Suspended state typically allow
priority
    updates.
    """);

try {
    actions.updateJobPriorityAsync(jobId, accountId)
        .exceptionally(ex -> {
            System.err.println("Update job priority failed: " +
ex.getMessage());
            return null;
        })
        .join();
} catch (CompletionException ex) {
    System.err.println("Failed to update job priority: " +
ex.getMessage());
}
waitForInputToContinue(scanner);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Cancel the S3 Batch job");
System.out.print("Do you want to cancel the Batch job? (y/n): ");
```

```
String cancelAns = scanner.nextLine();
if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
    try {
        actions.cancelJobAsync(jobId, accountId)
            .exceptionally(ex -> {
                System.err.println("Cancel job failed: " +
ex.getMessage());
                return null;
            })
            .join();
    } catch (CompletionException ex) {
        System.err.println("Failed to cancel job: " + ex.getMessage());
    }
} else {
    System.out.println("Job " + jobId + " was not canceled.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Describe the job that was just created");
waitForInputToContinue(scanner);
try {
    actions.describeJobAsync(jobId, accountId)
        .exceptionally(ex -> {
            System.err.println("Describe job failed: " +
ex.getMessage());
            return null;
        })
        .join();
} catch (CompletionException ex) {
    System.err.println("Failed to describe job: " + ex.getMessage());
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Describe the tags associated with the job");
waitForInputToContinue(scanner);
try {
    actions.getJobTagsAsync(jobId, accountId)
        .exceptionally(ex -> {
            System.err.println("Get job tags failed: " +
ex.getMessage());
            return null;
        })
        .join();
}
```



```
        .join();
    } catch (CompletionException ex) {
        System.err.println("Failed to get job tags: " + ex.getMessage());
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Update Batch Job Tags");
    waitForInputToContinue(scanner);
    try {
        actions.putJobTaggingAsync(jobId, accountId)
            .exceptionally(ex -> {
                System.err.println("Put job tagging failed: " +
ex.getMessage());
                return null;
            })
            .join();
    } catch (CompletionException ex) {
        System.err.println("Failed to put job tagging: " + ex.getMessage());
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Delete the Amazon S3 Batch job tagging.");
    System.out.print("Do you want to delete Batch job tagging? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        try {
            actions.deleteBatchJobTagsAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
        }
    } else {
        System.out.println("Tagging was not deleted.");
    }
    System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
        String delResAns = scanner.nextLine();
        if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
            actions.deleteFilesFromBucket(bucketName, fileNames, actions);
            actions.deleteBucketFolderAsync(bucketName);
            actions.deleteBucket(bucketName)
                .thenRun(() -> System.out.println("Bucket deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
        } else {
            System.out.println("The AWS resources were not deleted.");
        }
        System.out.println("The Amazon S3 Batch scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println();
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println();
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

## オペレーションをラップするアクションクラス。

```
public class S3BatchActions {

    private static S3ControlAsyncClient asyncClient;

    private static S3AsyncClient s3AsyncClient ;

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
     S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
     configuration:
     * <ul>
     * <li>Maximum concurrency: 100</li>
     * <li>Connection timeout: 60 seconds</li>
     * <li>Read timeout: 60 seconds</li>
     * <li>Write timeout: 60 seconds</li>
     * <li>API call timeout: 2 minutes</li>
     * <li>API call attempt timeout: 90 seconds</li>
     * <li>Retry policy: 3 retries</li>
     * <li>Region: US_EAST_1</li>
     * <li>Credentials provider: {@link
     EnvironmentVariableCredentialsProvider}</li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
                ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryPolicy(RetryPolicy.builder()
                        .numRetries(3)
```

```
        .build())
        .build();

    asyncClient = S3ControlAsyncClient.builder()
        .region(Region.US_EAST_1)
        .httpClient(httpClient)
        .overrideConfiguration(overrideConfig)
        .build();
    }
    return asyncClient;
}

private static S3AsyncClient getS3AsyncClient() {
    if (asyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
            ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

        s3AsyncClient = S3AsyncClient.builder()
            .region(Region.US_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return s3AsyncClient;
}

/**
 * Cancels a job asynchronously.
 *
 * @param jobId The ID of the job to be canceled.
 * @param accountId The ID of the account associated with the job.
```

```

    * @return A {@link CompletableFuture} that completes when the job status has
    been updated to "CANCELLED".
    *      If an error occurs during the update, the returned future will
    complete exceptionally.
    */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
    {
        UpdateJobStatusRequest updateJobStatusRequest =
        UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
    * Updates the priority of a job asynchronously.
    *
    * @param jobId      the ID of the job to update
    * @param accountId the ID of the account associated with the job
    * @return a {@link CompletableFuture} that represents the asynchronous
    operation, which completes when the job priority has been updated or an error
    has occurred
    */
    public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
        UpdateJobPriorityRequest priorityRequest =
        UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
    }

```

```

        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }

    /**
     * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
     *
     * @param jobId      the ID of the job for which to retrieve the tags
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
     * @throws RuntimeException if an error occurs while retrieving the job tags
     */
    public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
        GetJobTaggingRequest request = GetJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return asyncClient.getJobTagging(request)
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
    }

```

```
        }
    })
    .exceptionally(ex -> {
        System.err.println("Failed to get job tags: " + ex.getMessage());
        throw new RuntimeException(ex); // Propagate the exception
    });
}

/**
 * Asynchronously deletes the tags associated with a specific batch job.
 *
 * @param jobId      The ID of the batch job whose tags should be deleted.
 * @param accountId  The ID of the account associated with the batch job.
 * @return A CompletableFuture that completes when the job tags have been
 * successfully deleted, or an exception is thrown if the deletion fails.
 */
public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
    DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .build();

    return asyncClient.deleteJobTagging(jobTaggingRequest)
        .thenAccept(response -> {
            System.out.println("You have successfully deleted " + jobId + "
tagging.");
        })
        .exceptionally(ex -> {
            System.err.println("Failed to delete job tags: " +
ex.getMessage());
            throw new RuntimeException(ex);
        });
}

/**
 * Asynchronously describes the specified job.
 *
 * @param jobId      the ID of the job to describe
 * @param accountId  the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job
 * description is available
 * @throws RuntimeException if an error occurs while describing the job

```

```
    */
    public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {
        DescribeJobRequest jobRequest = DescribeJobRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return getAsyncClient().describeJob(jobRequest)
            .thenAccept(response -> {
                System.out.println("Job ID: " + response.job().jobId());
                System.out.println("Description: " +
response.job().description());
                System.out.println("Status: " + response.job().statusAsString());
                System.out.println("Role ARN: " + response.job().roleArn());
                System.out.println("Priority: " + response.job().priority());
                System.out.println("Progress Summary: " +
response.job().progressSummary());

                // Print out details about the job manifest.
                JobManifest manifest = response.job().manifest();
                System.out.println("Manifest Location: " +
manifest.location().objectArn());
                System.out.println("Manifest ETag: " +
manifest.location().eTag());

                // Print out details about the job operation.
                JobOperation operation = response.job().operation();
                if (operation.s3PutObjectTagging() != null) {
                    System.out.println("Operation: S3 Put Object Tagging");
                    System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
                }

                // Print out details about the job report.
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " +
report.reportScopeAsString());
            })
            .exceptionally(ex -> {
```



```

        System.err.println("Failed to describe job: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
}

/**
 * Creates an asynchronous S3 job using the AWS Java SDK.
 *
 * @param accountId      the AWS account ID associated with the job
 * @param iamRoleArn      the ARN of the IAM role to be used for the job
 * @param manifestLocation the location of the job manifest file in S3
 * @param reportBucketName the name of the S3 bucket to store the job report
 * @param uuid            a unique identifier for the job
 * @return a CompletableFuture that represents the asynchronous creation of
the S3 job.
 *      The CompletableFuture will return the job ID if the job is created
successfully,
 *      or throw an exception if there is an error.
 */
public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
                                                String manifestLocation,
String reportBucketName, String uuid) {

    String[] bucketName = new String[]{" "};
    String[] parts = reportBucketName.split("::");
    if (parts.length > 1) {
        bucketName[0] = parts[1];
    } else {
        System.out.println("The input string does not contain the expected
format.");
    }

    return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
        .thenCompose(eTag -> {
            ArrayList<S3Tag> tagSet = new ArrayList<>();
            S3Tag s3Tag = S3Tag.builder()
                .key("keyOne")
                .value("ValueOne")
                .build();
            S3Tag s3Tag2 = S3Tag.builder()
                .key("keyTwo")
                .value("ValueTwo")

```

```
        .build();
        tagSet.add(s3Tag);
        tagSet.add(s3Tag2);

        S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
        .tagSet(tagSet)
        .build();

        JobOperation jobOperation = JobOperation.builder()
        .s3PutObjectTagging(objectTaggingOperation)
        .build();

        JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
        .objectArn(manifestLocation)
        .eTag(eTag)
        .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
        .fieldsWithStrings("Bucket", "Key")
        .format("S3BatchOperations_CSV_20180820")
        .build();

        JobManifest jobManifest = JobManifest.builder()
        .spec(manifestSpec)
        .location(jobManifestLocation)
        .build();

        JobReport jobReport = JobReport.builder()
        .bucket(reportBucketName)
        .prefix("reports")
        .format("Report_CSV_20180820")
        .enabled(true)
        .reportScope("AllTasks")
        .build();

        CreateJobRequest jobRequest = CreateJobRequest.builder()
        .accountId(accountId)
        .description("Job created using the AWS Java SDK")
        .manifest(jobManifest)
        .operation(jobOperation)
        .report(jobReport)
        .priority(42)
```

```

        .roleArn(iamRoleArn)
        .clientRequestToken(uuid)
        .confirmationRequired(false)
        .build();

// Create the job asynchronously.
return getAsyncClient().createJob(jobRequest)
    .thenApply(CreateJobResponse::jobId);
})
.handle((jobId, ex) -> {
    if (ex != null) {
        Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
        if (cause instanceof S3ControlException) {
            throw new CompletionException(cause);
        } else {
            throw new RuntimeException(cause);
        }
    }
    return jobId;
});
}

/**
 * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3
bucket.
 *
 * @param bucketName the name of the Amazon S3 bucket where the object is
stored
 * @param key the key (file name) of the object in the Amazon S3 bucket
 * @return the ETag of the object
 */
public String getETag(String bucketName, String key) {
    S3Client s3Client = S3Client.builder()
        .region(Region.US_EAST_1)
        .build();

    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);

```

```
        return headObjectResponse.eTag();
    }

    /**
     * Asynchronously adds tags to a job in the system.
     *
     * @param jobId      the ID of the job to add tags to
     * @param accountId the account ID associated with the job
     * @return a CompletableFuture that completes when the tagging operation is
     finished
     */
    public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
        S3Tag departmentTag = S3Tag.builder()
            .key("department")
            .value("Marketing")
            .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest =
PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " +
ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    // Setup the S3 bucket required for this scenario.
    /**
     * Creates an Amazon S3 bucket with the specified name.
```

```
*
* @param bucketName the name of the S3 bucket to create
* @throws S3Exception if there is an error creating the bucket
*/
public void createBucket(String bucketName) {
    try {
        S3Client s3Client = S3Client.builder()
            .region(Region.US_EAST_1)
            .build();

        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Uploads a file to an Amazon S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param fileName the name of the file to be uploaded
 * @throws RuntimeException if an error occurs during the file upload
 */
public void populateBucket(String bucketName, String fileName) {
    // Define the path to the directory.
    Path filePath = Paths.get("src/main/resources/batch/",
fileName).toAbsolutePath();
```

```
PutObjectRequest putOb = PutObjectRequest.builder()
    .bucket(bucketName)
    .key(fileName)
    .build();

CompletableFuture<PutObjectResponse> future =
getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
future.whenComplete((result, ex) -> {
    if (ex != null) {
        System.err.println("Error uploading file: " + ex.getMessage());
    } else {
        System.out.println("Successfully placed " + fileName + " into
bucket " + bucketName);
    }
}).join();
}

// Update the bucketName in CSV.
public void updateCSV(String newValue) {
    Path csvFilePath = Paths.get("src/main/resources/batch/job-
manifest.csv").toAbsolutePath();
    try {
        // Read all lines from the CSV file.
        List<String> lines = Files.readAllLines(csvFilePath);

        // Update the first value in each line.
        List<String> updatedLines = lines.stream()
            .map(line -> {
                String[] parts = line.split(",");
                parts[0] = newValue;
                return String.join(",", parts);
            })
            .collect(Collectors.toList());

        // Write the updated lines back to the CSV file
        Files.write(csvFilePath, updatedLines);
        System.out.println("CSV file updated successfully.");
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
```

```

    * Deletes an object from an Amazon S3 bucket asynchronously.
    *
    * @param bucketName The name of the S3 bucket where the object is stored.
    * @param objectName The name of the object to be deleted.
    * @return A {@link CompletableFuture} that completes when the object has
    been deleted,
    *         or throws a {@link RuntimeException} if an error occurs during the
    deletion.
    */
    public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
    objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
            .key(objectName)
            .build());

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(Delete.builder()
                .objects(toDelete).build())
            .build();

        return getS3AsyncClient().deleteObjects(dor)
            .thenAccept(result -> {
                System.out.println("The object was deleted!");
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
            });
    }

    /**
    * Deletes a folder and all its contents asynchronously from an Amazon S3
    bucket.
    *
    * @param bucketName the name of the S3 bucket containing the folder to be
    deleted
    * @return a {@link CompletableFuture} that completes when the folder and its
    contents have been deleted
    * @throws RuntimeException if any error occurs during the deletion process
    */
    public void deleteBucketFolderAsync(String bucketName) {
        String folderName = "reports/";
    }

```

```
ListObjectsV2Request request = ListObjectsV2Request.builder()
    .bucket(bucketName)
    .prefix(folderName)
    .build();

CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
listObjectsFuture.thenCompose(response -> {
    List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
    .map(obj -> {
        DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(obj.key())
            .build();
        return getS3AsyncClient().deleteObject(deleteRequest)
            .thenApply(deleteResponse -> {
                System.out.println("Deleted object: " + obj.key());
                return deleteResponse;
            });
    })
    .collect(Collectors.toList());

    return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))
        .thenCompose(v -> {
            // Delete the folder.
            DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(folderName)
                .build();
            return getS3AsyncClient().deleteObject(deleteRequest)
                .thenApply(deleteResponse -> {
                    System.out.println("Deleted folder: " + folderName);
                    return deleteResponse;
                });
        });
    }).join();
}

/**
 * Deletes an Amazon S3 bucket.
```



```
*
* @param bucketName the name of the bucket to delete
* @return a {@link CompletableFuture} that completes when the bucket has
been deleted, or exceptionally if there is an error
* @throws RuntimeException if there is an error deleting the bucket
*/
public CompletableFuture<Void> deleteBucket(String bucketName) {
    S3AsyncClient s3Client = getS3AsyncClient();
    return s3Client.deleteBucket(DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build())
        .thenAccept(deleteBucketResponse -> {
            System.out.println(bucketName + " was deleted");
        })
        .exceptionally(ex -> {
            // Handle the exception or rethrow it.
            throw new RuntimeException("Failed to delete bucket: " +
bucketName, ex);
        });
}

/**
 * Uploads a set of files to an Amazon S3 bucket.
 *
 * @param bucketName the name of the S3 bucket to upload the files to
 * @param fileNames an array of file names to be uploaded
 * @param actions an instance of {@link S3BatchActions} that provides the
implementation for the necessary S3 operations
 * @throws IOException if there's an error creating the text files or
uploading the files to the S3 bucket
 */
public static void uploadFilesToBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
    actions.updateCSV(bucketName);
    createTextFiles(fileNames);
    for (String fileName : fileNames) {
        actions.populateBucket(bucketName, fileName);
    }
    System.out.println("All files are placed in the S3 bucket " +
bucketName);
}

/**
 * Deletes the specified files from the given S3 bucket.
```

```
*
* @param bucketName the name of the S3 bucket
* @param fileNames an array of file names to be deleted from the bucket
* @param actions the S3BatchActions instance to be used for the file
deletion
* @throws IOException if an I/O error occurs during the file deletion
*/
public void deleteFilesFromBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
    for (String fileName : fileNames) {
        actions.deleteBucketObjects(bucketName, fileName)
            .thenRun(() -> System.out.println("Object deletion completed"))
            .exceptionally(ex -> {
                System.err.println("Error occurred: " + ex.getMessage());
                return null;
            });
    }
    System.out.println("All files have been deleted from the bucket " +
bucketName);
}

public static void createTextFiles(String[] fileNames) {
    String currentDirectory = System.getProperty("user.dir");
    String directoryPath = currentDirectory + "\\src\\main\\resources\\
\\batch";
    Path path = Paths.get(directoryPath);

    try {
        // Create the directory if it doesn't exist.
        if (Files.notExists(path)) {
            Files.createDirectories(path);
            System.out.println("Created directory: " + path.toString());
        } else {
            System.out.println("Directory already exists: " +
path.toString());
        }

        for (String fileName : fileNames) {
            // Check if the file is a .txt file.
            if (fileName.endsWith(".txt")) {
                // Define the path for the new file.
                Path filePath = path.resolve(fileName);
                System.out.println("Attempting to create file: " +
filePath.toString());
            }
        }
    }
}
```

```
// Create and write content to the new file.
Files.write(filePath, "This is a test".getBytes());

// Verify the file was created.
if (Files.exists(filePath)) {
    System.out.println("Successfully created file: " +
filePath.toString());
} else {
    System.out.println("Failed to create file: " +
filePath.toString());
}

}

} catch (IOException e) {
    System.err.println("An error occurred: " + e.getMessage());
    e.printStackTrace();
}

}

public String getAccountId() {
    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    GetCallerIdentityResponse callerIdentityResponse =
stsClient.getCallerIdentity();
    return callerIdentityResponse.account();
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [CreateJob](#)
  - [DeleteJobTagging](#)
  - [DescribeJob](#)
  - [GetJobTagging](#)
  - [ListJobs](#)
  - [PutJobTagging](#)

- [UpdateJobPriority](#)
- [UpdateJobStatus](#)

## Python

### SDK for Python (Boto3)

#### Note

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

S3 バッチの基本シナリオについて説明します。

```
class S3BatchWrapper:
    """Wrapper class for managing S3 Batch Operations."""

    def __init__(self, s3_client: Any, s3control_client: Any, sts_client: Any) ->
None:
    """
    Initializes the S3BatchWrapper with AWS service clients.

    :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                        access to AWS S3 services.
    :param s3control_client: A Boto3 Amazon S3 Control client. This client
provides
                        low-level access to AWS S3 Control services.
    :param sts_client: A Boto3 AWS STS client. This client provides low-level
                        access to AWS STS services.
    """
    self.s3_client = s3_client
    self.s3control_client = s3control_client
    self.sts_client = sts_client
    # Get region from the client for bucket creation logic
    self.region_name = self.s3_client.meta.region_name

    def get_account_id(self) -> str:
        """
        Get AWS account ID.
```

```
Returns:
    str: AWS account ID
"""
return self.sts_client.get_caller_identity()["Account"]

def create_bucket(self, bucket_name: str) -> None:
    """
    Create an S3 bucket.

    Args:
        bucket_name (str): Name of the bucket to create

    Raises:
        ClientError: If bucket creation fails
    """
    try:
        if self.region_name and self.region_name != 'us-east-1':
            self.s3_client.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration={
                    'LocationConstraint': self.region_name
                }
            )
        else:
            self.s3_client.create_bucket(Bucket=bucket_name)
        print(f"Created bucket: {bucket_name}")
    except ClientError as e:
        print(f"Error creating bucket: {e}")
        raise

def upload_files_to_bucket(self, bucket_name: str, file_names: List[str]) ->
str:
    """
    Upload files to S3 bucket including manifest file.

    Args:
        bucket_name (str): Target bucket name
        file_names (list): List of file names to upload

    Returns:
        str: ETag of the manifest file

    Raises:
```

```

        ClientError: If file upload fails
    """
    try:
        for file_name in file_names:
            if file_name != "job-manifest.csv":
                content = f"Content for {file_name}"
                self.s3_client.put_object(
                    Bucket=bucket_name,
                    Key=file_name,
                    Body=content.encode('utf-8')
                )
                print(f"Uploaded {file_name} to {bucket_name}")

        manifest_content = ""
        for file_name in file_names:
            if file_name != "job-manifest.csv":
                manifest_content += f"{bucket_name},{file_name}\n"

        manifest_response = self.s3_client.put_object(
            Bucket=bucket_name,
            Key="job-manifest.csv",
            Body=manifest_content.encode('utf-8')
        )
        print(f"Uploaded manifest file to {bucket_name}")
        print(f"Manifest content:\n{manifest_content}")
        return manifest_response['ETag'].strip('')

    except ClientError as e:
        print(f"Error uploading files: {e}")
        raise

    def create_s3_batch_job(self, account_id: str, role_arn: str,
                           manifest_location: str,
                           report_bucket_name: str) -> str:
        """
        Create an S3 batch operation job.

        Args:
            account_id (str): AWS account ID
            role_arn (str): IAM role ARN for batch operations
            manifest_location (str): Location of the manifest file
            report_bucket_name (str): Bucket for job reports

        Returns:

```

```
str: Job ID

Raises:
    ClientError: If job creation fails
"""
try:
    bucket_name = manifest_location.split(':::')[1].split('/')[0]
    manifest_key = 'job-manifest.csv'
    manifest_obj = self.s3_client.head_object(
        Bucket=bucket_name,
        Key=manifest_key
    )
    etag = manifest_obj['ETag'].strip('""')

    response = self.s3control_client.create_job(
        AccountId=account_id,
        Operation={
            'S3PutObjectTagging': {
                'TagSet': [
                    {
                        'Key': 'BatchTag',
                        'Value': 'BatchValue'
                    },
                ],
            }
        },
        Report={
            'Bucket': report_bucket_name,
            'Format': 'Report_CSV_20180820',
            'Enabled': True,
            'Prefix': 'batch-op-reports',
            'ReportScope': 'AllTasks'
        },
        Manifest={
            'Spec': {
                'Format': 'S3BatchOperations_CSV_20180820',
                'Fields': ['Bucket', 'Key']
            },
            'Location': {
                'ObjectArn': manifest_location,
                'ETag': etag
            }
        },
        Priority=10,
```

```

        RoleArn=role_arn,
        Description='Batch job for tagging objects',
        ConfirmationRequired=True
    )
    job_id = response['JobId']
    print(f"The Job id is {job_id}")
    return job_id
except ClientError as e:
    print(f"Error creating batch job: {e}")
    if 'Message' in str(e):
        print(f"Detailed error message: {e.response['Message']}")
    raise

def check_job_failure_reasons(self, job_id: str, account_id: str) ->
List[Dict[str, Any]]:
    """
    Check for any failure reasons of a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID

    Returns:
        list: List of failure reasons

    Raises:
        ClientError: If checking job failure reasons fails
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        if 'FailureReasons' in response['Job']:
            for reason in response['Job']['FailureReasons']:
                print(f"- {reason}")
        return response['Job'].get('FailureReasons', [])
    except ClientError as e:
        print(f"Error checking job failure reasons: {e}")
        raise

def wait_for_job_ready(self, job_id: str, account_id: str, desired_status:
str = 'Ready') -> bool:
    """

```



Wait for a job to reach the desired status.

Args:

job\_id (str): ID of the batch job  
 account\_id (str): AWS account ID  
 desired\_status (str): Target status to wait for

Returns:

bool: True if desired status is reached, False otherwise

Raises:

ClientError: If checking job status fails

```

"""
print(f"Waiting for job to become {desired_status}...")
max_attempts = 60
attempt = 0
while attempt < max_attempts:
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")
        if current_status == desired_status:
            return True
        if current_status == 'Suspended':
            print("Job is in Suspended state, can proceed with
activation")
            return True
        if current_status in ['Active', 'Failed', 'Cancelled',
'Complete']:
            print(f"Job is in {current_status} state, cannot reach
{desired_status} status")
            if 'FailureReasons' in response['Job']:
                print("Failure reasons:")
                for reason in response['Job']['FailureReasons']:
                    print(f"- {reason}")
            return False

        time.sleep(20)
        attempt += 1
    except ClientError as e:
        print(f"Error checking job status: {e}")

```

```
        raise
    print(f"Timeout waiting for job to become {desired_status}")
    return False

def update_job_priority(self, job_id: str, account_id: str) -> None:
    """
    Update the priority of a batch job and start it.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended']:
            self.s3control_client.update_job_priority(
                AccountId=account_id,
                JobId=job_id,
                Priority=60
            )
            print("The job priority was updated")

        try:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,
                RequestedJobStatus='Ready'
            )
            print("Job activated successfully")
        except ClientError as activation_error:
            print(f"Note: Could not activate job automatically: {activation_error}")
            print("Job priority was updated successfully. Job may need manual activation in the console.")
        elif current_status in ['Active', 'Completing', 'Complete']:
            print(f"Job is in '{current_status}' state - priority cannot be updated")

        if current_status == 'Completing':
```

```

        print("Job is finishing up and will complete soon.")
    elif current_status == 'Complete':
        print("Job has already completed successfully.")
    else:
        print("Job is currently running.")
    else:
        print(f"Job is in '{current_status}' state - priority update not
allowed")

except ClientError as e:
    print(f"Error updating job priority: {e}")
    print("Continuing with the scenario...")
    return

def cancel_job(self, job_id: str, account_id: str) -> None:
    """
    Cancel an S3 batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended', 'Active']:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,
                RequestedJobStatus='Cancelled'
            )
            print(f"Job {job_id} was successfully canceled.")
        elif current_status in ['Completing', 'Complete']:
            print(f"Job is in '{current_status}' state - cannot be
cancelled")

            if current_status == 'Completing':
                print("Job is finishing up and will complete soon.")
            elif current_status == 'Complete':
                print("Job has already completed successfully.")

```

```

        else:
            print(f"Job is in '{current_status}' state - cancel not allowed")
    except ClientError as e:
        print(f"Error canceling job: {e}")
        raise

def describe_job_details(self, job_id: str, account_id: str) -> None:
    """
    Describe detailed information about a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        job = response['Job']
        print(f"Job ID: {job['JobId']}")
        print(f"Description: {job.get('Description', 'N/A')}")
        print(f"Status: {job['Status']}")
        print(f"Role ARN: {job['RoleArn']}")
        print(f"Priority: {job['Priority']}")
        if 'ProgressSummary' in job:
            progress = job['ProgressSummary']
            print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                  f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                  f"Failed={progress.get('NumberOfTasksFailed', 0)}")
    except ClientError as e:
        print(f"Error describing job: {e}")
        raise

def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:

```

```
        response = self.s3control_client.get_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        tags = response.get('Tags', [])
        if tags:
            print(f"Tags for job {job_id}:")
            for tag in tags:
                print(f"  {tag['Key']}: {tag['Value']}")
        else:
            print(f"No tags found for job ID: {job_id}")
    except ClientError as e:
        print(f"Error getting job tags: {e}")
        raise

def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Add tags to a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.put_job_tagging(
            AccountId=account_id,
            JobId=job_id,
            Tags=[
                {'Key': 'Environment', 'Value': 'Development'},
                {'Key': 'Team', 'Value': 'DataProcessing'}
            ]
        )
        print(f"Additional tags were added to job {job_id}")
    except ClientError as e:
        print(f"Error adding job tags: {e}")
        raise

def list_jobs(self, account_id: str) -> None:
    """
    List all batch jobs for the account.

    Args:
        account_id (str): AWS account ID
    """
```

```

    try:
        response = self.s3control_client.list_jobs(
            AccountId=account_id,
            JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
        )
        jobs = response.get('Jobs', [])
        for job in jobs:
            print(f"The job id is {job['JobId']}")
            print(f"The job priority is {job['Priority']}")
    except ClientError as e:
        print(f"Error listing jobs: {e}")
        raise

def delete_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Delete all tags from a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.delete_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        print(f"You have successfully deleted {job_id} tagging.")
    except ClientError as e:
        print(f"Error deleting job tags: {e}")
        raise

def cleanup_resources(self, bucket_name: str, file_names: List[str]) -> None:
    """
    Clean up all resources created during the scenario.

    Args:
        bucket_name (str): Name of the bucket to clean up
        file_names (list): List of files to delete

    Raises:
        ClientError: If cleanup fails
    """
    try:

```

```
for file_name in file_names:
    self.s3_client.delete_object(Bucket=bucket_name, Key=file_name)
    print(f"Deleted {file_name}")

response = self.s3_client.list_objects_v2(
    Bucket=bucket_name,
    Prefix='batch-op-reports/'
)
if 'Contents' in response:
    for obj in response['Contents']:
        self.s3_client.delete_object(
            Bucket=bucket_name,
            Key=obj['Key']
        )
        print(f"Deleted {obj['Key']}")

self.s3_client.delete_bucket(Bucket=bucket_name)
print(f"Deleted bucket {bucket_name}")
except ClientError as e:
    print(f"Error in cleanup: {e}")
    raise
```

- API の詳細については、『AWS SDK for Python (Boto3) API リファレンス』の以下のトピックを参照してください。
  - [CreateJob](#)
  - [DeleteJobTagging](#)
  - [DescribeJob](#)
  - [GetJobTagging](#)
  - [ListJobs](#)
  - [PutJobTagging](#)
  - [UpdateJobPriority](#)
  - [UpdateJobStatus](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用した Amazon S3 Control のアクション

次のコード例では、AWS SDK を使用して個々の Amazon S3 Control アクションを実行する方法を示しています。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。完全なリストについては、「[Amazon S3 API Control リファレンス](#)」を参照してください。

### 例

- [AWS SDK または CLI で CreateJob を使用する](#)
- [AWS SDK で DeleteJobTagging を使用する](#)
- [AWS SDK または CLI で DescribeJob を使用する](#)
- [AWS SDK で GetJobTagging を使用する](#)
- [AWS SDK で PutJobTagging を使用する](#)
- [AWS SDK または CLI で UpdateJobPriority を使用する](#)
- [AWS SDK または CLI で UpdateJobStatus を使用する](#)

### AWS SDK または CLI で **CreateJob** を使用する

次のサンプルコードは、CreateJob を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

### CLI

#### AWS CLI

Amazon S3 バッチオペレーションジョブを作成するには

次の create-job の例では、オブジェクトを confidential` in the bucket ``employee-records`としてタグ付けする Amazon S3 バッチオペレーションジョブを作成します。

```
aws s3control create-job \
```



```
--account-id 123456789012 \
--operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"confidential",
"Value":"true"}] }}' \
--report '{"Bucket":"arn:aws:s3:::employee-records-logs","Prefix":"batch-op-
create-job",
"Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
--manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::employee-records-logs/
inv-report/7a6a9be4-072c-407e-85a2-
ec3e982f773e.csv","ETag":"69f52a4e9f797e987155d9c8f5880897"}}' \
--priority 42 \
--role-arn arn:aws:iam::123456789012:role/S3BatchJobRole
```

出力:

```
{
  "JobId": "93735294-df46-44d5-8638-6356f335324e"
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[CreateJob](#)」を参照してください。

## Java

### SDK for Java 2.x

#### Note

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

非同期 S3 ジョブを作成します。

```
/**
 * Creates an asynchronous S3 job using the AWS Java SDK.
 *
 * @param accountId      the AWS account ID associated with the job
 * @param iamRoleArn     the ARN of the IAM role to be used for the job
 * @param manifestLocation the location of the job manifest file in S3
 * @param reportBucketName the name of the S3 bucket to store the job report
```

```

    * @param uuid          a unique identifier for the job
    * @return a CompletableFuture that represents the asynchronous creation of
the S3 job.
    *      The CompletableFuture will return the job ID if the job is created
successfully,
    *      or throw an exception if there is an error.
    */
    public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
                                                    String manifestLocation,
String reportBucketName, String uuid) {

        String[] bucketName = new String[]{" "};
        String[] parts = reportBucketName.split("::");
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();
            })
    }

```

```
        JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
    .objectArn(manifestLocation)
    .eTag(eTag)
    .build();

    JobManifestSpec manifestSpec = JobManifestSpec.builder()
    .fieldsWithStrings("Bucket", "Key")
    .format("S3BatchOperations_CSV_20180820")
    .build();

    JobManifest jobManifest = JobManifest.builder()
    .spec(manifestSpec)
    .location(jobManifestLocation)
    .build();

    JobReport jobReport = JobReport.builder()
    .bucket(reportBucketName)
    .prefix("reports")
    .format("Report_CSV_20180820")
    .enabled(true)
    .reportScope("AllTasks")
    .build();

    CreateJobRequest jobRequest = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Job created using the AWS Java SDK")
    .manifest(jobManifest)
    .operation(jobOperation)
    .report(jobReport)
    .priority(42)
    .roleArn(iamRoleArn)
    .clientRequestToken(uuid)
    .confirmationRequired(false)
    .build();

    // Create the job asynchronously.
    return getAsyncClient().createJob(jobRequest)
        .thenApply(CreateJobResponse::jobId);
    })
    .handle((jobId, ex) -> {
        if (ex != null) {
```

```

        Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
        if (cause instanceof S3ControlException) {
            throw new CompletionException(cause);
        } else {
            throw new RuntimeException(cause);
        }
    }
    return jobId;
});
}

```

コンプライアンス保持ジョブを作成します。

```

/**
 * Creates a compliance retention job in Amazon S3 Control.
 * <p>
 * A compliance retention job in Amazon S3 Control is a feature that allows
you to
 * set a retention period for objects stored in an S3 bucket.
 * This feature is particularly useful for organizations that need to comply
with
 * regulatory requirements or internal policies that mandate the retention of
data for
 * a specific duration.
 *
 * @param s3ControlClient The S3ControlClient instance to use for the API
call.
 * @return The job ID of the created compliance retention job.
 */
public static String createComplianceRetentionJob(final S3ControlClient
s3ControlClient, String roleArn, String bucketName, String accountId) {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
    JobOperation jobOperation = JobOperation.builder()
        .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.COMPLIANCE)

```

```
        .retainUntilDate(jan2025)
        .build())
    .build())
    .build();

JobManifestLocation manifestLocation = JobManifestLocation.builder()
    .objectArn(manifestObjectArn)
    .eTag(manifestObjectVersionId)
    .build();

JobManifestSpec manifestSpec = JobManifestSpec.builder()
    .fieldsWithStrings("Bucket", "Key")
    .format("S3BatchOperations_CSV_20180820")
    .build();

JobManifest manifestToPublicApi = JobManifest.builder()
    .location(manifestLocation)
    .spec(manifestSpec)
    .build();

// Report details.
final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
final String jobReportPrefix = "reports/compliance-objects-bops";

JobReport jobReport = JobReport.builder()
    .enabled(true)
    .reportScope(JobReportScope.ALL_TASKS)
    .bucket(jobReportBucketArn)
    .prefix(jobReportPrefix)
    .format(JobReportFormat.REPORT_CSV_20180820)
    .build();

final Boolean requiresConfirmation = true;
final int priority = 10;
CreateJobRequest request = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Set compliance retain-until to 1 Jan 2025")
    .manifest(manifestToPublicApi)
    .operation(jobOperation)
    .priority(priority)
    .roleArn(roleArn)
    .report(jobReport)
    .confirmationRequired(requiresConfirmation)
    .build();
```

```
// Create the job and get the result.  
CreateJobResponse result = s3ControlClient.createJob(request);  
return result.jobId();  
}
```

リーガルホールド解除ジョブを作成します。

```
/**  
 * Creates a compliance retention job in Amazon S3 Control.  
 * <p>  
 * A compliance retention job in Amazon S3 Control is a feature that allows  
you to  
 * set a retention period for objects stored in an S3 bucket.  
 * This feature is particularly useful for organizations that need to comply  
with  
 * regulatory requirements or internal policies that mandate the retention of  
data for  
 * a specific duration.  
 *  
 * @param s3ControlClient The S3ControlClient instance to use for the API  
call.  
 * @return The job ID of the created compliance retention job.  
 */  
public static String createComplianceRetentionJob(final S3ControlClient  
s3ControlClient, String roleArn, String bucketName, String accountId) {  
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-  
bucket/compliance-objects-manifest.csv";  
    final String manifestObjectVersionId = "your-object-version-Id";  
  
    Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");  
    JobOperation jobOperation = JobOperation.builder()  
        .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()  
            .retention(S3Retention.builder()  
                .mode(S3ObjectLockRetentionMode.COMPLIANCE)  
                .retainUntilDate(jan2025)  
                .build())  
            .build())  
        .build();  
  
    JobManifestLocation manifestLocation = JobManifestLocation.builder()
```

```
        .objectArn(manifestObjectArn)
        .eTag(manifestObjectVersionId)
        .build();

    JobManifestSpec manifestSpec = JobManifestSpec.builder()
        .fieldsWithStrings("Bucket", "Key")
        .format("S3BatchOperations_CSV_20180820")
        .build();

    JobManifest manifestToPublicApi = JobManifest.builder()
        .location(manifestLocation)
        .spec(manifestSpec)
        .build();

    // Report details.
    final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
    final String jobReportPrefix = "reports/compliance-objects-bops";

    JobReport jobReport = JobReport.builder()
        .enabled(true)
        .reportScope(JobReportScope.ALL_TASKS)
        .bucket(jobReportBucketArn)
        .prefix(jobReportPrefix)
        .format(JobReportFormat.REPORT_CSV_20180820)
        .build();

    final Boolean requiresConfirmation = true;
    final int priority = 10;
    CreateJobRequest request = CreateJobRequest.builder()
        .accountId(accountId)
        .description("Set compliance retain-until to 1 Jan 2025")
        .manifest(manifestToPublicApi)
        .operation(jobOperation)
        .priority(priority)
        .roleArn(roleArn)
        .report(jobReport)
        .confirmationRequired(requiresConfirmation)
        .build();

    // Create the job and get the result.
    CreateJobResponse result = s3ControlClient.createJob(request);
    return result.jobId();
}
```

新しいガバナンス保持ジョブを作成します。

```
/**
 * 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 CreateGovernanceRetentionJob {

    public static void main(String[]args) throws ParseException {
        final String usage = ""

        Usage:
            <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
            <manifestObjectVersionId>

        Where:
            manifestObjectArn - The Amazon Resource Name (ARN) of the S3
            object that contains the manifest file for the governance objects.\s
            bucketName - The ARN of the S3 bucket where the job report will
            be stored.
            roleArn - The ARN of the IAM role that will be used to perform
            the governance retention operation.
            accountId - Your AWS account Id.
            manifestObjectVersionId = A unique value that is used as the
            `eTag` property of the `JobManifestLocation` object.
            "";

        if (args.length != 4) {
            System.out.println(usage);
            return;
        }

        String manifestObjectArn = args[0];
        String jobReportBucketArn = args[1];
        String roleArn = args[2];
        String accountId = args[3];
```



```
String manifestObjectVersionId = args[4];

S3ControlClient s3ControlClient = S3ControlClient.create();
createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
}

public static String createGovernanceRetentionJob(final S3ControlClient
s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String
roleArn, String accountId, String manifestObjectVersionId) throws ParseException
{
    final JobManifestLocation manifestLocation =
JobManifestLocation.builder()
        .objectArn(manifestObjectArn)
        .eTag(manifestObjectVersionId)
        .build();

    final JobManifestSpec manifestSpec = JobManifestSpec.builder()
        .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
        .fields(Arrays.asList(JobManifestFieldName.BUCKET,
JobManifestFieldName.KEY))
        .build();

    final JobManifest manifestToPublicApi = JobManifest.builder()
        .location(manifestLocation)
        .spec(manifestSpec)
        .build();

    final String jobReportPrefix = "reports/governance-objects";
    final JobReport jobReport = JobReport.builder()
        .enabled(true)
        .reportScope(JobReportScope.ALL_TASKS)
        .bucket(jobReportBucketArn)
        .prefix(jobReportPrefix)
        .format(JobReportFormat.REPORT_CSV_20180820)
        .build();

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan30th = format.parse("30/01/2025");

    final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
S3SetObjectRetentionOperation.builder()
        .retention(S3Retention.builder()
            .mode(S3ObjectLockRetentionMode.GOVERNANCE)
```

```
        .retainUntilDate(jan30th.toInstant())
        .build())
    .build();

final JobOperation jobOperation = JobOperation.builder()
    .s3PutObjectRetention(s3SetObjectRetentionOperation)
    .build();

final Boolean requiresConfirmation = true;
final int priority = 10;

final CreateJobRequest request = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Put governance retention")
    .manifest(manifestToPublicApi)
    .operation(jobOperation)
    .priority(priority)
    .roleArn(roleArn)
    .report(jobReport)
    .confirmationRequired(requiresConfirmation)
    .build();

final CreateJobResponse result = s3ControlClient.createJob(request);
return result.jobId();
}
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[CreateJob](#)」を参照してください。

## Python

### SDK for Python (Boto3)

#### Note

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

```

def create_s3_batch_job(self, account_id: str, role_arn: str,
manifest_location: str,
                        report_bucket_name: str) -> str:
    """
    Create an S3 batch operation job.

    Args:
        account_id (str): AWS account ID
        role_arn (str): IAM role ARN for batch operations
        manifest_location (str): Location of the manifest file
        report_bucket_name (str): Bucket for job reports

    Returns:
        str: Job ID

    Raises:
        ClientError: If job creation fails
    """
    try:
        bucket_name = manifest_location.split('::')[1].split('/')[0]
        manifest_key = 'job-manifest.csv'
        manifest_obj = self.s3_client.head_object(
            Bucket=bucket_name,
            Key=manifest_key
        )
        etag = manifest_obj['ETag'].strip('""')

        response = self.s3control_client.create_job(
            AccountId=account_id,
            Operation={
                'S3PutObjectTagging': {
                    'TagSet': [
                        {
                            'Key': 'BatchTag',
                            'Value': 'BatchValue'
                        },
                    ],
                }
            },
            Report={
                'Bucket': report_bucket_name,
                'Format': 'Report_CSV_20180820',
                'Enabled': True,
            }
        )
    
```

```
        'Prefix': 'batch-op-reports',
        'ReportScope': 'AllTasks'
    },
    Manifest={
        'Spec': {
            'Format': 'S3BatchOperations_CSV_20180820',
            'Fields': ['Bucket', 'Key']
        },
        'Location': {
            'ObjectArn': manifest_location,
            'ETag': etag
        }
    },
    Priority=10,
    RoleArn=role_arn,
    Description='Batch job for tagging objects',
    ConfirmationRequired=True
)
job_id = response['JobId']
print(f"The Job id is {job_id}")
return job_id
except ClientError as e:
    print(f"Error creating batch job: {e}")
    if 'Message' in str(e):
        print(f"Detailed error message: {e.response['Message']}")
    raise
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```

TRY.
  " iv_manifest_arn  = 'arn:aws:s3::my-bucket/job-manifest.csv'
  " iv_manifest_etag = 'abc123def456'
  " iv_report_bucket = 'arn:aws:s3::my-report-bucket'
  DATA(lo_result) = lo_s3c->createjob(
    iv_accountid      = iv_account_id
    iv_rolearn        = iv_role_arn
    iv_confirmationrequired = abap_true
    iv_priority       = 10
    iv_description    = 'Batch job for tagging objects'
    io_operation      = NEW /aws1/cl_s3cjoboperation(
      io_s3putobjecttagging = NEW /aws1/cl_s3cs3setobjecttagop(
        it_tagset = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
          ( NEW /aws1/cl_s3cs3tag(
            iv_key   = 'BatchTag'
            iv_value = 'BatchValue' ) )
        )
      )
    )
    io_manifest      = NEW /aws1/cl_s3cjobmanifest(
      io_spec      = NEW /aws1/cl_s3cjobmanifestspec(
        iv_format = 'S3BatchOperations_CSV_20180820'
        it_fields = VALUE /aws1/
cl_s3cjobmanifestfield00=>tt_jobmanifestfieldlist(
          ( NEW /aws1/cl_s3cjobmanifestfield00( 'Bucket' ) )
          ( NEW /aws1/cl_s3cjobmanifestfield00( 'Key' ) )
        )
      )
      io_location = NEW /aws1/cl_s3cjobmanifestloc(
        iv_objectarn = iv_manifest_arn
        iv_etag      = iv_manifest_etag
      )
    )
    io_report      = NEW /aws1/cl_s3cjobreport(
      iv_bucket     = iv_report_bucket
      iv_format     = 'Report_CSV_20180820'
      iv_enabled    = abap_true
      iv_prefix     = 'batch-op-reports'
      iv_reportscope = 'AllTasks'
    )
  ).
  ov_job_id = lo_result->get_jobid( ).
  MESSAGE |S3 Batch Operations job created: { ov_job_id }| TYPE 'I'.

```

```
CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
  MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_bad.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
  MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
  MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **DeleteJobTagging** を使用する

次のサンプルコードは、DeleteJobTagging を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

### Java

#### SDK for Java 2.x

#### Note

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

```
/**
 * Asynchronously deletes the tags associated with a specific batch job.
 *
 * @param jobId      The ID of the batch job whose tags should be deleted.
 * @param accountId  The ID of the account associated with the batch job.
 * @return A CompletableFuture that completes when the job tags have been
 *         successfully deleted, or an exception is thrown if the deletion fails.
 */
public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
    DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .build();

    return asyncClient.deleteJobTagging(jobTaggingRequest)
        .thenAccept(response -> {
            System.out.println("You have successfully deleted " + jobId + "
tagging.");
        })
        .exceptionally(ex -> {
            System.err.println("Failed to delete job tags: " +
ex.getMessage());
            throw new RuntimeException(ex);
        });
}
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

```
def delete_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Delete all tags from a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.delete_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        print(f"You have successfully deleted {job_id} tagging.")
    except ClientError as e:
        print(f"Error deleting job tags: {e}")
        raise
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.
    lo_s3c->deletejobtagging(
        iv_accountid = iv_account_id
        iv_jobid      = iv_job_id
    ).
    MESSAGE |Tags deleted from job { iv_job_id }| TYPE 'I'.
CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
```



```
EXPORTING previous = lo_ex_nf.  
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).  
MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.  
RAISE EXCEPTION TYPE /aws1/cx_rt_generic  
EXPORTING previous = lo_ex_cli.  
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).  
MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.  
RAISE EXCEPTION TYPE /aws1/cx_rt_generic  
EXPORTING previous = lo_ex_srv.  
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **DescribeJob** を使用する

次のサンプルコードは、DescribeJob を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

## CLI

### AWS CLI

Amazon S3 バッチオペレーションジョブを記述するには

次の describe-job の例では、指定されたバッチオペレーションジョブの設定パラメータとステータスを示します。

```
aws s3control describe-job \  
  --account-id 123456789012 \  
  --job-id 93735294-df46-44d5-8638-6356f335324e
```

出力:

```
{
  "Job": {
    "TerminationDate": "2019-10-03T21:49:53.944Z",
    "JobId": "93735294-df46-44d5-8638-6356f335324e",
    "FailureReasons": [],
    "Manifest": {
      "Spec": {
        "Fields": [
          "Bucket",
          "Key"
        ],
        "Format": "S3BatchOperations_CSV_20180820"
      },
      "Location": {
        "ETag": "69f52a4e9f797e987155d9c8f5880897",
        "ObjectArn": "arn:aws:s3:::employee-records-logs/inv-report/7a6a9be4-072c-407e-85a2-ec3e982f773e.csv"
      }
    },
    "Operation": {
      "S3PutObjectTagging": {
        "TagSet": [
          {
            "Value": "true",
            "Key": "confidential"
          }
        ]
      }
    },
    "RoleArn": "arn:aws:iam::123456789012:role/S3BatchJobRole",
    "ProgressSummary": {
      "TotalNumberOfTasks": 8,
      "NumberOfTasksFailed": 0,
      "NumberOfTasksSucceeded": 8
    },
    "Priority": 42,
    "Report": {
      "ReportScope": "AllTasks",
      "Format": "Report_CSV_20180820",
      "Enabled": true,
      "Prefix": "batch-op-create-job",
      "Bucket": "arn:aws:s3:::employee-records-logs"
    }
  }
}
```

```
    },  
    "JobArn": "arn:aws:s3:us-west-2:123456789012:job/93735294-  
df46-44d5-8638-6356f335324e",  
    "CreationTime": "2019-10-03T21:48:48.048Z",  
    "Status": "Complete"  
  }  
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
/**  
 * Asynchronously describes the specified job.  
 *  
 * @param jobId      the ID of the job to describe  
 * @param accountId the ID of the AWS account associated with the job  
 * @return a {@link CompletableFuture} that completes when the job  
description is available  
 * @throws RuntimeException if an error occurs while describing the job  
 */  
public CompletableFuture<Void> describeJobAsync(String jobId, String  
accountId) {  
    DescribeJobRequest jobRequest = DescribeJobRequest.builder()  
        .jobId(jobId)  
        .accountId(accountId)  
        .build();  
  
    return getAsyncClient().describeJob(jobRequest)  
        .thenAccept(response -> {  
            System.out.println("Job ID: " + response.job().jobId());  
        });  
}
```

```
        System.out.println("Description: " +
response.job().description());
        System.out.println("Status: " + response.job().statusAsString());
        System.out.println("Role ARN: " + response.job().roleArn());
        System.out.println("Priority: " + response.job().priority());
        System.out.println("Progress Summary: " +
response.job().progressSummary());

        // Print out details about the job manifest.
        JobManifest manifest = response.job().manifest();
        System.out.println("Manifest Location: " +
manifest.location().objectArn());
        System.out.println("Manifest ETag: " +
manifest.location().eTag());

        // Print out details about the job operation.
        JobOperation operation = response.job().operation();
        if (operation.s3PutObjectTagging() != null) {
            System.out.println("Operation: S3 Put Object Tagging");
            System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
        }

        // Print out details about the job report.
        JobReport report = response.job().report();
        System.out.println("Report Bucket: " + report.bucket());
        System.out.println("Report Prefix: " + report.prefix());
        System.out.println("Report Format: " + report.format());
        System.out.println("Report Enabled: " + report.enabled());
        System.out.println("Report Scope: " +
report.reportScopeAsString());
    })
    .exceptionally(ex -> {
        System.err.println("Failed to describe job: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
}
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

```
def describe_job_details(self, job_id: str, account_id: str) -> None:
    """
    Describe detailed information about a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        job = response['Job']
        print(f"Job ID: {job['JobId']}")
        print(f"Description: {job.get('Description', 'N/A')}")
        print(f"Status: {job['Status']}")
        print(f"Role ARN: {job['RoleArn']}")
        print(f"Priority: {job['Priority']}")
        if 'ProgressSummary' in job:
            progress = job['ProgressSummary']
            print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                  f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                  f"Failed={progress.get('NumberOfTasksFailed', 0)}")
    except ClientError as e:
        print(f"Error describing job: {e}")
        raise
```

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

## SAP ABAP

## SDK for SAP ABAP

 Note

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

```

TRY.
    oo_result = lo_s3c->describejob(           " oo_result is returned for
testing purposes.
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    ).
    DATA(lo_job) = oo_result->get_job( ).
    DATA(lv_status) = lo_job->get_status( ).
    DATA(lv_priority) = lo_job->get_priority( ).
    DATA(lo_progress) = lo_job->get_progressummary( ).
    IF lo_progress IS NOT INITIAL.
        MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }, | &&
                |total={ lo_progress->get_totalnumberoftasks( ) }, | &&
                |succeeded={ lo_progress->get_numberoftaskssucceeded( ) }, | &&
                |failed={ lo_progress->get_numberoftasksfailed( ) }| TYPE 'I'.
    ELSE.
        MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }| TYPE 'I'.
    ENDIF.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_srv.

```

ENDTRY.

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **GetJobTagging** を使用する

次のサンプルコードは、GetJobTagging を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

## Java

### SDK for Java 2.x

#### Note

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

```
/**
 * Asynchronously retrieves the tags associated with a specific job in an AWS
 * account.
 *
 * @param jobId      the ID of the job for which to retrieve the tags
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job tags have
 *         been retrieved, or with an exception if the operation fails
 * @throws RuntimeException if an error occurs while retrieving the job tags
 */
```

```
public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
    GetJobTaggingRequest request = GetJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return asyncClient.getJobTagging(request)
        .thenAccept(response -> {
            List<S3Tag> tags = response.tags();
            if (tags.isEmpty()) {
                System.out.println("No tags found for job ID: " + jobId);
            } else {
                for (S3Tag tag : tags) {
                    System.out.println("Tag key is: " + tag.key());
                    System.out.println("Tag value is: " + tag.value());
                }
            }
        })
        .exceptionally(ex -> {
            System.err.println("Failed to get job tags: " + ex.getMessage());
            throw new RuntimeException(ex); // Propagate the exception
        });
}
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

```
def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
```



Get tags associated with a batch job.

Args:

job\_id (str): ID of the batch job

account\_id (str): AWS account ID

"""

try:

```
response = self.s3control_client.get_job_tagging(
    AccountId=account_id,
    JobId=job_id
)
```

```
tags = response.get('Tags', [])
```

```
if tags:
```

```
    print(f"Tags for job {job_id}:")
```

```
    for tag in tags:
```

```
        print(f"    {tag['Key']}: {tag['Value']}")
```

```
else:
```

```
    print(f"No tags found for job ID: {job_id}")
```

```
except ClientError as e:
```

```
    print(f"Error getting job tags: {e}")
```

```
raise
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

TRY.

```
oo_result = lo_s3c->getjobtagging(          " oo_result is returned for
testing purposes.
```

```
iv_accountid = iv_account_id
```

```
iv_jobid      = iv_job_id
```

```
    ).  
    DATA(lt_tags) = oo_result->get_tags( ).  
    MESSAGE |Retrieved { lines( lt_tags ) } tag(s) for job { iv_job_id }|  
    TYPE 'I'.  
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).  
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.  
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic  
        EXPORTING previous = lo_ex_nf.  
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).  
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.  
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic  
        EXPORTING previous = lo_ex_cli.  
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).  
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.  
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic  
        EXPORTING previous = lo_ex_srv.  
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **PutJobTagging** を使用する

次のサンプルコードは、PutJobTagging を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

## Java

### SDK for Java 2.x

#### Note

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

```
/**
 * Asynchronously adds tags to a job in the system.
 *
 * @param jobId      the ID of the job to add tags to
 * @param accountId the account ID associated with the job
 * @return a CompletableFuture that completes when the tagging operation is
         finished
 */
public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
    S3Tag departmentTag = S3Tag.builder()
        .key("department")
        .value("Marketing")
        .build();

    S3Tag fiscalYearTag = S3Tag.builder()
        .key("FiscalYear")
        .value("2020")
        .build();

    PutJobTaggingRequest putJobTaggingRequest =
PutJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .tags(departmentTag, fiscalYearTag)
        .build();

    return asyncClient.putJobTagging(putJobTaggingRequest)
        .thenRun(() -> {
            System.out.println("Additional Tags were added to job " + jobId);
        })
        .exceptionally(ex -> {
```

```
        System.err.println("Failed to add tags to job: " +
ex.getMessage());
        throw new RuntimeException(ex); // Propagate the exception
    });
}
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

```
def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Add tags to a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.put_job_tagging(
            AccountId=account_id,
            JobId=job_id,
            Tags=[
                {'Key': 'Environment', 'Value': 'Development'},
                {'Key': 'Team', 'Value': 'DataProcessing'}
            ]
        )
        print(f"Additional tags were added to job {job_id}")
    except ClientError as e:
        print(f"Error adding job tags: {e}")
        raise
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.
  lo_s3c->putjobtagging(
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    it_tags       = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
      ( NEW /aws1/cl_s3cs3tag(
        iv_key   = 'Environment'
        iv_value = 'Development' ) )
      ( NEW /aws1/cl_s3cs3tag(
        iv_key   = 'Team'
        iv_value = 'DataProcessing' ) )
    )
  ).
  MESSAGE |Tags added to job { iv_job_id }| TYPE 'I'.
CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
  MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_nf.
CATCH /aws1/cx_s3ctoомanytagsex INTO DATA(lo_ex_tags).
  MESSAGE lo_ex_tags->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_tags.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
  MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
```

```
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).  
  MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.  
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic  
    EXPORTING previous = lo_ex_srv.  
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **UpdateJobPriority** を使用する

次のサンプルコードは、UpdateJobPriority を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

## CLI

### AWS CLI

Amazon S3 バッチオペレーションジョブのジョブ優先度を更新するには

次の update-job-priority の例では、指定されたジョブを新しい優先度に更新します。

```
aws s3control update-job-priority \  
  --account-id 123456789012 \  
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \  
  --priority 52
```

出力:

```
{  
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",  
  "Priority": 52
```

```
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
/**
 * Updates the priority of a job asynchronously.
 *
 * @param jobId      the ID of the job to update
 * @param accountId  the ID of the account associated with the job
 * @return a {@link CompletableFuture} that represents the asynchronous
 * operation, which completes when the job priority has been updated or an error
 * has occurred
 */
public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
    UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .priority(60)
        .build();

    CompletableFuture<Void> future = new CompletableFuture<>();
    getAsyncClient().updateJobPriority(priorityRequest)
        .thenAccept(response -> {
            System.out.println("The job priority was updated");
            future.complete(null); // Complete the CompletableFuture on
successful execution
        })
        .exceptionally(ex -> {
```

```
        System.err.println("Failed to update job priority: " +
ex.getMessage());
        future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
        return null; // Return null to handle the exception
    });

    return future;
}
```

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

## Python

### SDK for Python (Boto3)

#### Note

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

```
def update_job_priority(self, job_id: str, account_id: str) -> None:
    """
    Update the priority of a batch job and start it.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended']:
```



```
self.s3control_client.update_job_priority(
    AccountId=account_id,
    JobId=job_id,
    Priority=60
)
print("The job priority was updated")

try:
    self.s3control_client.update_job_status(
        AccountId=account_id,
        JobId=job_id,
        RequestedJobStatus='Ready'
    )
    print("Job activated successfully")
except ClientError as activation_error:
    print(f>Note: Could not activate job automatically:
{activation_error}")
    print("Job priority was updated successfully. Job may need
manual activation in the console.")
    elif current_status in ['Active', 'Completing', 'Complete']:
        print(f"Job is in '{current_status}' state - priority cannot be
updated")

        if current_status == 'Completing':
            print("Job is finishing up and will complete soon.")
        elif current_status == 'Complete':
            print("Job has already completed successfully.")
        else:
            print("Job is currently running.")
    else:
        print(f"Job is in '{current_status}' state - priority update not
allowed")

except ClientError as e:
    print(f"Error updating job priority: {e}")
    print("Continuing with the scenario...")
    return
```

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

## SAP ABAP

### SDK for SAP ABAP

#### Note

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

```
TRY.
    oo_result = lo_s3c->updatejobpriority(    " oo_result is returned for
testing purposes.
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    iv_priority   = 60
    ).
    MESSAGE |Job { oo_result->get_jobid( ) } priority updated to { oo_result-
>get_priority( ) }| TYPE 'I'.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
    MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_bad.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI で **UpdateJobStatus** を使用する

次のサンプルコードは、UpdateJobStatus を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

## CLI

### AWS CLI

Amazon S3 バッチオペレーションジョブのステータスを更新するには

次の update-job-status の例では、承認待ちの指定されたジョブをキャンセルします。

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
  --requested-job-status Cancelled
```

出力:

```
{
  "Status": "Cancelled",
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"
}
```

次の update-job-status の例では、承認待ちの指定されたジョブを確認して実行します。

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
  --requested-job-status Ready
```

**Output::**

```
{
  "Status": "Ready",
  "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"
}
```

次の update-job-status の例では、実行中の指定されたジョブをキャンセルします。

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
  --requested-job-status Cancelled

Output::
{
  "Status": "Cancelling",
  "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"
}
```

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

## Java

### SDK for Java 2.x

#### Note

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

```
/**
 * Cancels a job asynchronously.
 *
 * @param jobId The ID of the job to be canceled.
 * @param accountId The ID of the account associated with the job.
 * @return A {@link CompletableFuture} that completes when the job status has
 *         been updated to "CANCELLED".
 *
 *         If an error occurs during the update, the returned future will
 *         complete exceptionally.
 */
```

```

    */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
    {
        UpdateJobStatusRequest updateJobStatusRequest =
        UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
                updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
}

```

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

## Python

### SDK for Python (Boto3)

#### Note

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

```

def cancel_job(self, job_id: str, account_id: str) -> None:
    """
    Cancel an S3 batch job.

    Args:
        job_id (str): ID of the batch job
    """

```

```
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended', 'Active']:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,
                RequestedJobStatus='Cancelled'
            )
            print(f"Job {job_id} was successfully canceled.")
        elif current_status in ['Completing', 'Complete']:
            print(f"Job is in '{current_status}' state - cannot be
cancelled")
            if current_status == 'Completing':
                print("Job is finishing up and will complete soon.")
            elif current_status == 'Complete':
                print("Job has already completed successfully.")
        else:
            print(f"Job is in '{current_status}' state - cancel not allowed")
    except ClientError as e:
        print(f"Error canceling job: {e}")
        raise
```

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

## SAP ABAP

## SDK for SAP ABAP

 Note

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

```
TRY.
    " iv_requested_status = 'Cancelled'
    oo_result = lo_s3c->updatejobstatus(      " oo_result is returned for
testing purposes.
    iv_accountid      = iv_account_id
    iv_jobid          = iv_job_id
    iv_requestedjobstatus = iv_requested_status
).
MESSAGE |Job { oo_result->get_jobid( ) } status updated to { oo_result-
>get_status( ) }| TYPE 'I'.
CATCH /aws1/cx_s3cjobstatusexception INTO DATA(lo_ex_js).
MESSAGE lo_ex_js->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
EXPORTING previous = lo_ex_js.
CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
EXPORTING previous = lo_ex_nf.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
EXPORTING previous = lo_ex_cli.
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
EXPORTING previous = lo_ex_srv.
ENDTRY.
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用した S3 ディレクトリバケットのコード例

次のコード例は、AWS Software Development Kit (SDK) で S3 ディレクトリバケットを使用する方法を示しています。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

### S3 ディレクトリバケット

- [AWS SDK を使用した S3 ディレクトリバケットの基本的な例](#)
  - [Hello Amazon S3 ディレクトリバケット](#)
  - [AWS SDK を使用して S3 ディレクトリバケットの基本を学ぶ](#)
  - [AWS SDK を使用した S3 ディレクトリバケットのアクション](#)
    - [AWS SDK で AbortMultipartUpload を使用する](#)
    - [AWS SDK で CompleteMultipartUpload を使用する](#)
    - [AWS SDK で CopyObject を使用する](#)
    - [AWS SDK で CreateBucket を使用する](#)
    - [AWS SDK で CreateMultipartUpload を使用する](#)
    - [AWS SDK で CreateSession を使用する](#)
    - [AWS SDK で DeleteBucket を使用する](#)
    - [AWS SDK で DeleteBucketEncryption を使用する](#)
    - [AWS SDK で DeleteBucketPolicy を使用する](#)



- [AWS SDK で DeleteObject を使用する](#)
- [AWS SDK で DeleteObjects を使用する](#)
- [AWS SDK で GetBucketEncryption を使用する](#)
- [AWS SDK で GetBucketPolicy を使用する](#)
- [AWS SDK で GetObject を使用する](#)
- [AWS SDK で GetObjectAttributes を使用する](#)
- [AWS SDK で HeadBucket を使用する](#)
- [AWS SDK で HeadObject を使用する](#)
- [AWS SDK で ListDirectoryBuckets を使用する](#)
- [AWS SDK で ListMultipartUploads を使用する](#)
- [AWS SDK で ListObjectsV2 を使用する](#)
- [AWS SDK で ListParts を使用する](#)
- [AWS SDK で PutBucketEncryption を使用する](#)
- [AWS SDK で PutBucketPolicy を使用する](#)
- [AWS SDK で PutObject を使用する](#)
- [AWS SDK で UploadPart を使用する](#)
- [AWS SDK で UploadPartCopy を使用する](#)
- [AWS SDK を使用する S3 ディレクトリバケットのシナリオ](#)
  - [AWS SDK を使用して、オブジェクトを取得するための Amazon S3 ディレクトリバケットの署名付き URL を作成する](#)

## AWS SDK を使用した S3 ディレクトリバケットの基本的な例

次のコード例は、AWS SDK で Amazon S3 ディレクトリバケットを使用する基本的な方法を説明しています。

### 例

- [Hello Amazon S3 ディレクトリバケット](#)
- [AWS SDK を使用して S3 ディレクトリバケットの基本を学ぶ](#)
- [AWS SDK を使用した S3 ディレクトリバケットのアクション](#)
  - [AWS SDK で AbortMultipartUpload を使用する](#)
  - [AWS SDK で CompleteMultipartUpload を使用する](#)

- [AWS SDK で CopyObject を使用する](#)
- [AWS SDK で CreateBucket を使用する](#)
- [AWS SDK で CreateMultipartUpload を使用する](#)
- [AWS SDK で CreateSession を使用する](#)
- [AWS SDK で DeleteBucket を使用する](#)
- [AWS SDK で DeleteBucketEncryption を使用する](#)
- [AWS SDK で DeleteBucketPolicy を使用する](#)
- [AWS SDK で DeleteObject を使用する](#)
- [AWS SDK で DeleteObjects を使用する](#)
- [AWS SDK で GetBucketEncryption を使用する](#)
- [AWS SDK で GetBucketPolicy を使用する](#)
- [AWS SDK で GetObject を使用する](#)
- [AWS SDK で GetObjectAttributes を使用する](#)
- [AWS SDK で HeadBucket を使用する](#)
- [AWS SDK で HeadObject を使用する](#)
- [AWS SDK で ListDirectoryBuckets を使用する](#)
- [AWS SDK で ListMultipartUploads を使用する](#)
- [AWS SDK で ListObjectsV2 を使用する](#)
- [AWS SDK で ListParts を使用する](#)
- [AWS SDK で PutBucketEncryption を使用する](#)
- [AWS SDK で PutBucketPolicy を使用する](#)
- [AWS SDK で PutObject を使用する](#)
- [AWS SDK で UploadPart を使用する](#)
- [AWS SDK で UploadPartCopy を使用する](#)

## Hello Amazon S3 ディレクトリバケット

次のコード例は、Amazon S3 ディレクトリバケットの使用を開始する方法を示しています。

## Java

## SDK for Java 2.x

 Note

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

```
package com.example.s3.directorybucket;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
```

```

* authentication for SDKs and tools, see
* https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
* and Tools Reference Guide.
* <p>
* You must have a runtime environment configured with the Java SDK.
* See
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
* the Developer Guide if this is not set up.
* <p>
* To use S3 directory buckets, configure a gateway VPC endpoint. This is the
* recommended method to enable directory bucket traffic without
* requiring an internet gateway or NAT device. For more information on
* configuring VPC gateway endpoints, visit
* https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-networking.html#s3-express-networking-vpc-gateway.
* <p>
* Directory buckets are available in specific AWS Regions and Zones. For
* details on Regions and Zones supporting directory buckets, see
* https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-networking.html#s3-express-endpoints.
*/

```

```

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
        LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.
        Region region = Region.US_WEST_2;
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
            // Create the directory bucket
            createDirectoryBucket(s3Client, bucketName, zone);
            logger.info("Created bucket: {}", bucketName);

            // List all directory buckets
            List<String> bucketNames = listDirectoryBuckets(s3Client);
            bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
        } catch (S3Exception e) {
            logger.error("An error occurred during S3 operations: {} - Error
code: {}",

```

```

        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
    } finally {
        try {
            // Delete the created bucket
            deleteDirectoryBucket(s3Client, bucketName);
            logger.info("Deleted bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to delete the bucket due to S3 error: {} -
Error code: {}",
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        } catch (RuntimeException e) {
            logger.error("Failed to delete the bucket due to unexpected
error: {}", e.getMessage(), e);
        } finally {
            s3Client.close();
        }
    }
}

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client    The S3 client used to create the bucket
 * @param bucketName The name of the bucket to be created
 * @param zone        The region where the bucket will be created
 * @throws S3Exception if there's an error creating the bucket
 */
public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
    logger.info("Creating bucket: {}", bucketName);

    CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
        .location(LocationInfo.builder()
            .type(LocationType.AVAILABILITY_ZONE)
            .name(zone).build())
        .bucket(BucketInfo.builder()
            .type(BucketType.DIRECTORY)
            .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
            .build())
        .build();

```

```
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Lists all S3 directory buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
            ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                .map(Bucket::name)
                .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
}
```

```
    }  
}  
  
/**  
 * Deletes the specified S3 directory bucket.  
 *  
 * @param s3Client    The S3 client used to interact with S3  
 * @param bucketName The name of the bucket to delete  
 */  
public static void deleteDirectoryBucket(S3Client s3Client, String  
bucketName) {  
    try {  
        DeleteBucketRequest deleteBucketRequest =  
DeleteBucketRequest.builder()  
            .bucket(bucketName)  
            .build();  
        s3Client.deleteBucket(deleteBucketRequest);  
    } catch (S3Exception e) {  
        logger.error("Failed to delete bucket: " + bucketName + " - Error  
code: " + e.awsErrorDetails().errorCode(),  
            e);  
        throw e;  
    }  
}  
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [CreateBucket](#)
  - [ListDirectoryBuckets](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#)を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用して S3 ディレクトリバケットの基本を学ぶ

次のコード例は、以下を実行する方法を示しています。

- VPC と VPC エンドポイントを設定します。
- S3 ディレクトリバケットと S3 Express One Zone ストレージクラスを操作するようにポリシー、ロール、ユーザーを設定します。
- 2 つの S3 クライアントを作成します。
- バケットを 2 つ作成します。
- オブジェクトを作成し、それをコピーします。
- パフォーマンスの違いを示します。
- 辞書順の違いを示すために、バケットにデータを追加します。
- リソースをクリーンアップするかどうかをユーザーに確認します。

## Java

### SDK for Java 2.x

#### Note

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

Amazon S3 機能を示すインタラクティブなシナリオを実行します。

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");

    private static final Logger logger =
    LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;

    private static String mdirectoryBucketName;
    private static String mregularBucketName;

    private static String stackName = "cfn-stack-s3-express-basics--" +
    UUID.randomUUID();
```



```
private static String regularUser = "";
private static String vpcId = "";
private static String expressUser = "";

private static String vpcEndpointId = "";

private static final S3DirectoriesActions s3DirectoriesActions = new
S3DirectoriesActions();

public static void main(String[] args) {
    try {
        s3ExpressScenario();
    } catch (RuntimeException e) {
        logger.info(e.getMessage());
    }
}

// Runs the scenario.
private static void s3ExpressScenario() {
    logger.info(DASHES);
    logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK
for Java V2.");
    logger.info("""
        Let's get started! First, please note that S3 Express One Zone works
best when working within the AWS infrastructure,
        specifically when working in the same Availability Zone (AZ). To see
the best results in this example and when you implement
        directory buckets into your infrastructure, it is best to put your
compute resources in the same AZ as your directory
        bucket.
        """);
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    // Create an optional VPC and create 2 IAM users.
    UserNames userNames = createVpcUsers();
    String expressUserName = userNames.getExpressUserName();
    String regularUserName = userNames.getRegularUserName();

    // Set up two S3 clients, one regular and one express,
    // and two buckets, one regular and one directory.
    setupClientsAndBuckets(expressUserName, regularUserName);
}
```

```
// Create an S3 session for the express S3 client and add objects to the
buckets.
logger.info("Now let's add some objects to our buckets and demonstrate
how to work with S3 Sessions.");
waitForInputToContinue(scanner);
String bucketObject = createSessionAdd0bjects();

// Demonstrate performance differences between regular and directory
buckets.
demonstratePerformance(bucketObject);

// Populate the buckets to show the lexicographical difference between
// regular and express buckets.
showLexicographicalDifferences(bucketObject);

logger.info(DASHES);
logger.info("That's it for our tour of the basic operations for S3
Express One Zone.");
logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
String response = scanner.next().trim().toLowerCase();
if (response.equals("y")) {
    cleanUp(stackName);
}
}

/*
Delete resources created by this scenario.
*/
public static void cleanUp(String stackName) {
    try {
        if (mdirectoryBucketName != null) {

s3DirectoriesActions.deleteBucketAnd0bjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
        }
        logger.info("Deleted directory bucket " + mdirectoryBucketName);
        mdirectoryBucketName = null;
        if (mregularBucketName != null) {

s3DirectoriesActions.deleteBucketAnd0bjectsAsync(mS3RegularClient,
mregularBucketName).join();
        }
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
    }
}
```

```

        if (cause instanceof S3Exception) {
            logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
    }

    logger.info("Deleted regular bucket " + mregularBucketName);
    mregularBucketName = null;
    CloudFormationHelper.destroyCloudFormationStack(stackName);
}

private static void showLexicographicalDifferences(String bucketObject) {
    logger.info(DASHES);
    logger.info("
7. Populate the buckets to show the lexicographical (alphabetical)
difference
store
name
    objects in a different manner to regular buckets. The key is in the
    \"Directory\". Where regular buckets store their key/value pairs in a
    flat manner, directory buckets use actual directories/folders.
    This allows for more rapid indexing, traversing, and therefore
    retrieval times!

    The more segmented your bucket is, with lots of
    directories, sub-directories, and objects, the more efficient it
    becomes.
    This structural difference also causes `ListObject` operations to
    behave
    differently, which can cause unexpected results. Let's add a few
    more
    objects in sub-directories to see how the output of
    ListObjects changes.
    \"\");

    waitForInputToContinue(scanner);

    // Populate a few more files in each bucket so that we can use
    // ListObjects and show the difference.
    String otherObject = "other/" + bucketObject;
    String altObject = "alt/" + bucketObject;

```

```
String otherAltObject = "other/alt/" + bucketObject;

try {
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchBucketException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    try {
        // List objects in both S3 buckets.
        List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
        List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient,
mregularBucketName).join();

        logger.info("Directory bucket content");
        for (String obj : dirBucketObjects) {
            logger.info(obj);
        }

        logger.info("Regular bucket content");
        for (String obj : regBucketObjects) {
            logger.info(obj);
        }
    }
}
```

```

    }
    } catch (CompletionException e) {
        logger.error("Async operation failed: {} ",
e.getCause().getMessage());
        return;
    }

    logger.info("""
        Notice how the regular bucket lists objects in lexicographical order,
        while the directory bucket does not. This is
        because the regular bucket considers the whole "key" to be the object
        identifier, while the directory bucket actually
        creates directories and uses the object "key" as a path to the
        object.
        """);
    waitForInputToContinue(scanner);
}

/**
 * Demonstrates the performance difference between downloading an object from
 * a directory bucket and a regular bucket.
 *
 * <p>This method:
 * <ul>
 *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
 *     <li>Downloads the specified object from the directory bucket and
measures the total time.</li>
 *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
 *     <li>Compares the time differences and prints the results.</li>
 * </ul>
 *
 * <p>Note: The performance difference will be more pronounced if this
example is run on an EC2 instance
 * in the same Availability Zone as the buckets.
 *
 * @param bucketObject the name of the object to download
 */
private static void demonstratePerformance(String bucketObject) {
    logger.info(DASHES);
    logger.info("6. Demonstrate the performance difference.");
    logger.info("""

```

```
Now, let's do a performance test. We'll download the same object from
each
bucket repeatedly and compare the total time needed.

Note: the performance difference will be much more pronounced if this
example is run in an EC2 instance in the same Availability Zone as
the bucket.
""");
waitForInputToContinue(scanner);

int downloads = 1000; // Default value.
logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");

// Ask if the user wants to download a different number.
logger.info("Would you like to download the file a different number of
times? (y/n): ");
String response = scanner.next().trim().toLowerCase();
if (response.equals("y")) {
    int maxDownloads = 1_000_000;

    // Ask for a valid number of downloads.
    while (true) {
        logger.info("Enter a number between 1 and " + maxDownloads + "
for the number of downloads: ");
        if (scanner.hasNextInt()) {
            downloads = scanner.nextInt();
            if (downloads >= 1 && downloads <= maxDownloads) {
                break;
            } else {
                logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
            }
        } else {
            logger.info("Invalid input. Please enter a valid integer.");
            scanner.next();
        }
    }

    logger.info("You have chosen to download {} items.", downloads);
} else {
    logger.info("No changes made. Using default downloads: {}",
downloads);
}
```

```
// Simulating the download process for the directory bucket.
logger.info("Downloading from the directory bucket.");
long directoryTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        // Get the object from the directory bucket.
        s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
}
long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

// Download from the regular bucket.
logger.info("Downloading from the regular bucket.");
long normalTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
```

```

        logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
    }
    return;
}
}

    long normalTimeDifference = System.nanoTime() - normalTimeStart;
    logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
    long difference = normalTimeDifference - directoryTimeDifference;
    logger.info("That's a difference of " + difference + " nanoseconds, or");
    logger.info(difference / 1_000_000_000.0 + " seconds.");

    if (difference < 0) {
        logger.info("The directory buckets were slower. This can happen if
you are not running on the cloud within a VPC.");
    }
    waitForInputToContinue(scanner);
}

private static String createSessionAddObjects() {
    logger.info(DASHES);
    logger.info("""
        5. Create an object and copy it.
        We'll create an object consisting of some text and upload it to the
        regular bucket.
        """);
    waitForInputToContinue(scanner);

    String bucketObject = "basic-text-object.txt";
    try {
        s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
        s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

        // Copy the object to the destination S3 bucket.
        s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof S3Exception) {

```



```

        logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
    } else {
        logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
    }
}
logger.info("""
    It worked! This is because the S3Client that performed the copy
operation
    is the expressClient using the credentials for the user with
permission to
    work with directory buckets.

    It's important to remember the user permissions when interacting
with
    directory buckets. Instead of validating permissions on every call
as
    regular buckets do, directory buckets utilize the user credentials
and session
    token to validate. This allows for much faster connection speeds on
every call.
    For single calls, this is low, but for many concurrent calls
    this adds up to a lot of time saved.
    """);
    waitForInputToContinue(scanner);
    return bucketObject;
}

/**
 * Creates VPC users for the S3 Express One Zone scenario.
 * <p>
 * This method performs the following steps:
 * <ol>
 *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
 *     <li>Creates two IAM users: one with S3 Express One Zone permissions
and one without.</li>
 * </ol>
 *
 * @return a {@link UserNames} object containing the names of the created IAM
users
 */
public static UserNames createVpcUsers() {

```

```

    /*
    Optionally create a VPC.
    Create two IAM users, one with S3 Express One Zone permissions and one
without.
    */
    logger.info(DASHES);
    logger.info("""
        1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
        directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
        """);

    logger.info("Do you want to setup a VPC Endpoint? (y/n)");
    String endpointAns = scanner.nextLine().trim();
    if (endpointAns.equalsIgnoreCase("y")) {
        logger.info("""
            Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
            """);
        try {
            s3DirectoriesActions.setupVPCAsync().join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
        waitForInputToContinue(scanner);
    } else {
        logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
    }
    logger.info(DASHES);
    logger.info("""
        2. Create a RegularUser and ExpressUser by using the AWS CDK.
One IAM User, named RegularUser, will have permissions to work only
with regular buckets and one IAM user, named ExpressUser, will have
permissions to work only with directory buckets.
        """);

```

```
        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");

        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

    /**
     * Creates users using AWS CloudFormation.
     *
     * @return a {@link Map} of String keys and String values representing the
     stack outputs,
     * which may include user-related information such as user names and IDs.
     */
    public static Map<String, String> createUsersUsingCDK(String stackName) {
        logger.info("We'll use an AWS CloudFormation template to create the IAM
        users and policies.");
        CloudFormationHelper.deployCloudFormationStack(stackName);
        return CloudFormationHelper.getStackOutputsAsync(stackName).join();
    }

    /**
     * Sets up the necessary clients and buckets for the S3 Express service.
     *
     * @param expressUserName the username for the user with S3 Express
        permissions
     * @param regularUserName the username for the user with regular S3
        permissions
     */
    public static void setupClientsAndBuckets(String expressUserName, String
        regularUserName) {
        Scanner locscanner = new Scanner(System.in);
        String accessKeyIdforRegUser;
        String secretAccessforRegUser;
        try {
            CreateAccessKeyResponse keyResponse =
            s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
            accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
        }
```

```

        secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof IamException) {
            logger.error("IamException occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    String accessKeyIdforExpressUser;
    String secretAccessforExpressUser;
    try {
        CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
        accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
        secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof IamException) {
            logger.error("IamException occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    logger.info(DASHES);
    logger.info("""
        3. Create two S3Clients; one uses the ExpressUser's credentials and
one uses the RegularUser's credentials.
        The 2 S3Clients will use different credentials.
        """);
    waitForInputToContinue(locscanner);
    try {

```

```

        mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
        mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof IllegalArgumentException) {
            logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    logger.info("""
        We can now use the ExpressUser client to make calls to S3 Express
operations.
        """);
    waitForInputToContinue(locscanner);
    logger.info(DASHES);
    logger.info("""
        4. Create two buckets.
        Now we will create a directory bucket which is the linchpin of the S3
Express One Zone service. Directory buckets
        behave differently from regular S3 buckets which we will explore
here. We'll also create a regular bucket, put
        an object into the regular bucket, and copy it to the directory
bucket.
        """);

    logger.info("""
        Now, let's choose an availability zone (AZ) for the directory
bucket.
        We'll choose one that is supported.
        """);
    String zoneId;
    String regularBucketName;
    try {
        zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();

```

```
        regularBucketName = "amzn-s3-demo-bucket"; // Replace with your
bucket name.
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof Ec2Exception) {
            logger.error("EC2Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
    logger.info("""
Now, let's create the actual directory bucket, as well as a regular
bucket."
        """);

    String directoryBucketName = "amzn-s3-demo-bucket--" + zoneId + "--x-
s3"; // Replace with your directory bucket name.
    try {
        s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
        logger.info("Created directory bucket {}", directoryBucketName);
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof BucketAlreadyExistsException) {
            logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    // Assign to the data member.
    mdirectoryBucketName = directoryBucketName;
    try {
        s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
        logger.info("Created regular bucket {} ", regularBucketName);
        mregularBucketName = regularBucketName;
    } catch (CompletionException ce) {
```

```

        Throwable cause = ce.getCause();
        if (cause instanceof BucketAlreadyExistsException) {
            logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            return;
        }
    }
    logger.info("Great! Both buckets were created.");
    waitForInputToContinue(locscanner);
}

/**
 * Creates an asynchronous S3 client with the specified access key and secret
access key.
 *
 * @param accessKeyId      the AWS access key ID
 * @param secretAccessKey the AWS secret access key
 * @return a {@link CompletableFuture} that asynchronously creates the S3
client
 * @throws IllegalArgumentException if the access key ID or secret access key
is null
 */
public static CompletableFuture<S3AsyncClient>
createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
    return CompletableFuture.supplyAsync(() -> {
        // Validate input parameters
        if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey
== null || secretAccessKey.isBlank()) {
            throw new IllegalArgumentException("Access Key ID and Secret
Access Key must not be null or empty");
        }

        AwsBasicCredentials awsCredentials =
AwsBasicCredentials.create(accessKeyId, secretAccessKey);
        return S3AsyncClient.builder()

.credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
        .region(Region.US_WEST_2)
        .build();
    });
}

```

```
private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        logger.info("");
        logger.info("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            logger.info("Continuing with the program...");
            logger.info("");
            break;
        } else {
            logger.info("Invalid input. Please try again.");
        }
    }
}
```

Amazon S3 SDK メソッドのラッパークラス。

```
public class S3DirectoriesActions {

    private static IamAsyncClient iamAsyncClient;

    private static Ec2AsyncClient ec2AsyncClient;
    private static final Logger logger =
        LoggerFactory.getLogger(S3DirectoriesActions.class);

    private static IamAsyncClient getIAMAsyncClient() {
        if (iamAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
                ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
```



```

        .build();

        iamAsyncClient = IamAsyncClient.builder()
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return iamAsyncClient;
}

private static Ec2AsyncClient getEc2AsyncClient() {
    if (ec2AsyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryStrategy(RetryMode.STANDARD)
            .build();

        ec2AsyncClient = Ec2AsyncClient.builder()
            .httpClient(httpClient)
            .region(Region.US_WEST_2)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return ec2AsyncClient;
}

/**
 * Deletes the specified S3 bucket and all the objects within it
asynchronously.
 *
 * @param s3AsyncClient the S3 asynchronous client to use for the operations
 * @param bucketName the name of the S3 bucket to be deleted
 * @return a {@link CompletableFuture} that completes with a {@link
WaiterResponse} containing the

```

```

    *      {@link HeadBucketResponse} when the bucket has been successfully
deleted
    * @throws CompletionException if there was an error deleting the bucket or
its objects
    */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
    ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3AsyncClient.listObjectsV2(listRequest)
        .thenCompose(listResponse -> {
            if (!listResponse.contents().isEmpty()) {
                List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()
                    .map(s3Object ->
ObjectIdentifier.builder().key(s3Object.key()).build())
                    .collect(Collectors.toList());

                DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
                    .bucket(bucketName)

.delete(Delete.builder().objects(objectIdentifiers).build())
                    .build();

                return s3AsyncClient.deleteObjects(deleteRequest)
                    .thenAccept(deleteResponse -> {
                        if (!deleteResponse.errors().isEmpty()) {
                            deleteResponse.errors().forEach(error ->
logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                        }
                    });
            }
            return CompletableFuture.completedFuture(null);
        })
        .thenCompose(ignored -> {
            DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
            return s3AsyncClient.deleteBucket(deleteBucketRequest);
        });
}

```

```

    })
    .thenCompose(ignored -> {
        S3AsyncWaiter waiter = s3AsyncClient.waiter();
        HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
        return waiter.waitUntilBucketNotExists(headBucketRequest);
    })
    .whenComplete((ignored, exception) -> {
        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof S3Exception) {
                throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
            }
            throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
        }
        logger.info("Bucket deleted successfully: " + bucketName);
    });
}

/**
 * Lists the objects in an S3 bucket asynchronously.
 *
 * @param s3Client the S3 async client to use for the operation
 * @param bucketName the name of the S3 bucket containing the objects to list
 * @return a {@link CompletableFuture} that contains the list of object keys
in the specified bucket
 */
public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient
s3Client, String bucketName) {
    ListObjectsV2Request request = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3Client.listObjectsV2(request)
        .thenApply(response -> response.contents().stream()
            .map(S3Object::key)
            .toList())
        .whenComplete((result, exception) -> {
            if (exception != null) {
                throw new CompletionException("Couldn't list objects in
bucket: " + bucketName, exception);
            }
        })
}

```

```

    });
}

/**
 * Retrieves an object from an Amazon S3 bucket asynchronously.
 *
 * @param s3Client    the S3 async client to use for the operation
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName     the unique identifier (key) of the object to retrieve
 * @return a {@link CompletableFuture} that, when completed, contains the
 * object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
 */
public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

    // Get the object asynchronously and transform it into a byte array
    return s3Client.getObject(objectRequest,
        AsyncResponseTransformer.toBytes())
        .exceptionally(exception -> {
            Throwable cause = exception.getCause();
            if (cause instanceof NoSuchKeyException) {
                throw new CompletionException("Failed to get the object.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
            }
            throw new CompletionException("Failed to get the object",
exception);
        });
}

/**
 * Asynchronously copies an object from one S3 bucket to another.
 *
 * @param s3Client    the S3 async client to use for the copy
operation
 * @param sourceBucket the name of the source bucket
 * @param sourceKey    the key of the object to be copied in the source
bucket
 * @param destinationBucket the name of the destination bucket
 * @param destinationKey  the key of the copied object in the destination
bucket

```

```

    * @return a {@link CompletableFuture} that completes when the copy operation
    is finished
    */
    public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey)
{
    CopyObjectRequest copyRequest = CopyObjectRequest.builder()
        .sourceBucket(sourceBucket)
        .sourceKey(sourceKey)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    return s3Client.copyObject(copyRequest)
        .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
        .whenComplete((ignored, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof S3Exception) {
                    throw new CompletionException("Couldn't
copy object '" + sourceKey + "' from bucket '" + sourceBucket + "'
to bucket '" + destinationBucket + "'. Reason: " + ((S3Exception)
cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to copy object",
exception);
            }
        });
}

/**
 * Asynchronously creates a session for the specified S3 bucket.
 *
 * @param s3Client the S3 asynchronous client to use for creating the
session
 * @param bucketName the name of the S3 bucket for which to create the
session
 * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
 */
    public CompletableFuture<CreateSessionResponse>
createSessionAsync(S3AsyncClient s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()

```

```

        .bucket(bucketName)
        .build();

    return s3Client.createSession(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof S3Exception) {
                    throw new CompletionException("Couldn't create the
session. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(),
cause);
                }
                throw new CompletionException("Unexpected error occurred
while creating session", exception);
            }
            logger.info("Created session for bucket: " + bucketName);
        });
    }

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client    The asynchronous S3 client used to create the bucket
 * @param bucketName The name of the bucket to be created
 * @param zone        The Availability Zone where the bucket will be created
 * @throws CompletionException if there's an error creating the bucket
 */
    public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String
zone) {
        logger.info("Creating bucket: " + bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                .type(LocationType.AVAILABILITY_ZONE)
                .name(zone)
                .build())
            .bucket(BucketInfo.builder()
                .type(BucketType.DIRECTORY)
                .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                .build())

```

```

        .build();

        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration)
            .build();

        return s3Client.createBucket(bucketRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The bucket already exists:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred
while creating bucket", exception);
                }
                logger.info("Bucket created successfully with location: " +
response.location());
            });
    }

    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param s3Client    the S3 async client to use for the bucket creation
     * @param bucketName  the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
     *         when the bucket is successfully created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createBucket(bucketRequest)
            .thenCompose(response -> {
                S3AsyncWaiter s3Waiter = s3Client.waiter();
                HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)

```

```

        .build();
        return s3Waiter.waitUntilBucketExists(bucketRequestWait);
    })
    .whenComplete((response, exception) -> {
        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
            } else {
                throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
            }
        }
        logger.info(bucketName + " is ready");
    });
}

/**
 * Uploads an object to an Amazon S3 bucket asynchronously.
 *
 * @param s3Client    the S3 async client to use for the upload
 * @param bucketName  the destination S3 bucket name
 * @param bucketObject the name of the object to be uploaded
 * @param text        the content to be uploaded as the object
 */
public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(bucketObject)
        .build();

    return s3Client.putObject(objectRequest,
AsyncRequestBody.fromString(text))
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchBucketException) {
                    throw new CompletionException("The S3 bucket does not
exist: " + cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);

```



```

        }
    }
    });
}

/**
 * Creates an AWS IAM access key asynchronously for the specified user name.
 *
 * @param userName the name of the IAM user for whom to create the access key
 * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
 */
public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
    CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
        .userName(userName)
        .build();

    return getIAMAsyncClient().createAccessKey(request)
        .whenComplete((response, exception) -> {
            if (response != null) {
                logger.info("Access Key Created.");
            } else {
                if (exception == null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof IamException) {
                        throw new CompletionException("IAM error while
creating access key: " + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create
access key: " + exception.getMessage(), exception);
                    }
                }
            }
        });
}

/**
 * Asynchronously selects an Availability Zone ID from the available EC2
zones.
 *
 * @return A {@link CompletableFuture} that resolves to the selected
Availability Zone ID.

```

```
    * @throws CompletionException if an error occurs during the request or
    processing.
    */
    public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
        DescribeAvailabilityZonesRequest zonesRequest =
        DescribeAvailabilityZonesRequest.builder()
            .build();

        return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
            .thenCompose(response -> {
                List<AvailabilityZone> zonesList = response.availabilityZones();
                if (zonesList.isEmpty()) {
                    logger.info("No availability zones found.");
                    return CompletableFuture.completedFuture(null); // Return
null if no zones are found
                }

                List<String> zoneIds = zonesList.stream()
                    .map(AvailabilityZone::zoneId) // Get the zoneId (e.g.,
"usw2-az1")
                    .toList();

                return CompletableFuture.supplyAsync(() ->
promptUserForZoneSelection(zonesList, zoneIds))
                    .thenApply(selectedZone -> {
                        // Return only the selected Zone ID (e.g., "usw2-az1").
                        return selectedZone.zoneId();
                    });
            })
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Selected Availability Zone ID: " + result);
                    } else {
                        logger.info("No availability zone selected.");
                    }
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof Ec2Exception) {
                        throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
                }
            });
    }
```

```

        }
    });
}

/**
 * Prompts the user to select an Availability Zone from the given list.
 *
 * @param zonesList the list of Availability Zones
 * @param zoneIds the list of zone IDs
 * @return the selected Availability Zone
 */
private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String>
zoneIds) {
    Scanner scanner = new Scanner(System.in);
    int index = -1;

    while (index < 0 || index >= zoneIds.size()) {
        logger.info("Select an availability zone:");
        IntStream.range(0, zoneIds.size()).forEach(i ->
            logger.info(i + ": " + zoneIds.get(i))
        );

        logger.info("Enter the number corresponding to your choice: ");
        if (scanner.hasNextInt()) {
            index = scanner.nextInt();
        } else {
            scanner.next();
        }
    }

    AvailabilityZone selectedZone = zonesList.get(index);
    logger.info("You selected: " + selectedZone.zoneId());
    return selectedZone;
}

/**
 * Asynchronously sets up a new VPC, including creating the VPC, finding the
 * associated route table, and
 * creating a VPC endpoint for the S3 service.
 *
 * @return a {@link CompletableFuture} that, when completed, contains a
 * AbstractMap with the
 * VPC ID and VPC endpoint ID.

```

```
    */
    public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
    setupVPCAsync() {
        String cidr = "10.0.0.0/16";
        CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
            .cidrBlock(cidr)
            .build();

        return getEc2AsyncClient().createVpc(vpcRequest)
            .thenCompose(vpcResponse -> {
                String vpcId = vpcResponse.vpc().vpcId();
                logger.info("VPC Created: {}", vpcId);

                Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
                DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .vpcIds(vpcId)
                    .build();

                return waiter.waitUntilVpcAvailable(request)
                    .thenApply(waiterResponse -> vpcId);
            })
            .thenCompose(vpcId -> {
                Filter filter = Filter.builder()
                    .name("vpc-id")
                    .values(vpcId)
                    .build();

                DescribeRouteTablesRequest describeRouteTablesRequest =
                DescribeRouteTablesRequest.builder()
                    .filters(filter)
                    .build();

                return
                getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
                    .thenApply(routeTablesResponse -> {
                        if (routeTablesResponse.routeTables().isEmpty()) {
                            throw new CompletionException("No route tables found
for VPC: " + vpcId, null);
                        }
                        String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                        logger.info("Route table found: {}", routeTableId);
                        return new AbstractMap.SimpleEntry<>(vpcId,
routeTableId);
                    });
            });
    }
```

```

        });
    })
    .thenCompose(vpcAndRouteTable -> {
        String vpcId = vpcAndRouteTable.getKey();
        String routeTableId = vpcAndRouteTable.getValue();
        Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
        String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

        CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
            .vpcId(vpcId)
            .routeTableIds(routeTableId)
            .serviceName(serviceName)
            .build();

        return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
            .thenApply(vpcEndpointResponse -> {
                String vpcEndpointId =
vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                logger.info("VPC Endpoint created: {}", vpcEndpointId);
                return new AbstractMap.SimpleEntry<>(vpcId,
vpcEndpointId);
            });
    })
    .exceptionally(exception -> {
        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
        if (cause instanceof Ec2Exception) {
            logger.error("EC2 error during VPC setup: {}",
cause.getMessage(), cause);
            throw new CompletionException("EC2 error during VPC setup: "
+ cause.getMessage(), cause);
        }

        logger.error("VPC setup failed: {}", cause.getMessage(), cause);
        throw new CompletionException("VPC setup failed: " +
cause.getMessage(), cause);
    });
}
}

```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObject](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## PHP

### SDK for PHP

#### Note

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

Amazon S3 デイレクトリバケットと S3 Express One Zone の基本を示すシナリオを実行します。

```
echo "\n";
echo "-----\n";
echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
echo "-----\n";

// Change these both of these values to use a different region/
availability zone.
$region = "us-west-2";
$az = "usw2-az1";

$this->s3Service = new S3Service(new S3Client(['region' => $region]));
$this->iamService = new IAMService(new IamClient(['region' => $region]));
```

```

    $uuid = uniqid();

    echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
in this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute
resources in the same AZ as your Directory
bucket.\n
INTRO;

    pressEnter();
    // 1. Configure a gateway VPC endpoint. This is the recommended method to
allow S3 Express One Zone traffic without
    // the need to pass through an internet gateway or NAT device.
    echo "\n";
    echo "1. First, we'll set up a new VPC and VPC Endpoint if this program
is running in an EC2 instance in the same AZ as your Directory buckets will be.
\n";

    $ec2Choice = testable_readline("Are you running this in an EC2 instance
located in the same AZ as your intended Directory buckets? Enter Y/y to setup a
VPC Endpoint, or N/n/blank to skip this section.");
    if($ec2Choice == "Y" || $ec2Choice == "y") {
        echo "Great! Let's set up a VPC, retrieve the Route Table from it,
and create a VPC Endpoint to connect the S3 Client to.\n";
        pressEnter();
        $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
$region]));
        $cidr = "10.0.0.0/16";
        $vpc = $this->ec2Service->createVpc($cidr);
        $this->resources['vpcId'] = $vpc['VpcId'];

        $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);

        $routeTable = $this->ec2Service->describeRouteTables([], [
            [
                'Name' => "vpc-id",
                'Values' => [$vpc['VpcId']],
            ],
        ]);

        $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
"s3express";

```

```

        $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
        $vpc['VpcId'], [$routeTable[0]]);
        $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
    }else{
        echo "Skipping the VPC setup. Don't forget to use this in production!
\n";
    }

    // 2. Policies, user, and roles with CDK.
    echo "\n";
    echo "2. Policies, users, and roles with CDK.\n";
    echo "Now, we'll set up some policies, roles, and a user. This user will
only have permissions to do S3 Express One Zone actions.\n";
    pressEnter();

    $this->cloudFormationClient = new CloudFormationClient([]);
    $stackName = "cfn-stack-s3-express-basics-" . uniqid();
    $file = file_get_contents(__DIR__ . "../resources/cfn/
s3_express_basics/s3_express_template.yml");
    $result = $this->cloudFormationClient->createStack([
        'StackName' => $stackName,
        'TemplateBody' => $file,
        'Capabilities' => ['CAPABILITY_IAM'],
    ]);
    $waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
    ['StackName' => $stackName]);
    try {
        $waiter->promise()->wait();
    }catch(CloudFormationException $caught){
        echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
    $this->resources['stackName'] = $stackName;
    $stackInfo = $this->cloudFormationClient->describeStacks([
        'StackName' => $result['StackId'],
    ]);

    $expressUserName = "";
    $regularUserName = "";
    foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
        if ($output['OutputKey'] == "RegularUser") {
            $regularUserName = $output['OutputValue'];
        }
    }

```



```
        if ($output['OutputKey'] == "ExpressUser") {
            $expressUserName = $output['OutputValue'];
        }
    }
    $regularKey = $this->iamService->createAccessKey($regularUserName);
    $regularCredentials = new Credentials($regularKey['AccessKeyId'],
    $regularKey['SecretAccessKey']);
    $expressKey = $this->iamService->createAccessKey($expressUserName);
    $expressCredentials = new Credentials($expressKey['AccessKeyId'],
    $expressKey['SecretAccessKey']);

    // 3. Create an additional client using the credentials with S3 Express
    permissions.
    echo "\n";
    echo "3. Create an additional client using the credentials with S3
    Express permissions.\n";
    echo "This client is created with the credentials associated with the
    user account with the S3 Express policy attached, so it can perform S3 Express
    operations.\n";
    pressEnter();
    $s3RegularClient = new S3Client([
        'Region' => $region,
        'Credentials' => $regularCredentials,
    ]);
    $s3RegularService = new S3Service($s3RegularClient);
    $s3ExpressClient = new S3Client([
        'Region' => $region,
        'Credentials' => $expressCredentials,
    ]);
    $s3ExpressService = new S3Service($s3ExpressClient);
    echo "All the roles and policies were created an attached to the user.
    Then, a new S3 Client and Service were created using that user's credentials.
    \n";

    echo "We can now use this client to make calls to S3 Express operations.
    Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
    Express.\n";
    pressEnter();

    // 4. Create two buckets.
    echo "\n";
    echo "3. Create two buckets.\n";
    echo "Now we will create a Directory bucket, which is the linchpin of the
    S3 Express One Zone service.\n";
```

```
    echo "Directory buckets behave in different ways from regular S3 buckets,
which we will explore here.\n";
    echo "We'll also create a normal bucket, put an object into the normal
bucket, and copy it over to the Directory bucket.\n";
    pressEnter();

    // Create a directory bucket. These are different from normal S3 buckets
in subtle ways.
    $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-
s3";
    echo "Now, let's create the actual Directory bucket, as well as a regular
bucket.\n";
    pressEnter();
    $s3ExpressService->createBucket($directoryBucketName, [
        'CreateBucketConfiguration' => [
            'Bucket' => [
                'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
                'DataRedundancy' => "SingleAvailabilityZone",
            ],
            'Location' => [
                'Name' => $az,
                'Type' => "AvailabilityZone",
            ],
        ],
    ]);
    $this->resources['directoryBucketName'] = $directoryBucketName;

    // Create a normal bucket.
    $normalBucketName = "normal-bucket-$uuid";
    $s3RegularService->createBucket($normalBucketName);
    $this->resources['normalBucketName'] = $normalBucketName;
    echo "Great! Both buckets were created.\n";
    pressEnter();

    // 5. Create an object and copy it over.
    echo "\n";
    echo "5. Create an object and copy it over.\n";
    echo "We'll create a basic object consisting of some text and upload it
to the normal bucket.\n";
    echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
    echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
```

```
pressEnter();

$objectKey = "basic-text-object";
$s3RegularService->putObject($normalBucketName, $objectKey, $args =
['Body' => "Look Ma, I'm a bucket!"]);
$this->resources['objectKey'] = $objectKey;

// Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
$s3ExpressService->createSession($directoryBucketName);
$s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");

echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";
echo "Instead of validating permissions on every call as normal buckets
do, Directory buckets utilize the user credentials and session token to
validate.\n";
echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot
of time saved.\n";
pressEnter();

// 6. Demonstrate performance difference.
echo "\n";
echo "6. Demonstrate performance difference.\n";
$downloads = 1000;
echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
$downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
    $downloads = $downloadChoice;
}

// Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
$directoryStartTime = hrtime(true);
for($i = 0; $i < $downloads; ++$i){
    $s3ExpressService->getObject($directoryBucketName, $objectKey);
```

```
}
$directoryEndTime = hrtime(true);
$directoryTimeDiff = $directoryEndTime - $directoryStartTime;

$normalStartTime = hrtime(true);
for($i = 0; $i < $downloads; ++$i){
    $s3RegularService->getObject($normalBucketName, $objectKey);
}
$normalEndTime = hrtime(true);
$normalTimeDiff = $normalEndTime - $normalStartTime;

echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
normal bucket took $normalTimeDiff.\n";
echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
" nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
seconds.\n";
pressEnter();

// 7. Populate the buckets to show the lexicographical difference.
echo "\n";
echo "7. Populate the buckets to show the lexicographical difference.\n";
echo "Now let's explore how Directory buckets store objects in a
different manner to regular buckets.\n";
echo "The key is in the name \"Directory!\".\n";
echo "Where regular buckets store their key/value pairs in a flat manner,
Directory buckets use actual directories/folders.\n";
echo "This allows for more rapid indexing, traversing, and therefore
retrieval times!\n";
echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
echo "This structural difference also causes ListObjects to behave
differently, which can cause unexpected results.\n";
echo "Let's add a few more objects with layered directories as see how
the output of ListObjects changes.\n";
pressEnter();

// Populate a few more files in each bucket so that we can use
ListObjects and show the difference.
$otherObject = "other/$objectKey";
$altObject = "alt/$objectKey";
$otherAltObject = "other/alt/$objectKey";
$s3ExpressService->putObject($directoryBucketName, $otherObject);
$s3RegularService->putObject($normalBucketName, $otherObject);
$this->resources['otherObject'] = $otherObject;
```

```
$s3ExpressService->putObject($directoryBucketName, $altObject);
$s3RegularService->putObject($normalBucketName, $altObject);
$this->resources['altObject'] = $altObject;
$s3ExpressService->putObject($directoryBucketName, $otherAltObject);
$s3RegularService->putObject($normalBucketName, $otherAltObject);
$this->resources['otherAltObject'] = $otherAltObject;

$listDirectoryBucket = $s3ExpressService-
>listObjects($directoryBucketName);
$listNormalBucket = $s3RegularService->listObjects($normalBucketName);

// Directory bucket content
echo "Directory bucket content\n";
foreach($listDirectoryBucket['Contents'] as $result){
    echo $result['Key'] . "\n";
}

// Normal bucket content
echo "\nNormal bucket content\n";
foreach($listNormalBucket['Contents'] as $result){
    echo $result['Key'] . "\n";
}

echo "Notice how the normal bucket lists objects in lexicographical
order, while the directory bucket does not. This is because the normal bucket
considers the whole \"key\" to be the object identifies, while the directory
bucket actually creates directories and uses the object \"key\" as a path to the
object.\n";
pressEnter();

echo "\n";
echo "That's it for our tour of the basic operations for S3 Express One
Zone.\n";
$cleanUp = testable_readline("Would you like to delete all the resources
created during this demo? Enter Y/y to delete all the resources.");
if($cleanUp){
    $this->cleanUp();
}

namespace S3;

use Aws\CommandInterface;
```

```
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }

    public function setClient(S3Client $client)
    {
        $this->client = $client;
    }
}
```

```
public function emptyAndDeleteBucket($bucketName, array $args = [])
{
    try {
        $objects = $this->listAllObjects($bucketName, $args);
        $this->deleteObjects($bucketName, $objects, $args);
        if ($this->verbose) {
            echo "Deleted all objects and folders from $bucketName.\n";
        }
        $this->deleteBucket($bucketName, $args);
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "\nPlease fix error with bucket deletion before continuing.
\n";
        }
        throw $exception;
    }
}

public function createBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->createBucket($parameters);
        if ($this->verbose) {
            echo "Created the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket creation before continuing.";
        }
        throw $exception;
    }
}
```

```
public function putObject(string $bucketName, string $key, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
    $args);
    try {
        $this->client->putObject($parameters);
        if ($this->verbose) {
            echo "Uploaded the object named: $key to the bucket named:
    $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $key in $bucketName with error:
    {$exception->getMessage()}\n";
            echo "Please fix error with object uploading before continuing.";
        }
        throw $exception;
    }
}

public function getObject(string $bucketName, string $key, array $args = []):
Result
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
    $args);
    try {
        $object = $this->client->getObject($parameters);
        if ($this->verbose) {
            echo "Downloaded the object named: $key to the bucket named:
    $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to download $key from $bucketName with error:
    {$exception->getMessage()}\n";
            echo "Please fix error with object downloading before
    continuing.";
        }
        throw $exception;
    }
    return $object;
}
```



```
public function copyObject($bucketName, $key, $copySource, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
    try {
        $this->client->copyObject($parameters);
        if ($this->verbose) {
            echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object copying before continuing.";
        }
        throw $exception;
    }
}

public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
    try {
        $objects = $this->client->listObjectsV2($parameters);
        if ($this->verbose) {
            echo "Retrieved the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with list objects before continuing.";
        }
        throw $exception;
    }
    return $objects;
}
```

```
}

public function listAllObjects($bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);

    $contents = [];
    $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

    foreach ($paginator as $result) {
        if($result['KeyCount'] == 0){
            break;
        }
        foreach ($result['Contents'] as $object) {
            $contents[] = $object;
        }
    }
    return $contents;
}

public function deleteObjects(string $bucketName, array $objects, array $args
= [])
{
    $listOfObjects = array_map(
        function ($object) {
            return ['Key' => $object];
        },
        array_column($objects, 'Key')
    );
    if(!$listOfObjects){
        return;
    }

    $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
    try {
        $this->client->deleteObjects($parameters);
        if ($this->verbose) {
            echo "Deleted the list of objects from: $bucketName.\n";
        }
    }
```

```
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function deleteObject(string $bucketName, string $fileName, array
$args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
```

```
        echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
        echo "Please fix error with object deletion before continuing.";
    }
    throw $exception;
}

}

public function listBuckets(array $args = [])
{
    try {
        $buckets = $this->client->listBuckets($args);
        if ($this->verbose) {
            echo "Retrieved all " . count($buckets) . "\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket lists before continuing.";
        }
        throw $exception;
    }
    return $buckets;
}

public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
{
    $request = $this->client->createPresignedRequest($command, $expires,
$options);
    try {
        $presignedUrl = (string)$request->getUri();
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
            echo "Please fix error with presigned urls before continuing.";
        }
        throw $exception;
    }
}
```

```
    }  
    return $presignedUrl;  
}  
  
public function createSession(string $bucketName)  
{  
    try{  
        $result = $this->client->createSession([  
            'Bucket' => $bucketName,  
        ]);  
        return $result;  
    }catch(S3Exception $caught){  
        if($caught->getAwsErrorType() == "NoSuchBucket"){  
            echo "The specified bucket does not exist.";  
        }  
        throw $caught;  
    }  
}  
}
```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の以下のトピックを参照してください。
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObject](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## Python

### SDK for Python (Boto3)

#### Note

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

Amazon S3 ディレクトリバケットと S3 Express One Zone の基本を示すシナリオを実行します。

```
class S3ExpressScenario:
    """Runs an interactive scenario that shows how to get started with S3
    Express."""

    def __init__(
        self,
        cloud_formation_resource: ServiceResource,
        ec2_client: client,
        iam_client: client,
    ):
        self.cloud_formation_resource = cloud_formation_resource
        self.ec2_client = ec2_client
        self.iam_client = iam_client
        self.region = ec2_client.meta.region_name
        self.stack = None
        self.vpc_id = None
        self.vpc_endpoint_id = None
        self.regular_bucket_name = None
        self.directory_bucket_name = None
        self.s3_express_wrapper = None
        self.s3_regular_wrapper = None

    def s3_express_scenario(self):
        """
        Runs the scenario.
        """
        print("")
        print_dashes()
```

```

        print("Welcome to the Amazon S3 Express Basics demo using Python (Boto
3)!")
        print_dashes()
        print(
            """
Let's get started! First, please note that S3 Express One Zone works best when
working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
in this example and when you implement
Directory buckets into your infrastructure, it is best to put your compute
resources in the same AZ as your Directory
bucket.
        """
        )
        press_enter_to_continue()

        # Create an optional VPC and create 2 IAM users.
        express_user_name, regular_user_name = self.create_vpc_and_users()

        # Set up two S3 clients, one regular and one express, and two buckets,
        one regular and one express.
        self.setup_clients_and_buckets(express_user_name, regular_user_name)

        # Create an S3 session for the express S3 client and add objects to the
        buckets.
        bucket_object = self.create_session_and_add_objects()

        # Demonstrate performance differences between regular and express
        buckets.
        self.demonstrate_performance(bucket_object)

        # Populate the buckets to show the lexicographical difference between
        regular and express buckets.
        self.show_lexicographical_differences(bucket_object)

        print("")
        print("That's it for our tour of the basic operations for S3 Express One
Zone.")

        if q.ask(
            "Would you like to delete all the resources created during this demo
(y/n)? ",
            q.is_yesno,
        ):

```

```

        self.cleanup()

    def create_vpc_and_users(self) -> None:
        """
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
        without.
        """
        # Configure a gateway VPC endpoint. This is the recommended method to
        allow S3 Express One Zone traffic without
        # the need to pass through an internet gateway or NAT device.
        print(
            """
1. First, we'll set up a new VPC and VPC Endpoint if this program is running in
an EC2 instance in the same AZ as your
Directory buckets will be. Are you running this in an EC2 instance located in the
same AZ as your intended Directory buckets?
"""
        )
        if q.ask("Do you want to setup a VPC Endpoint? (y/n) ", q.is_yesno):
            print(
                "Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to."
            )
            self.setup_vpc()
            press_enter_to_continue()
        else:
            print("Skipping the VPC setup. Don't forget to use this in
production!")
            print(
                """
2. Policies, users, and roles with CDK.
Now, we'll set up some policies, roles, and a user. This user will only have
permissions to do S3 Express One Zone actions.
"""
            )
            press_enter_to_continue()
            stack_name = f"cfn-stack-s3-express-basics--{uuid.uuid4()}"
            template_as_string = S3ExpressScenario.get_template_as_string()
            self.stack = self.deploy_cloudformation_stack(stack_name,
template_as_string)
            regular_user_name = None
            express_user_name = None
            outputs = self.stack.outputs

```



```

    for output in outputs:
        if output.get("OutputKey") == "RegularUser":
            regular_user_name = output.get("OutputValue")
        elif output.get("OutputKey") == "ExpressUser":
            express_user_name = output.get("OutputValue")
    if not regular_user_name or not express_user_name:
        error_string = f"""
        Failed to retrieve required outputs from CloudFormation stack.
        'regular_user_name'={regular_user_name},
'express_user_name'={express_user_name}
        """
        logger.error(error_string)
        raise ValueError(error_string)
    return express_user_name, regular_user_name

def setup_clients_and_buckets(
    self, express_user_name: str, regular_user_name: str
) -> None:
    """
    Set up two S3 clients, one regular and one express, and two buckets, one
    regular and one express.
    :param express_user_name: The name of the user with S3 Express
    permissions.
    :param regular_user_name: The name of the user with regular S3
    permissions.
    """
    regular_credentials = self.create_access_key(regular_user_name)
    express_credentials = self.create_access_key(express_user_name)
    # 3. Create an additional client using the credentials with S3 Express
    permissions.
    print(
        """
3. Create an additional client using the credentials with S3 Express permissions.
This client is created with the
credentials associated with the user account with the S3 Express policy attached,
so it can perform S3 Express operations.
"""
    )
    press_enter_to_continue()
    s3_regular_client = self.create_s3_client_with_access_key_credentials(
        regular_credentials
    )
    self.s3_regular_wrapper = S3ExpressWrapper(s3_regular_client)
    s3_express_client = self.create_s3_client_with_access_key_credentials(

```

```

        express_credentials
    )
    self.s3_express_wrapper = S3ExpressWrapper(s3_express_client)
    print(
        """

```

All the roles and policies were created and attached to the user. Then a new S3 Client were created using that user's credentials. We can now use this client to make calls to S3 Express operations. Keeping permissions in mind (and adhering to least-privilege) is crucial to S3 Express.

```

    """

```

```

    )
    press_enter_to_continue()
    # 4. Create two buckets.
    print(
        """

```

3. Create two buckets.

Now we will create a Directory bucket which is the linchpin of the S3 Express One Zone service. Directory buckets

behave in different ways from regular S3 buckets which we will explore here.

We'll also create a normal bucket, put an object into the normal bucket, and copy it over to the Directory bucket.

```

    """

```

```

    )

```

```

    # Create a directory bucket. These are different from normal S3 buckets
    in subtle ways.

```

```

    bucket_prefix = q.ask(
        "Enter a bucket name prefix that will be used for both buckets: ",
        q.re_match(r"[a-z0-9](?:[a-z0-9-\.]*)[a-z0-9]$"),
    )

```

```

    # Some availability zones are not supported for Directory buckets. We'll
    choose one that is supported.

```

```

    print(
        "Now, let's choose an availability zone for the Directory bucket.

```

```

    We'll choose one that is supported."

```

```

    )

```

```

    while True:

```

```

        availability_zone = self.select_availability_zone_id(self.region)

```

```

        # Construct the parts of a directory bucket name that is made unique
        with a UUID string.

```

```

        directory_bucket_suffix = f"--{availability_zone['ZoneId']}--x-s3"

```

```

        max_uuid_length = 63 - len(bucket_prefix) -
len(directory_bucket_suffix) - 1
        bucket_uuid = str(uuid.uuid4()).replace("-", "")[:max_uuid_length]
        directory_bucket_name = (
            f"{bucket_prefix}-{bucket_uuid}{directory_bucket_suffix}"
        )
        regular_bucket_name = f"{bucket_prefix}-regular-{bucket_uuid}"
        configuration = {
            "Bucket": {
                "Type": "Directory",
                "DataRedundancy": "SingleAvailabilityZone",
            },
            "Location": {
                "Name": availability_zone["ZoneId"],
                "Type": "AvailabilityZone",
            },
        }
        press_enter_to_continue()
        print(
            "Now, let's create the actual Directory bucket, as well as a
regular bucket."
        )
        press_enter_to_continue()
        try:
            self.s3_express_wrapper.create_bucket(
                directory_bucket_name, configuration
            )
            break
        except ClientError as client_error:
            if client_error.response["Error"]["Code"] == "InvalidBucketName":
                print(
                    f"Bucket '{directory_bucket_name}' is invalid. This may
be because of selected availability zone."
                )
                if q.ask(
                    "Would you like to select a different availability zone?"
                ),
                    q.is_yesno,
                ):
                    continue
                else:
                    raise
            else:
                raise

```

```

        print(f"Created directory bucket, '{directory_bucket_name}'")
        self.directory_bucket_name = directory_bucket_name

        self.s3_regular_wrapper.create_bucket(regular_bucket_name)
        print(f"Created regular bucket, '{regular_bucket_name}'")
        self.regular_bucket_name = regular_bucket_name
        print("Great! Both buckets were created.")
        press_enter_to_continue()

    def create_session_and_add_objects(self) -> None:
        """
        Create a session for the express S3 client and add objects to the
        buckets.
        """
        print(
            """
5. Create an object and copy it over.
We'll create a basic object consisting of some text and upload it to the normal
    bucket. Next we'll copy the object
into the Directory bucket using the regular client. This works fine because copy
    operations are not restricted for
Directory buckets.
            """
        )
        press_enter_to_continue()
        bucket_object = "basic-text-object"
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, bucket_object, "Look Ma, I'm a bucket!"
        )
        self.s3_express_wrapper.create_session(self.directory_bucket_name)
        self.s3_express_wrapper.copy_object(
            self.regular_bucket_name,
            bucket_object,
            self.directory_bucket_name,
            bucket_object,
        )
        print(
            """
It worked! It's important to remember the user permissions when interacting with
    Directory buckets. Instead of validating
permissions on every call as normal buckets do, Directory buckets utilize the
    user credentials and session token to validate.
This allows for much faster connection speeds on every call. For single calls,
    this is low, but for many concurrent calls
            """
        )

```

```
this adds up to a lot of time saved.
```

```
"""
```

```
    )
    press_enter_to_continue()
    return bucket_object
```

```
def demonstrate_performance(self, bucket_object: str) -> None:
```

```
    """
```

```
    Demonstrate performance differences between regular and Directory
    buckets.
```

```
    :param bucket_object: The name of the object to download from each
    bucket.
```

```
    """
```

```
    print("")
    print("6. Demonstrate performance difference.")
    print(
        """
```

Now, let's do a performance test. We'll download the same object from each bucket  
'downloads' times

and compare the total time needed. Note: the performance difference will be much  
more pronounced if this

example is run in an EC2 instance in the same Availability Zone as the bucket.

```
"""
```

```
    )
    downloads = 1000
    print(
```

```
        f"The number of downloads of the same object for this example is set
    at {downloads}."
    )
```

```
    if q.ask("Would you like to download a different number? (y/n) ",
    q.is_yesno):
```

```
        max_downloads = 1000000
```

```
        downloads = q.ask(
```

```
            f"Enter a number between 1 and {max_downloads} for the number of
    downloads: ",
```

```
            q.is_int,
            q.in_range(1, max_downloads),
        )
```

```
    # Download the object 'downloads' times from each bucket and time it to
    demonstrate the speed difference.
```

```
    print("Downloading from the Directory bucket.")
```

```
    directory_time_start = time.time_ns()
```

```
    for index in range(downloads):
```

```

        if index % 10 == 0:
            print(f"Download {index} of {downloads}")

        self.s3_express_wrapper.get_object(
            self.directory_bucket_name, bucket_object
        )

    directory_time_difference = time.time_ns() - directory_time_start
    print("Downloading from the normal bucket.")
    normal_time_start = time.time_ns()

    for index in range(downloads):
        if index % 10 == 0:
            print(f"Download {index} of {downloads}")
            self.s3_regular_wrapper.get_object(self.regular_bucket_name,
bucket_object)

        normal_time_difference = time.time_ns() - normal_time_start
        print(
            f"The directory bucket took {directory_time_difference} nanoseconds,
while the normal bucket took {normal_time_difference}."
        )
        difference = normal_time_difference - directory_time_difference
        print(f"That's a difference of {difference} nanoseconds, or")
        print(f"{{(difference) / 1000000000}} seconds.")
        if difference < 0:
            print(
                "The directory buckets were slower. This can happen if you are
not running on the cloud within a vpc."
            )
            press_enter_to_continue()

    def show_lexicographical_differences(self, bucket_object: str) -> None:
        """
        Show the lexicographical difference between Directory buckets and regular
buckets.
        This is done by creating a few objects in each bucket and listing them to
show the difference.
        :param bucket_object: The object to use for the listing operations.
        """
        print(
            """
7. Populate the buckets to show the lexicographical difference.

```

Now let's explore how Directory buckets store objects in a different manner to regular buckets. The key is in the name "Directory". Where regular buckets store their key/value pairs in a flat manner, Directory buckets use actual directories/folders. This allows for more rapid indexing, traversing, and therefore retrieval times! The more segmented your bucket is, with lots of directories, sub-directories, and objects, the more efficient it becomes. This structural difference also causes ListObjects to behave differently, which can cause unexpected results. Let's add a few more objects with layered directories to see how the output of ListObjects changes.

```
""")
    )
    press_enter_to_continue()
    # Populate a few more files in each bucket so that we can use ListObjects
    and show the difference.
    other_object = f"other/{bucket_object}"
    alt_object = f"alt/{bucket_object}"
    other_alt_object = f"other/alt/{bucket_object}"
    self.s3_regular_wrapper.put_object(self.regular_bucket_name,
other_object, "")
    self.s3_express_wrapper.put_object(self.directory_bucket_name,
other_object, "")
    self.s3_regular_wrapper.put_object(self.regular_bucket_name, alt_object,
""")
    self.s3_express_wrapper.put_object(self.directory_bucket_name,
alt_object, "")
    self.s3_regular_wrapper.put_object(
        self.regular_bucket_name, other_alt_object, ""
    )
    self.s3_express_wrapper.put_object(
        self.directory_bucket_name, other_alt_object, ""
    )
    directory_bucket_objects = self.s3_express_wrapper.list_objects(
        self.directory_bucket_name
    )

    regular_bucket_objects = self.s3_regular_wrapper.list_objects(
        self.regular_bucket_name
    )

    print("Directory bucket content")
    for bucket_object in directory_bucket_objects:
        print(f"    {bucket_object['Key']}")
```

```

print("Normal bucket content")
for bucket_object in regular_bucket_objects:
    print(f"    {bucket_object['Key']}")
print(
    """

```

Notice how the normal bucket lists objects in lexicographical order, while the directory bucket does not. This is because the normal bucket considers the whole "key" to be the object identifier, while the directory bucket actually creates directories and uses the object "key" as a path to the object.

```

    """
)
press_enter_to_continue()

def cleanup(self) -> None:
    """
    Delete resources created by this scenario.
    """
    if self.directory_bucket_name is not None:
        self.s3_express_wrapper.delete_bucket_and_objects(
            self.directory_bucket_name
        )
        print(f"Deleted directory bucket, '{self.directory_bucket_name}'")
        self.directory_bucket_name = None

    if self.regular_bucket_name is not None:

self.s3_regular_wrapper.delete_bucket_and_objects(self.regular_bucket_name)
        print(f"Deleted regular bucket, '{self.regular_bucket_name}'")
        self.regular_bucket_name = None

    if self.stack is not None:
        self.destroy_cloudformation_stack(self.stack)
        self.stack = None

    self.tear_down_vpc()

def create_access_key(self, user_name: str) -> dict[str, any]:
    """
    Creates an access key for the user.
    :param user_name: The name of the user.
    :return: The access key for the user.
    """
    try:

```



```

        access_key = self.iam_client.create_access_key(UserName=user_name)
        return access_key["AccessKey"]
    except ClientError as client_error:
        logging.error(
            "Couldn't create the access key. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def create_s3_client_with_access_key_credentials(
    self, access_key: dict[str, any]
) -> client:
    """
    Creates an S3 client with access key credentials.
    :param access_key: The access key for the user.
    :return: The S3 Express One Zone client.
    """
    try:
        s3_express_client = boto3.client(
            "s3",
            aws_access_key_id=access_key["AccessKeyId"],
            aws_secret_access_key=access_key["SecretAccessKey"],
            region_name=self.region,
        )
        return s3_express_client
    except ClientError as client_error:
        logging.error(
            "Couldn't create the S3 Express One Zone client. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def select_availability_zone_id(self, region: str) -> dict[str, any]:
    """
    Selects an availability zone.
    :param region: The region to select the availability zone from.
    :return: The availability zone dictionary.
    """
    try:
        response = self.ec2_client.describe_availability_zones(
            Filters=[{"Name": "region-name", "Values": [region]}]
        )
        availability_zones = response["AvailabilityZones"]
        zone_names = [zone["ZoneName"] for zone in availability_zones]

```

```

        index = q.choose("Select an availability zone: ", zone_names)
        return availability_zones[index]
    except ClientError as client_error:
        logging.error(
            "Couldn't describe availability zones. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def deploy_cloudformation_stack(
    self, stack_name: str, cfn_template: str
) -> ServiceResource:
    """
    Deploys prerequisite resources used by the scenario. The resources are
    defined in the associated `cfn_template.yaml` AWS CloudFormation script
    and are deployed
    as a CloudFormation stack, so they can be easily managed and destroyed.

    :param stack_name: The name of the CloudFormation stack.
    :param cfn_template: The CloudFormation template as a string.
    :return: The CloudFormation stack resource.
    """
    print(f"Deploying CloudFormation stack: {stack_name}.")
    stack = self.cloud_formation_resource.create_stack(
        StackName=stack_name,
        TemplateBody=cfn_template,
        Capabilities=["CAPABILITY_NAMED_IAM"],
    )
    print(f"CloudFormation stack creation started: {stack_name}")
    print("Waiting for CloudFormation stack creation to complete...")
    waiter = self.cloud_formation_resource.meta.client.get_waiter(
        "stack_create_complete"
    )
    waiter.wait(StackName=stack.name)
    stack.load()
    print("CloudFormation stack creation complete.")

    return stack

def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
    """
    Destroys the resources managed by the CloudFormation stack, and the
    CloudFormation
    stack itself.

```

```

        :param stack: The CloudFormation stack that manages the example
resources.
        """
        try:
            print(
                f"CloudFormation stack '{stack.name}' is being deleted. This may
take a few minutes."
            )
            stack.delete()
            waiter = self.cloud_formation_resource.meta.client.get_waiter(
                "stack_delete_complete"
            )
            waiter.wait(StackName=stack.name)
            print(f"CloudFormation stack '{stack.name}' has been deleted.")
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the CloudFormation stack. Here's why: %s",
                client_error.response["Error"]["Message"],
            )

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory,
"s3_express_template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def setup_vpc(self):
        cidr = "10.0.0.0/16"
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr)
            self.vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[self.vpc_id])
            print(f"Created vpc {self.vpc_id}")

        except ClientError as client_error:
            logging.error(

```

```

        "Couldn't create the vpc. Here's why: %s",
        client_error.response["Error"]["Message"],
    )
    raise
try:
    response = self.ec2_client.describe_route_tables(
        Filters=[{"Name": "vpc-id", "Values": [self.vpc_id]}]
    )
    route_table_id = response["RouteTables"][0]["RouteTableId"]
    service_name = f"com.amazonaws.
{self.ec2_client.meta.region_name}.s3express"

    response = self.ec2_client.create_vpc_endpoint(
        VpcId=self.vpc_id,
        RouteTableIds=[route_table_id],
        ServiceName=service_name,
    )
    self.vpc_endpoint_id = response["VpcEndpoint"]["VpcEndpointId"]
    print(f"Created vpc endpoint {self.vpc_endpoint_id}")

except ClientError as client_error:
    logging.error(
        "Couldn't create the vpc endpoint. Here's why: %s",
        client_error.response["Error"]["Message"],
    )
    raise

def tear_down_vpc(self) -> None:
    if self.vpc_endpoint_id is not None:
        try:
            self.ec2_client.delete_vpc_endpoints(
                VpcEndpointIds=[self.vpc_endpoint_id]
            )
            print(f"Deleted vpc endpoint {self.vpc_endpoint_id}.")
            self.vpc_endpoint_id = None
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the vpc endpoint %s. Here's why: %s",
                self.vpc_endpoint_id,
                client_error.response["Error"]["Message"],
            )
    if self.vpc_id is not None:
        try:
            self.ec2_client.delete_vpc(VpcId=self.vpc_id)

```

```

        print(f"Deleted vpc {self.vpc_id}")
        self.vpc_id = None
    except ClientError as client_error:
        logging.error(
            "Couldn't delete the vpc %s. Here's why: %s",
            self.vpc_id,
            client_error.response["Error"]["Message"],
        )

```

Amazon S3 Express SDK 関数のラッパークラス。

```

class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
    interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                           access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def create_bucket(
        self, bucket_name: str, bucket_configuration: dict[str, any] = None
    ) -> None:

```

```

"""
Creates a bucket.
:param bucket_name: The name of the bucket.
:param bucket_configuration: The optional configuration for the bucket.
"""
try:
    params = {"Bucket": bucket_name}
    if bucket_configuration:
        params["CreateBucketConfiguration"] = bucket_configuration

    self.s3_client.create_bucket(**params)
except ClientError as client_error:
    # Do not log InvalidBucketName error because it is logged elsewhere.
    if client_error.response["Error"]["Code"] != "InvalidBucketName":
        logging.error(
            "Couldn't create the bucket %s. Here's why: %s",
            bucket_name,
            client_error.response["Error"]["Message"],
        )
    raise

def delete_bucket_and_objects(self, bucket_name: str) -> None:
    """
    Deletes a bucket and its objects.
    :param bucket_name: The name of the bucket.
    """
    try:
        # Delete the objects in the bucket first. This is required for a
        bucket to be deleted.
        paginator = self.s3_client.get_paginator("list_objects_v2")
        page_iterator = paginator.paginate(Bucket=bucket_name)
        for page in page_iterator:
            if "Contents" in page:
                delete_keys = {
                    "Objects": [{"Key": obj["Key"]} for obj in
page["Contents"]]
                }
                response = self.s3_client.delete_objects(
                    Bucket=bucket_name, Delete=delete_keys
                )
                if "Errors" in response:
                    for error in response["Errors"]:
                        logging.error(
                            "Couldn't delete object %s. Here's why: %s",

```

```
        error["Key"],
        error["Message"],
    )

    self.s3_client.delete_bucket(Bucket=bucket_name)
except ClientError as client_error:
    logging.error(
        "Couldn't delete the bucket %s. Here's why: %s",
        bucket_name,
        client_error.response["Error"]["Message"],
    )

def put_object(self, bucket_name: str, object_key: str, content: str) ->
None:
    """
    Puts an object into a bucket.
    :param bucket_name: The name of the bucket.
    :param object_key: The key of the object.
    :param content: The content of the object.
    """
    try:
        self.s3_client.put_object(Body=content, Bucket=bucket_name,
Key=object_key)
    except ClientError as client_error:
        logging.error(
            "Couldn't put the object %s into bucket %s. Here's why: %s",
            object_key,
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise

def list_objects(self, bucket: str) -> list[str]:
    """
    Lists objects in a bucket.
    :param bucket: The name of the bucket.
    :return: The list of objects in the bucket.
    """
    try:
        response = self.s3_client.list_objects_v2(Bucket=bucket)
        return response.get("Contents", [])
    except ClientError as client_error:
        logging.error(
            "Couldn't list objects in bucket %s. Here's why: %s",
```

```

        bucket,
        client_error.response["Error"]["Message"],
    )
    raise

def copy_object(
    self,
    source_bucket: str,
    source_key: str,
    destination_bucket: str,
    destination_key: str,
) -> None:
    """
    Copies an object from one bucket to another.
    :param source_bucket: The source bucket.
    :param source_key: The source key.
    :param destination_bucket: The destination bucket.
    :param destination_key: The destination key.
    :return: None
    """
    try:
        self.s3_client.copy_object(
            CopySource={"Bucket": source_bucket, "Key": source_key},
            Bucket=destination_bucket,
            Key=destination_key,
        )
    except ClientError as client_error:
        logging.error(
            "Couldn't copy object %s from bucket %s to bucket %s. Here's why:
%s",
            source_key,
            source_bucket,
            destination_bucket,
            client_error.response["Error"]["Message"],
        )
        raise

def create_session(self, bucket_name: str) -> None:
    """
    Creates an express session.
    :param bucket_name: The name of the bucket.
    """
    try:
        self.s3_client.create_session(Bucket=bucket_name)

```



```
except ClientError as client_error:
    logging.error(
        "Couldn't create the express session for bucket %s. Here's why:
%s",
        bucket_name,
        client_error.response["Error"]["Message"],
    )
    raise

def get_object(self, bucket_name: str, object_key: str) -> None:
    """
    Gets an object from a bucket.
    :param bucket_name: The name of the bucket.
    :param object_key: The key of the object.
    """
    try:
        self.s3_client.get_object(Bucket=bucket_name, Key=object_key)
    except ClientError as client_error:
        logging.error(
            "Couldn't get the object %s from bucket %s. Here's why: %s",
            object_key,
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise
```

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

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObject](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用した S3 ディレクトリバケットのアクション

以下は、AWS SDK を使用して個々の S3 ディレクトリバケットアクションを実行する方法を説明するコード例です。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

これらは S3 Directory Buckets API を呼び出すもので、コンテキスト内で実行する必要がある大規模なプログラムからのコードの抜粋です。アクションは [AWS SDK を使用する S3 ディレクトリバケットのシナリオ](#) のコンテキスト内で確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。完全なリストについては、「[Amazon S3 ディレクトリバケット API リファレンス](#)」を参照してください。

### 例

- [AWS SDK で AbortMultipartUpload を使用する](#)
- [AWS SDK で CompleteMultipartUpload を使用する](#)
- [AWS SDK で CopyObject を使用する](#)
- [AWS SDK で CreateBucket を使用する](#)
- [AWS SDK で CreateMultipartUpload を使用する](#)
- [AWS SDK で CreateSession を使用する](#)
- [AWS SDK で DeleteBucket を使用する](#)
- [AWS SDK で DeleteBucketEncryption を使用する](#)
- [AWS SDK で DeleteBucketPolicy を使用する](#)
- [AWS SDK で DeleteObject を使用する](#)
- [AWS SDK で DeleteObjects を使用する](#)
- [AWS SDK で GetBucketEncryption を使用する](#)
- [AWS SDK で GetBucketPolicy を使用する](#)
- [AWS SDK で GetObject を使用する](#)
- [AWS SDK で GetObjectAttributes を使用する](#)
- [AWS SDK で HeadBucket を使用する](#)

- [AWS SDK で HeadObject を使用する](#)
- [AWS SDK で ListDirectoryBuckets を使用する](#)
- [AWS SDK で ListMultipartUploads を使用する](#)
- [AWS SDK で ListObjectsV2 を使用する](#)
- [AWS SDK で ListParts を使用する](#)
- [AWS SDK で PutBucketEncryption を使用する](#)
- [AWS SDK で PutBucketPolicy を使用する](#)
- [AWS SDK で PutObject を使用する](#)
- [AWS SDK で UploadPart を使用する](#)
- [AWS SDK で UploadPartCopy を使用する](#)

## AWS SDK で **AbortMultipartUpload** を使用する

次の例は、AbortMultipartUpload を使用方法を説明しています。

Java

SDK for Java 2.x

### Note

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

ディレクトリバケット内のマルチパートアップロードを中止します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Aborts a specific multipart upload for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be uploaded
 * @param uploadId The upload ID of the multipart upload to abort
 * @return True if the multipart upload is successfully aborted, false
 * otherwise
 */
public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
String bucketName,
String objectKey, String uploadId) {
    logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
bucketName);
    try {
        // Abort the multipart upload
        AbortMultipartUploadRequest abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .uploadId(uploadId)
            .build();

        s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        logger.info("Aborted multipart upload: {} for object: {}", uploadId,
objectKey);
        return true;
    } catch (S3Exception e) {
        logger.error("Failed to abort multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **CompleteMultipartUpload** を使用する

次の例は、CompleteMultipartUpload を使用する方法を説明しています。

### Java

#### SDK for Java 2.x

#### Note

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

ディレクトリバケット内のマルチパートアップロードを完了します。

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * This method completes the multipart upload request by collating all the
 * upload parts.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param objectKey   The key (name) of the object to be uploaded
 * @param uploadId    The upload ID used to track the multipart upload
 * @param uploadParts The list of completed parts
 * @return True if the multipart upload is successfully completed, false
 *         otherwise
 */
public static boolean completeDirectoryBucketMultipartUpload(S3Client
s3Client, String bucketName, String objectKey,
String uploadId, List<CompletedPart> uploadParts) {
    try {
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
            .parts(uploadParts)
            .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .uploadId(uploadId)
            .multipartUpload(completedMultipartUpload)
            .build();

        CompleteMultipartUploadResponse response =
s3Client.completeMultipartUpload(completeMultipartUploadRequest);
        logger.info("Multipart upload completed. ETag: {}", response.eTag());
        return true;
    } catch (S3Exception e) {
        logger.error("Failed to complete multipart upload: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **CopyObject** を使用する

次の例は、CopyObject を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

## Java

### SDK for Java 2.x

#### Note

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

ディレクトリバケットから別のディレクトリバケットにオブジェクトをコピーします。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Copies an object from one S3 general purpose bucket to one S3 directory
 * bucket.
 *
 * @param s3Client      The S3 client used to interact with S3
 * @param sourceBucket  The name of the source bucket
 * @param objectKey     The key (name) of the object to be copied
 * @param targetBucket  The name of the target bucket
 */
public static void copyDirectoryBucketObject(S3Client s3Client, String
sourceBucket, String objectKey,
        String targetBucket) {
    logger.info("Copying object: {} from bucket: {} to bucket: {}",
objectKey, sourceBucket, targetBucket);

    try {
        // Create a CopyObjectRequest
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(objectKey)
            .destinationBucket(targetBucket)
            .destinationKey(objectKey)
            .build();

        // Copy the object
        CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
        logger.info("Successfully copied {} from bucket {} into bucket {}.
CopyObjectResponse: {}",
            objectKey, sourceBucket, targetBucket,
copyRes.copyObjectResult().toString());

    } catch (S3Exception e) {
        logger.error("Failed to copy object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
    }
}
```



```
        throw e;
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[CopyObject](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **CreateBucket** を使用する

次の例は、CreateBucket を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

## Java

### SDK for Java 2.x

#### Note

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

S3 ディレクトリバケットを作成します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
```

```
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client The S3 client used to create the bucket
 * @param bucketName The name of the bucket to be created
 * @param zone The region where the bucket will be created
 * @throws S3Exception if there's an error creating the bucket
 */
public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
    logger.info("Creating bucket: {}", bucketName);

    CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
        .location(LocationInfo.builder()
            .type(LocationType.AVAILABILITY_ZONE)
            .name(zone).build())
        .bucket(BucketInfo.builder()
            .type(BucketType.DIRECTORY)
            .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
            .build())
        .build();

    try {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration).build();
        CreateBucketResponse response = s3Client.createBucket(bucketRequest);
        logger.info("Bucket created successfully with location: {}",
response.location());
    } catch (S3Exception e) {
        logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
    }
}
```

```
        throw e;
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **CreateMultipartUpload** を使用する

次の例は、CreateMultipartUpload を使用する方法を説明しています。

Java

SDK for Java 2.x

### Note

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

ディレクトリバケットにマルチパートアップロードを作成します。

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
/**
 * This method creates a multipart upload request that generates a unique
upload
 * ID used to track
 * all the upload parts.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be uploaded
 * @return The upload ID used to track the multipart upload
 */
public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
String bucketName, String objectKey) {
    logger.info("Creating multipart upload for object: {} in bucket: {}",
objectKey, bucketName);

    try {
        // Create a CreateMultipartUploadRequest
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Initiate the multipart upload
        CreateMultipartUploadResponse response =
s3Client.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
        return uploadId;

    } catch (S3Exception e) {
        logger.error("Failed to create multipart upload: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **CreateSession** を使用する

次の例は、CreateSession を使用する方法を説明しています。

### Python

#### SDK for Python (Boto3)

#### Note

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

```
class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
    interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                           access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)
```

```
def create_session(self, bucket_name: str) -> None:
    """
    Creates an express session.
    :param bucket_name: The name of the bucket.
    """
    try:
        self.s3_client.create_session(Bucket=bucket_name)
    except ClientError as client_error:
        logging.error(
            "Couldn't create the express session for bucket %s. Here's why:
%s",
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **DeleteBucket** を使用する

次の例は、DeleteBucket を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

## Java

### SDK for Java 2.x

#### Note

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

S3 ディレクトリバケットを削除します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Deletes the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket to delete
 */
public static void deleteDirectoryBucket(S3Client s3Client, String
bucketName) {
    logger.info("Deleting bucket: {}", bucketName);

    try {
        // Create a DeleteBucketRequest
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Delete the bucket
        s3Client.deleteBucket(deleteBucketRequest);
        logger.info("Successfully deleted bucket: {}", bucketName);
    }
```

```
    } catch (S3Exception e) {  
        logger.error("Failed to delete bucket: {} - Error code: {}",  
            e.awsErrorDetails().errorMessage(),  
            e.awsErrorDetails().errorCode(), e);  
        throw e;  
    }  
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[DeleteBucket](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **DeleteBucketEncryption** を使用する

次の例は、DeleteBucketEncryption を使用する方法を説明しています。

Java

SDK for Java 2.x

### Note

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

ディレクトリバケットの暗号化設定を削除します。

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```



```
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Deletes the encryption configuration from an S3 bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 */
public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
    DeleteBucketEncryptionRequest deleteRequest =
DeleteBucketEncryptionRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3Client.deleteBucketEncryption(deleteRequest);
        logger.info("Bucket encryption deleted for bucket: {}", bucketName);
    } catch (S3Exception e) {
        logger.error("Failed to delete bucket encryption: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **DeleteBucketPolicy** を使用する

次の例は、DeleteBucketPolicy を使用する方法を説明しています。

## Java

## SDK for Java 2.x

 Note

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

ディレクトリバケット内のバケットポリシーを削除します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Deletes the bucket policy for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 */
public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
    logger.info("Deleting policy for bucket: {}", bucketName);

    try {
        // Create a DeleteBucketPolicyRequest
        DeleteBucketPolicyRequest deletePolicyReq =
DeleteBucketPolicyRequest.builder()
```

```
        .bucket(bucketName)
        .build();

// Delete the bucket policy
s3Client.deleteBucketPolicy(deletePolicyReq);
logger.info("Successfully deleted bucket policy");

    } catch (S3Exception e) {
        logger.error("Failed to delete bucket policy: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[DeleteBucketPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **DeleteObject** を使用する

次の例は、DeleteObject を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

### Java

#### SDK for Java 2.x

#### Note

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

ディレクトリバケット内のオブジェクトを削除します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Deletes an object from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be deleted
 */
public static void deleteDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Deleting object: {} from bucket: {}", objectKey,
bucketName);

    try {
        // Create a DeleteObjectRequest
        DeleteObjectRequest deleteObjectRequest =
DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Delete the object
        s3Client.deleteObject(deleteObjectRequest);
    }
}
```

```
        logger.info("Object {} has been deleted", objectKey);

    } catch (S3Exception e) {
        logger.error("Failed to delete object: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[DeleteObject](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **DeleteObjects** を使用する

次の例は、DeleteObjects を使用する方法を説明しています。

### Java

#### SDK for Java 2.x

#### Note

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

ディレクトリバケット内の複数のオブジェクトを削除します。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Deletes multiple objects from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKeys The list of keys (names) of the objects to be deleted
 */
public static void deleteDirectoryBucketObjects(S3Client s3Client, String
bucketName, List<String> objectKeys) {
    logger.info("Deleting objects from bucket: {}", bucketName);

    try {
        // Create a list of ObjectIdentifier.
        List<ObjectIdentifier> identifiers = objectKeys.stream()
            .map(key -> ObjectIdentifier.builder().key(key).build())
            .toList();

        Delete delete = Delete.builder()
            .objects(identifiers)
            .build();

        DeleteObjectsRequest deleteObjectsRequest =
DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(delete)
            .build();
```

```
        DeleteObjectsResponse deleteObjectsResponse =
s3Client.deleteObjects(deleteObjectsRequest);
        deleteObjectsResponse.deleted().forEach(deleted ->
logger.info("Deleted object: {}", deleted.key()));

    } catch (S3Exception e) {
        logger.error("Failed to delete objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[DeleteObjects](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **GetBucketEncryption** を使用する

次の例は、GetBucketEncryption を使用する方法を説明しています。

Java

SDK for Java 2.x

### Note

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

ディレクトリバケットの暗号化設定を取得します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Retrieves the encryption configuration for an S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return The type of server-side encryption applied to the bucket (e.g.,
 *         AES256, aws:kms)
 */
public static String getDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
    try {
        // Create a GetBucketEncryptionRequest
        GetBucketEncryptionRequest getRequest =
GetBucketEncryptionRequest.builder()
            .bucket(bucketName)
            .build();

        // Retrieve the bucket encryption configuration
        GetBucketEncryptionResponse response =
s3Client.getBucketEncryption(getRequest);
        ServerSideEncryptionRule rule =
response.serverSideEncryptionConfiguration().rules().get(0);

        String encryptionType =
rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
        logger.info("Bucket encryption algorithm: {}", encryptionType);
        logger.info("KMS Customer Managed Key ID: {}",
rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
        logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

        return encryptionType;
    } catch (S3Exception e) {
        logger.error("Failed to get bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
    }
}
```



```
        throw e;
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **GetBucketPolicy** を使用する

次の例は、GetBucketPolicy を使用する方法を説明しています。

Java

SDK for Java 2.x

### Note

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

ディレクトリバケットのポリシーを取得します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
```

```
import static
com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

/**
 * Retrieves the bucket policy for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return The bucket policy text
 */
public static String getDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
    logger.info("Getting policy for bucket: {}", bucketName);

    try {
        // Create a GetBucketPolicyRequest
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        // Retrieve the bucket policy
        GetBucketPolicyResponse response =
s3Client.getBucketPolicy(policyReq);

        // Print and return the policy text
        String policyText = response.policy();
        logger.info("Bucket policy: {}", policyText);
        return policyText;

    } catch (S3Exception e) {
        logger.error("Failed to get bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **GetObject** を使用する

次の例は、GetObject を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [オブジェクトを取得する署名付き URL を作成する](#)

## Java

### SDK for Java 2.x

#### Note

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

ディレクトリバケットからオブジェクトを取得します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
```

```
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves an object from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be retrieved
 * @return The retrieved object as a ResponseInputStream
 */
public static boolean getDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Retrieving object: {} from bucket: {}", objectKey,
bucketName);

    try {
        // Create a GetObjectRequest
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(objectKey)
            .bucket(bucketName)
            .build();

        // Retrieve the object as bytes
        ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Print object contents to console
        String objectContent = new String(data, StandardCharsets.UTF_8);
        logger.info("Object contents: \n{}", objectContent);

        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

```
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[GetObject](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **GetObjectAttributes** を使用する

次の例は、GetObjectAttributes を使用する方法を説明しています。

Java

SDK for Java 2.x

### Note

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

ディレクトリバケットからオブジェクト属性を取得します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves attributes for an object in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to retrieve attributes for
 * @return True if the object attributes are successfully retrieved, false
 *         otherwise
 */
public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
String bucketName, String objectKey) {
    logger.info("Retrieving attributes for object: {} from bucket: {}",
objectKey, bucketName);

    try {
        // Create a GetObjectAttributesRequest
        GetObjectAttributesRequest getObjectAttributesRequest =
GetObjectAttributesRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .objectAttributes(ObjectAttributes.E_TAG,
ObjectAttributes.STORAGE_CLASS,
                ObjectAttributes.OBJECT_SIZE)
            .build();

        // Retrieve the object attributes
        GetObjectAttributesResponse response =
s3Client.getObjectAttributes(getObjectAttributesRequest);
        logger.info("Attributes for object {}:", objectKey);
        logger.info("ETag: {}", response.eTag());
        logger.info("Storage Class: {}", response.storageClass());
        logger.info("Object Size: {}", response.objectSize());
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object attributes: {} - Error code:
{}",
            e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
    }
}
```

```
        return false;
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で HeadBucket を使用する

次の例は、HeadBucket を使用する方法を説明しています。

### Java

#### SDK for Java 2.x

#### Note

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

指定された S3 ディレクトリバケットが存在し、アクセス可能であるかどうかを確認します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
```

```
* Checks if the specified S3 directory bucket exists and is accessible.
*
* @param s3Client    The S3 client used to interact with S3
* @param bucketName The name of the directory bucket to check
* @return True if the bucket exists and is accessible, false otherwise
*/
public static boolean headDirectoryBucket(S3Client s3Client, String
bucketName) {
    logger.info("Checking if bucket exists: {}", bucketName);

    try {
        // Create a HeadBucketRequest
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();
        // If the bucket doesn't exist, the following statement throws
        NoSuchBucketException,
        // which is a subclass of S3Exception.
        s3Client.headBucket(headBucketRequest);
        logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to access bucket: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **HeadObject** を使用する

次の例は、HeadObject を使用する方法を説明しています。



## Java

## SDK for Java 2.x

 Note

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

ディレクトリバケット内のオブジェクトのメタデータを取得します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves metadata for an object in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to retrieve metadata for
 * @return True if the object exists, false otherwise
 */
public static boolean headDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
```

```
logger.info("Retrieving metadata for object: {} from bucket: {}",
objectKey, bucketName);

try {
    // Create a HeadObjectRequest
    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .build();

    // Retrieve the object metadata
    HeadObjectResponse response = s3Client.headObject(headObjectRequest);
    logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with
ETag: \"{}\"", objectKey, bucketName,
        response.eTag());
    logger.info("Content-Type: {}", response.contentType());
    logger.info("Content-Length: {}", response.contentLength());
    logger.info("Last Modified: {}", response.lastModified());
    return true;

} catch (S3Exception e) {
    logger.error("Failed to retrieve object metadata: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
    return false;
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **ListDirectoryBuckets** を使用する

次の例は、ListDirectoryBuckets を使用する方法を説明しています。

## Java

### SDK for Java 2.x

#### Note

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

すべてのディレクトリバケットを一覧表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Lists all S3 directory buckets and no general purpose buckets.
 *
 * @param s3Client The S3 client used to interact with S3
 * @return A list of bucket names
 */
public static List<String> listDirectoryBuckets(S3Client s3Client) {
    logger.info("Listing all directory buckets");

    try {
        // Create a ListBucketsRequest
        ListDirectoryBucketsRequest listDirectoryBucketsRequest =
            ListDirectoryBucketsRequest.builder().build();
```

```
// Retrieve the list of buckets
ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

// Extract bucket names
List<String> bucketNames = response.buckets().stream()
    .map(Bucket::name)
    .collect(Collectors.toList());

return bucketNames;
} catch (S3Exception e) {
    logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode());
    throw e;
}
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **ListMultipartUploads** を使用する

次の例は、ListMultipartUploads を使用する方法を説明しています。

Java

SDK for Java 2.x

### Note

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

ディレクトリバケット内のマルチパートアップロードを一覧表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Lists multipart uploads for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return A list of MultipartUpload objects representing the multipart
 uploads
 */
public static List<MultipartUpload>
listDirectoryBucketMultipartUploads(S3Client s3Client, String bucketName) {
    logger.info("Listing in-progress multipart uploads for bucket: {}",
bucketName);

    try {
        // Create a ListMultipartUploadsRequest
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
```

```
        .bucket(bucketName)
        .build();

    // List the multipart uploads
    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();
    for (MultipartUpload upload : uploads) {
        logger.info("In-progress multipart upload: Upload ID: {}, Key:
        {}, Initiated: {}", upload.uploadId(),
            upload.key(), upload.initiated());
    }
    return uploads;

    } catch (S3Exception e) {
        logger.error("Failed to list multipart uploads: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        return List.of(); // Return an empty list if an exception is thrown
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK で **ListObjectsV2** を使用する

次の例は、ListObjectsV2 を使用する方法を説明しています。

## Java

## SDK for Java 2.x

 Note

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

ディレクトリバケット内のオブジェクトを一覧表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Lists objects in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return A list of object keys in the bucket
 */
```

```
public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
String bucketName) {
    logger.info("Listing objects in bucket: {}", bucketName);

    try {
        // Create a ListObjectsV2Request
        ListObjectsV2Request listObjectsV2Request =
ListObjectsV2Request.builder()
                        .bucket(bucketName)
                        .build();

        // Retrieve the list of objects
        ListObjectsV2Response response =
s3Client.listObjectsV2(listObjectsV2Request);

        // Extract and return the object keys
        return response.contents().stream()
                        .map(S3Object::key)
                        .collect(Collectors.toList());

    } catch (S3Exception e) {
        logger.error("Failed to list objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                        e.awsErrorDetails().errorCode());
        throw e;
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[ListObjectsV2](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **ListParts** を使用する

次の例は、ListParts を使用する方法を説明しています。



## Java

## SDK for Java 2.x

 Note

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

ディレクトリバケット内のマルチパートアップロードの一部を一覧表示します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Lists the parts of a multipart upload for the specified S3 directory
 * bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
```

```
* @param objectKey The key (name) of the object being uploaded
* @param uploadId The upload ID used to track the multipart upload
* @return A list of Part representing the parts of the multipart upload
*/
public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
s3Client, String bucketName,
    String objectKey, String uploadId) {
    logger.info("Listing parts for object: {} in bucket: {}", objectKey,
bucketName);

    try {
        // Create a ListPartsRequest
        ListPartsRequest listPartsRequest = ListPartsRequest.builder()
            .bucket(bucketName)
            .uploadId(uploadId)
            .key(objectKey)
            .build();

        // List the parts of the multipart upload
        ListPartsResponse response = s3Client.listParts(listPartsRequest);
        List<Part> parts = response.parts();
        for (Part part : parts) {
            logger.info("Uploaded part: Part number = \"{}\", etag = {}",
part.partNumber(), part.eTag());
        }
        return parts;

    } catch (S3Exception e) {
        logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        return List.of(); // Return an empty list if an exception is thrown
    }
}
```

- APIの詳細については、AWS SDK for Java 2.x API リファレンスの「[ListParts](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **PutBucketEncryption** を使用する

次の例は、PutBucketEncryption を使用する方法を説明しています。

Java

SDK for Java 2.x

### Note

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

バケット暗号化をディレクトリバケットに設定します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import
    software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

/**
 * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param kmsKeyId    The ID of the customer-managed KMS key
 */
```

```
public static void putDirectoryBucketEncryption(S3Client s3Client, String
bucketName, String kmsKeyId) {
    // Define the default encryption configuration to use SSE-KMS. For
    directory
    // buckets, AWS managed KMS keys aren't supported. Only customer-managed
    keys
    // are supported.
    ServerSideEncryptionByDefault encryptionByDefault =
    ServerSideEncryptionByDefault.builder()
        .sseAlgorithm(ServerSideEncryption.AWS_KMS)
        .kmsMasterKeyID(kmsKeyId)
        .build();

    // Create a server-side encryption rule to apply the default encryption
    // configuration. For directory buckets, the bucketKeyEnabled field is
    enforced
    // to be true.
    ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
        .bucketKeyEnabled(true)
        .applyServerSideEncryptionByDefault(encryptionByDefault)
        .build();

    // Create the server-side encryption configuration for the bucket
    ServerSideEncryptionConfiguration encryptionConfiguration =
    ServerSideEncryptionConfiguration.builder()
        .rules(rule)
        .build();

    // Create the PutBucketEncryption request
    PutBucketEncryptionRequest putRequest =
    PutBucketEncryptionRequest.builder()
        .bucket(bucketName)
        .serverSideEncryptionConfiguration(encryptionConfiguration)
        .build();

    // Set the bucket encryption
    try {
        s3Client.putBucketEncryption(putRequest);
        logger.info("SSE-KMS Bucket encryption configuration set for the
        directory bucket: {}", bucketName);
    } catch (S3Exception e) {
        logger.error("Failed to set bucket encryption: {} - Error code: {}",
        e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode());
    }
}
```

```
        throw e;
    }
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **PutBucketPolicy** を使用する

次の例は、PutBucketPolicy を使用する方法を説明しています。

Java

SDK for Java 2.x

### Note

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

バケットポリシーをディレクトリバケットに適用します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

/**
```

```

* Sets the following bucket policy for the specified S3 directory bucket.
*<pre>
* {
*     "Version":"2012-10-17",
*     "Statement": [
*         {
*             "Sid": "AdminPolicy",
*             "Effect": "Allow",
*             "Principal": {
*                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
*             },
*             "Action": "s3express:*",
*             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
*         }
*     ]
* }
* </pre>
* This policy grants all S3 directory bucket actions to identities in the
same account as the bucket.
*
* @param s3Client    The S3 client used to interact with S3
* @param bucketName The name of the directory bucket
* @param policyText The policy text to be applied
*/
public static void putDirectoryBucketPolicy(S3Client s3Client, String
bucketName, String policyText) {
    logger.info("Setting policy on bucket: {}", bucketName);
    logger.info("Policy: {}", policyText);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
            .bucket(bucketName)
            .policy(policyText)
            .build();

        s3Client.putBucketPolicy(policyReq);
        logger.info("Bucket policy set successfully!");

    } catch (S3Exception e) {
        logger.error("Failed to set bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

```

```
}  
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[PutBucketPolicy](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **PutObject** を使用する

次の例は、PutObject を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

## Java

### SDK for Java 2.x

#### Note

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

ディレクトリバケットにオブジェクトを配置します。

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.PutObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import java.io.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

/**
 * Puts an object into the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be placed in the bucket
 * @param filePath The path of the file to be uploaded
 */
public static void putDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey, Path filePath) {
    logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);

    try {
        // Create a PutObjectRequest
        PutObjectRequest putObj = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Upload the object
        s3Client.putObject(putObj, filePath);
        logger.info("Successfully placed {} into bucket {}", objectKey,
bucketName);

    } catch (UncheckedIOException e) {
        throw S3Exception.builder().message("Failed to read the file: " +
e.getMessage()).cause(e)
            .awsErrorDetails(AwsErrorDetails.builder()
                .errorCode("ClientSideException:FailedToReadFile")
                .errorMessage(e.getMessage())
                .build())
            .build();
    } catch (S3Exception e) {
        logger.error("Failed to put object: {}", e.getMessage(), e);
    }
}
```



```
        throw e;
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[PutObject](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **UploadPart** を使用する

次の例は、UploadPart を使用する方法を説明しています。

Java

SDK for Java 2.x

### Note

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

ディレクトリバケット内のマルチパートアップロードの一部をアップロードします。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
```

```
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

/**
 * This method creates part requests and uploads individual parts to S3.
 * While it uses the UploadPart API to upload a single part, it does so
 * sequentially to handle multiple parts of a file, returning all the
 * completed
 * parts.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param objectKey   The key (name) of the object to be uploaded
 * @param uploadId    The upload ID used to track the multipart upload
 * @param filePath    The path to the file to be uploaded
 * @return A list of uploaded parts
 * @throws IOException if an I/O error occurs
 */
public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
s3Client, String bucketName,
    String objectKey, String uploadId, Path filePath) throws IOException
{
    logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
bucketName);

    int partNumber = 1;
    List<CompletedPart> uploadedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    // Read the local file, break down into chunks and process
    try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(),
"r")) {
```

```
        long fileSize = file.length();
        int position = 0;

        // Sequentially upload parts of the file
        while (position < fileSize) {
            file.seek(position);
            int read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer
            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .uploadId(uploadId)
                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
                RequestBody.fromByteBuffer(bb));

            // Build the uploaded part
            CompletedPart uploadedPart = CompletedPart.builder()
                .partNumber(partNumber)
                .eTag(partResponse.eTag())
                .build();

            // Add the uploaded part to the list
            uploadedParts.add(uploadedPart);

            // Log to indicate the part upload is done
            logger.info("Uploaded part number: {} with ETag: {}", partNumber,
partResponse.eTag());

            bb.clear();
            position += read;
            partNumber++;
        }
    } catch (S3Exception e) {
        logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        throw e;
    }
}
```

```
        return uploadedParts;
    }
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK で **UploadPartCopy** を使用する

次の例は、UploadPartCopy を使用する方法を説明しています。

Java

SDK for Java 2.x

### Note

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

ソースオブジェクトのサイズに基づいてコピーパーツを作成し、個々のパーツをディレクトリバケットにコピーします。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;
import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
```

```
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
    com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Creates copy parts based on source object size and copies over individual
 * parts.
 *
 * @param s3Client      The S3 client used to interact with S3
 * @param sourceBucket  The name of the source bucket
 * @param sourceKey      The key (name) of the source object
 * @param destinationBucket The name of the destination bucket
 * @param destinationKey The key (name) of the destination object
 * @param uploadId      The upload ID used to track the multipart upload
 * @return A list of completed parts
 */
public static List<CompletedPart>
multipartUploadCopyForDirectoryBucket(S3Client s3Client, String sourceBucket,
    String sourceKey, String destinationBucket, String destinationKey,
    String uploadId) {
    // Get the object size to track the end of the copy operation
    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(sourceBucket)
        .key(sourceKey)
        .build();
    HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
    long objectSize = headObjectResponse.contentLength();
```

```
logger.info("Source Object size: {}", objectSize);

// Copy the object using 20 MB parts
long partSize = 20 * 1024 * 1024; // 20 MB
long bytePosition = 0;
int partNum = 1;
List<CompletedPart> uploadedParts = new ArrayList<>();

while (bytePosition < objectSize) {
    long lastByte = Math.min(bytePosition + partSize - 1, objectSize -
1);

    logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
partNum, bytePosition, lastByte);

    try {
        UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(sourceKey)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .uploadId(uploadId)
            .copySourceRange("bytes=" + bytePosition + "-" +
lastByte)

            .partNumber(partNum)
            .build();

        UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNum)
            .eTag(uploadPartCopyResponse.copyPartResult().eTag())
            .build();
        uploadedParts.add(part);

        bytePosition += partSize;
        partNum++;
    } catch (S3Exception e) {
        logger.error("Failed to copy part number {}: {} - Error code:
{}", partNum,
            e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode());
        throw e;
    }
}
```

```
    }  
  
    return uploadedParts;  
}
```

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

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

## AWS SDK を使用する S3 ディレクトリバケットのシナリオ

次のコード例は、AWS SDK を使用して S3 ディレクトリバケットで一般的なシナリオを実装する方法を示しています。これらのシナリオは、S3 ディレクトリバケット内で複数の関数を呼び出すか、その他の AWS のサービス と組み合わせて、特定のタスクを実行する方法を示しています。各シナリオには、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

シナリオは、サービスアクションをコンテキストで理解するのに役立つ中級レベルの経験を対象としています。

### 例

- [AWS SDK を使用して、オブジェクトを取得するための Amazon S3 ディレクトリバケットの署名付き URL を作成する](#)

## AWS SDK を使用して、オブジェクトを取得するための Amazon S3 ディレクトリバケットの署名付き URL を作成する

次のコード例は、S3 ディレクトリバケットの署名付き URL を作成し、オブジェクトを取得する方法を示しています。

## Java

## SDK for Java 2.x

 Note

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

S3 ディレクトリバケット内のオブジェクトにアクセスするための署名付き GET URL を生成します。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;

import java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Generates a presigned URL for accessing an object in the specified S3
```



```
* directory bucket.
*
* @param s3Presigner The S3 presigner client used to generate the presigned
URL
* @param bucketName The name of the directory bucket
* @param objectKey The key (name) of the object to access
* @return A presigned URL for accessing the specified object
*/
public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
s3Presigner, String bucketName,
    String objectKey) {
    logger.info("Generating presigned URL for object: {} in bucket: {}",
objectKey, bucketName);

    try {
        // Create a GetObjectRequest
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Create a GetObjectPresignRequest
        GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
valid for 10 minutes
            .getObjectRequest(getObjectRequest)
            .build();

        // Generate the presigned URL
        PresignedGetObjectRequest presignedGetObjectRequest =
s3Presigner.presignGetObject(getObjectPresignRequest);

        // Get the presigned URL
        String presignedURL = presignedGetObjectRequest.url().toString();
        logger.info("Presigned URL: {}", presignedURL);
        return presignedURL;

    } catch (S3Exception e) {
        logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

```
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[GetObject](#)」を参照してください。

AWS SDK デベロッパーガイドとコード例の詳細なリストについては、[AWS SDK を使用した Amazon S3 での開発](#) を参照してください。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

# リクエストの認証 (AWS 署名バージョン 4)

## トピック

- [認証方法](#)
- [リクエストへの署名の概要](#)
- [リクエストの認証: Authorization ヘッダーの使用 \(AWS 署名バージョン 4\)](#)
- [リクエストの認証: クエリパラメータの使用 \(AWS 署名バージョン 4\)](#)
- [例: AWS 署名バージョン 4 の署名計算](#)
- [リクエストの認証: POST \(AWS 署名バージョン 4\) を使用したブラウザベースのアップロード](#)
- [Amazon S3 署名バージョン 4 の認証固有のポリシーキー](#)

Amazon S3 とのすべてのやり取りは認証されるか匿名で行われます。このセクションでは、AWS 署名バージョン 4 アルゴリズムを使用したリクエスト認証について説明します。

### Note

AWS SDK (「[サンプルコードとライブラリ](#)」を参照) を使用してリクエストを送信する場合、SDK クライアントがユーザーが指定したアクセスキーを使用してリクエストを認証するため、このセクションを読む必要はありません。正当な理由がない限り、常に AWS SDK を使用してください。両方の署名バージョンをサポートするリージョンでは、特定の署名バージョンを使用するように AWS SDK にリクエストできます。詳細については、「[AWS SDK を使用した、認証されたリクエストの送信](#)」を参照してください。カスタムクライアントで AWS 署名バージョン 4 アルゴリズムを実装する場合にのみ、このセクションを読む必要があります。

AWS 署名バージョン 4 による認証では、リクエストの署名方法に応じて、以下の一部またはすべてが提供されます。

- リクエストの ID の確認 – 認証されたリクエストには、アクセスキー (アクセスキー ID、シークレットアクセスキー) を使用して作成した署名が必要です。アクセスキーの取得に関する詳細については、「AWS 全般のリファレンス」の「[セキュリティ認証情報の理解と取得](#)」を参照してください。一時的なセキュリティ認証情報を使用している場合、署名の計算にはセキュリティトロー

クンも必要です。詳細については、「IAM ユーザーガイド」の「[Requesting Temporary Security Credentials](#)」を参照してください。

- 転送中のデータ保護 – 転送中にリクエストが改ざんされないようにするには、リクエスト要素の一部を使用してリクエスト署名を計算します。リクエストを受信すると、Amazon S3 は同じリクエスト要素を使用して署名を計算します。Amazon S3 が受信したリクエストコンポーネントが、署名の計算に使用されたコンポーネントと一致しない場合、Amazon S3 はリクエストを拒否します。
- リクエストの署名済み部分の再利用に対する保護 – リクエストの署名済み部分 (AWS 署名を使用) は、リクエストのタイムスタンプから 15 分間有効です。署名済みリクエストにアクセス可能な権限のない関係者は、15 分間の有効期間内であれば、リクエストの有効性に影響を与えることなく、リクエストの署名されていない部分を変更できます。このため、リクエストヘッダーと本文に署名し、Amazon S3 に HTTPS リクエストを行い、AWS ポリシーで `s3:x-amz-content-sha256` 条件キー ([Amazon S3 署名バージョン 4 の認証固有のポリシーキー](#) を参照) を使用して Amazon S3 リクエスト本文への署名をユーザーに要求することで、保護を最大化することをお勧めします。

#### Note

Amazon S3 では、署名バージョン 4 がサポートされています。署名バージョン 4 は、すべての AWS リージョンで AWS のサービスに対するインバウンドの API リクエストを認証するためのプロトコルです。現時点では、2014 年 1 月 30 日以前に作成された AWS リージョンは、以前のプロトコルである署名バージョン 2 のサポートを継続します。2014 年 1 月 30 日以降の新しいリージョンでは、署名バージョン 4 のみがサポートされます。そのため、これらのリージョンに対するすべてのリクエストは署名バージョン 4 で実行される必要があります。AWS 署名バージョン 2 の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[REST リクエストの署名と認証](#)」を参照してください。

## 認証方法

認証情報は、次のいずれかの方法で表現できます。

- HTTP Authorization ヘッダー – HTTP Authorization ヘッダーの使用は、Amazon S3 リクエストを認証する最も一般的な方法です。すべての Amazon S3 REST オペレーション (POST リクエストを使用したブラウザベースのアップロードを除く) には、このヘッダーが必要で

す。Authorization ヘッダー値、および署名の計算方法や関連するオプションの詳細については、「[リクエストの認証: Authorization ヘッダーの使用 \(AWS 署名バージョン 4\)](#)」を参照してください。

- クエリ文字列パラメータ – クエリ文字列を使用すると、リクエスト全体を URL で表現できます。この場合、クエリパラメータを使用して、認証情報を含むリクエスト情報を提供します。リクエスト署名は URL の一部であるため、この種類の URL は署名付き URL と呼ばれることがよくあります。署名付き URL を使用して、クリック可能なリンクを HTML に埋め込むことができます。このリンクは最大 7 日間有効です。詳細については、「[リクエストの認証: クエリパラメータの使用 \(AWS 署名バージョン 4\)](#)」を参照してください。

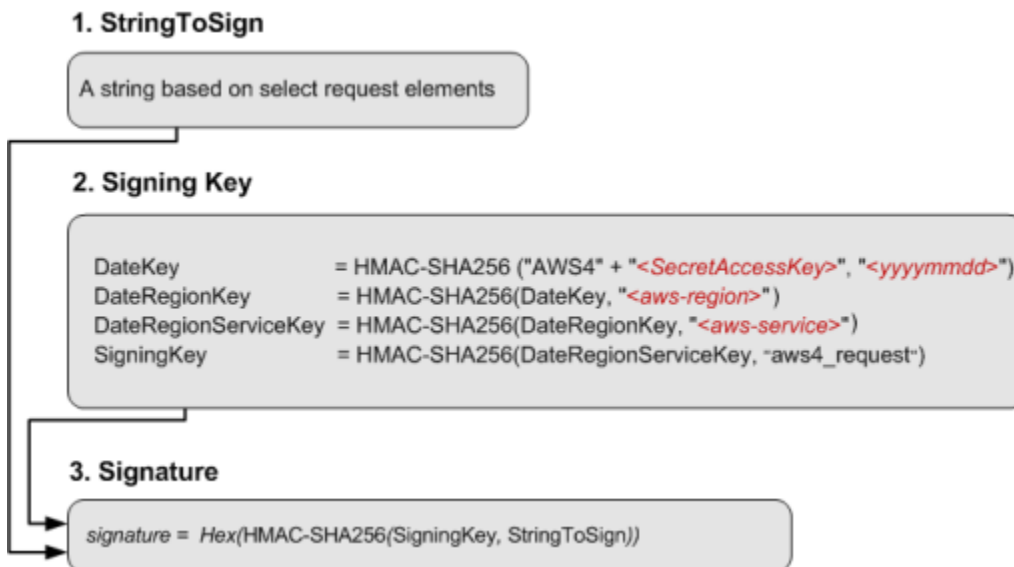
Amazon S3 は、HTTP POST リクエストを使用するブラウザベースのアップロードもサポートしています。HTTP POST リクエストを使用すると、ブラウザから直接 Amazon S3 にコンテンツをアップロードできます。POST リクエストの認証については、「[POST \(AWS 署名バージョン 4\) を使用したブラウザベースのアップロード](#)」を参照してください。

## リクエストへの署名の概要

リクエストで送信する認証情報には、署名が含まれている必要があります。署名を計算するには、選択されているリクエストの要素を連結し、文字列を作成します。これは、署名する文字列と呼ばれます。次に、署名キーを使用して、署名する文字列の Hash-based Message Authentication Code (HMAC) を計算します。

AWS の署名バージョン 4 では、リクエストの署名にシークレットアクセスキーは使用しません。代わりに、まずシークレットアクセスキーを使用して署名キーを生成します。生成される署名キーは、日付、サービス、およびリージョンによって異なります。さまざまなプログラミング言語で署名キーを取得する方法の詳細については、「[署名バージョン 4 の署名キーを取得する方法の例](#)」を参照してください。

次の図は、署名を計算する一般的なプロセスを示しています。



署名する文字列はリクエストのタイプによって異なります。たとえば、認証に HTTP Authorization ヘッダーまたはクエリパラメータを使用する場合、リクエスト要素のさまざまな組み合わせを使用して、署名する文字列を作成します。HTTP POST リクエストでは、リクエスト内の POST ポリシーは、署名する文字列です。署名する文字列の計算の詳細については、このセクションの最後にあるリンクを参照してください。

署名キーについては、図に一連の計算が示されており、各手順の結果は次の手順に入力されます。最後の手順は署名キーです。

認証されたリクエストを受信すると、Amazon S3 サーバーはリクエストに含まれる認証情報を使用して署名を再作成します。署名が一致した場合、Amazon S3 はリクエストを処理します。一致しない場合、リクエストは拒否されます。

リクエストの認証に関する詳細については、以下のトピックを参照してください。

- [リクエストの認証: Authorization ヘッダーの使用 \(AWS 署名バージョン 4\)](#)
- [リクエストの認証: クエリパラメータの使用 \(AWS 署名バージョン 4\)](#)
- [POST \(AWS 署名バージョン 4\) を使用したブラウザベースのアップロード](#)

## リクエストの認証: Authorization ヘッダーの使用 (AWS 署名バージョン 4)

### トピック

- [概要](#)

- [Authorization ヘッダーの署名計算: 単一チャンクでのペイロードの転送 \(AWS 署名バージョン 4\)](#)
- [Authorization ヘッダーの署名計算: 複数チャンクでのペイロードの転送 \(チャンクアップロード\) \(AWS 署名バージョン 4\)](#)
- [末尾ヘッダーの署名計算 \(チャンクアップロード\) \(AWS 署名バージョン 4\)](#)

## 概要

HTTP Authorization ヘッダーの使用は、認証情報を提供する最も一般的な方法です。[POST リクエスト](#)およびクエリパラメータを使用して署名されたリクエストを除き、すべての Amazon S3 オペレーションは Authorization リクエストヘッダーを使用して認証情報を提供します。

以下は、Authorization ヘッダー値の例です。この例では、読みやすいように改行が追加されています。

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

次の表では、前述の例にある Authorization ヘッダー値のさまざまなコンポーネントについて説明しています。

| コンポーネント          | 説明                                                                                                                              |
|------------------|---------------------------------------------------------------------------------------------------------------------------------|
| AWS4-HMAC-SHA256 | 署名の計算に使用されたアルゴリズム。認証に AWS 署名バージョン 4 を使用する場合は、この値を指定する必要があります。<br><br>文字列は、AWS 署名バージョン 4 (AWS4) と署名アルゴリズム (HMAC-SHA256 ) を指定します。 |
| Credential       | アクセスキー ID と、署名の計算に使用された日付、リージョン、およびサービスを含むスコープ情報。                                                                               |

| コンポーネント       | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | <p data-bbox="602 212 1125 243">この文字列の形式は次のとおりです。</p> <div data-bbox="602 331 1507 436"><pre data-bbox="618 338 1490 411">&lt;your-access-key-id&gt; /&lt;date&gt;/&lt;aws-region&gt; /&lt;aws-service&gt; /aws4_request</pre></div> <p data-bbox="602 495 967 527">説明は以下のとおりです。</p> <ul data-bbox="602 615 1425 814" style="list-style-type: none"><li data-bbox="602 615 1425 678">• <b>&lt;date&gt;</b> 値は YYYYMMDD 形式を使用して指定します。</li><li data-bbox="602 699 1425 814">• Amazon S3 にリクエストを送信する場合、<b>&lt;aws-service&gt;</b> の値は s3 です。</li></ul> |
| SignedHeaders | <p data-bbox="602 955 1500 1134">Signature の計算に使用したリクエストヘッダーをセミコロンで区切ったリスト。リストにはヘッダー名のみが含まれており、ヘッダー名は小文字である必要があります。例えば、次のようになります。</p> <div data-bbox="602 1192 954 1224"><pre data-bbox="618 1199 954 1224">host;range;x-amz-date</pre></div>                                                                                                                                                                                                                                                                                                       |
| Signature     | <p data-bbox="602 1314 1487 1398">64 個の小文字の 16 進数文字で表現される 256 ビットの署名。例えば、次のようになります。</p> <div data-bbox="602 1457 1338 1530"><pre data-bbox="618 1463 1338 1530">fe5f80f77d5fa3beca038a248ff027d0445342fe2855d dc963176630326f1024</pre></div> <p data-bbox="602 1589 1490 1673">署名計算は、ペイロードの転送に選択したオプションによって異なります。</p>                                                                                                                                                                                                                              |

署名計算は、リクエストペイロードの転送に選択した方法によって異なります。S3 は、以下のオプションをサポートしています。



- 単一のチャンクでペイロードを転送する – この場合、次の署名計算オプションがあります。
- 署名付きペイロードオプション – オプションでペイロード全体のチェックサムを計算し、署名計算に含めることができます。これによりセキュリティが向上しますが、ペイロードを 2 回読み取るか、メモリにバッファする必要があります。

例えば、ファイルをアップロードするには、まずファイルを読み取って署名計算用のペイロードハッシュを計算し、リクエストの作成時に再びファイルを読み込む必要があります。ペイロードが小さい場合は、このアプローチが適しています。ただし、大きなファイルの場合、ファイルを 2 回読み取ると非効率になる可能性があるため、代わりにデータをチャンクでアップロードすることをお勧めします。

セキュリティを強化するために、ペイロードのチェックサムを含めることをお勧めします。

- 署名なしペイロードオプション – 署名計算にペイロードチェックサムを含めない。

署名を計算し、Authorization ヘッダー値を構築するステップバイステップの手順については、「[Authorization ヘッダーの署名計算: 単一チャンクでのペイロードの転送 \(AWS 署名バージョン 4\)](#)」を参照してください。

- 複数のチャンクでペイロードを転送する (チャンクアップロード) – この場合、ペイロードをチャンクで転送します。ペイロードのサイズに関係なく、ペイロードをチャンクで転送できます。

ペイロードをチャンクに分割できます。固定または可変サイズのチャンクを使用できます。データをチャンクでアップロードすることで、ペイロード全体を読み取って署名を計算する必要がなくなります。代わりに、最初のチャンクで、リクエストヘッダーのみを使用するシード署名を計算します。2 番目のチャンクには最初のチャンクの署名が含まれ、後続の各チャンクにはそれより前のチャンクの署名が含まれます。アップロードの最後に、ペイロードの最後のチャンクの署名を含む、データ長が 0 バイトの最後のチャンクを送信します。詳細については、「[Authorization ヘッダーの署名計算: 複数チャンクでのペイロードの転送 \(チャンクアップロード\) \(AWS 署名バージョン 4\)](#)」を参照してください。

リクエストに署名するときは、AWS Signature Version 4 または AWS Signature Version 4A のいずれかを使用できます。これら 2 つの大きな違いを決定づけるのは、署名の計算方法です。AWS Signature Version 4A では、署名にはリージョン固有の情報は含まれず、署名の計算には AWS4-ECDSA-P256-SHA256 アルゴリズムが使用されます。

これらのオプションに加えて、リクエストにトレーラーを含めることもできます。リクエストにトレーラーを含めるには、`x-amz-content-sha256` を適切な値に設定して、ヘッダーで指定する必要があります。末尾ヘッダーを使用している場合は、ヘッダーに `x-amz-trailer` を含め、末尾

ヘッダー名をカンマ区切りリストの文字列として指定する必要があります。すべての末尾ヘッダーは、最後のチャンクの後に書き込まれます。データを複数のチャンクに分けてアップロードする場合、末尾ヘッダーを送信する前に、データ長が 0 バイトの最後のチャンクを送信する必要があります。

リクエストを送信するときは、次のいずれかの値を持つ `x-amz-content-sha256` ヘッダーを追加して、署名計算で前述のオプションのどれを選択したかを Amazon S3 に伝える必要があります。

| ヘッダー値                                            | 説明                                                                                                                                         |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| 実際のペイロードチェックサム値                                  | この値はオブジェクトの実際のチェックサムであり、データを単一のチャンクでアップロードする場合にのみ使用できます。                                                                                   |
| UNSIGNED-PAYLOAD                                 | オブジェクトを単一の署名なしチャンクとしてアップロードする場合に使用します。                                                                                                     |
| STREAMING-UNSIGNED-PAYLOAD-TRAILER               | 署名なしペイロードを複数のチャンクに分けて送信する場合に使用します。この場合、チャンクのアップロード後に末尾ヘッダーも付加されます。                                                                         |
| STREAMING-AWS4-HMAC-SHA256-PAYLOAD               | 複数のチャンクに分けてペイロードを送信する場合に使用します。チャンクは <code>AWS4-HMAC-SHA256</code> を使用して署名されます。これにより、SigV4 署名が生成されます。                                       |
| STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER       | 複数のチャンクに分けてペイロードを送信する場合に使用します。チャンクは <code>AWS4-HMAC-SHA256</code> を使用して署名されます。これにより、SigV4 署名が生成されます。さらに、チャンクのダイジェストが末尾ヘッダーとして含まれます。        |
| STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD         | 複数のチャンクに分けてペイロードを送信する場合に使用します。チャンクは <code>AWS4-ECDSA-P256-SHA256</code> を使用して署名されます。これにより、SigV4A 署名が生成されます。                                |
| STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD-TRAILER | 複数のチャンクに分けてペイロードを送信する場合に使用します。チャンクは <code>AWS4-ECDSA-P256-SHA256</code> を使用して署名されます。これにより、SigV4A 署名が生成されます。さらに、チャンクのダイジェストが末尾ヘッダーとして含まれます。 |

リクエストを受信すると、Amazon S3 は Authorization ヘッダーと date ヘッダーの情報を使用して署名する文字列を再作成します。次に、署名が一致することを認証サービスで検証します。リクエスト日は、HTTP Date または x-amz-date ヘッダーを使用して指定できます。両方のヘッダーが存在する場合、x-amz-date が優先されます。

署名が一致した場合、Amazon S3 はリクエストを処理します。一致しない場合、リクエストは失敗します。

詳細については、以下の各トピックを参照してください。

[Authorization ヘッダーの署名計算: 単一チャンクでのペイロードの転送 \(AWS 署名バージョン 4\)](#)

[Authorization ヘッダーの署名計算: 複数チャンクでのペイロードの転送 \(チャンクアップロード\) \(AWS 署名バージョン 4\)](#)

[末尾ヘッダーの署名計算 \(チャンクアップロード\) \(AWS 署名バージョン 4\)](#)

## Authorization ヘッダーの署名計算: 単一チャンクでのペイロードの転送 (AWS 署名バージョン 4)

Authorization ヘッダーを使用してリクエストを認証する場合、ヘッダー値には特に署名が含まれます。署名の計算は、ペイロードの転送 ([概要](#)) の選択によって異なります。このセクションでは、ペイロードを単一のチャンクで転送することを選択した場合の署名計算について説明します。サンプルセクション ([例: 署名計算](#) を参照) には、コードを検証するためのテストスイートとして使用できる署名計算と結果として生成される Authorization ヘッダーが表示されます。

### Important

ペイロードを単一のチャンクで転送する場合、署名計算にペイロードハッシュを含めることをオプションで選択できます。これを含める場合、署名付きペイロードと呼ばれます (含めない場合、ペイロードは署名なしと見なされます)。次のセクションで説明する署名手順は両方に適用されますが、次の違いに注意してください。

- 署名付きペイロードオプション – 正規リクエストを作成するときにペイロードハッシュを含めます (署名計算セクションで説明されているように、これは StringToSign の一部になります)。また、リクエストを S3 に送信するときに、x-amz-content-sha256 ヘッダー値と同じ値を指定します。

- 署名なしペイロード – 正規リクエストを作成するときにはリテラル文字列 UNSIGNED-PAYLOAD を含め、Amazon S3 にリクエストを送信するときにも、その文字列と同じ値を x-amz-content-sha256 ヘッダー値として設定します。

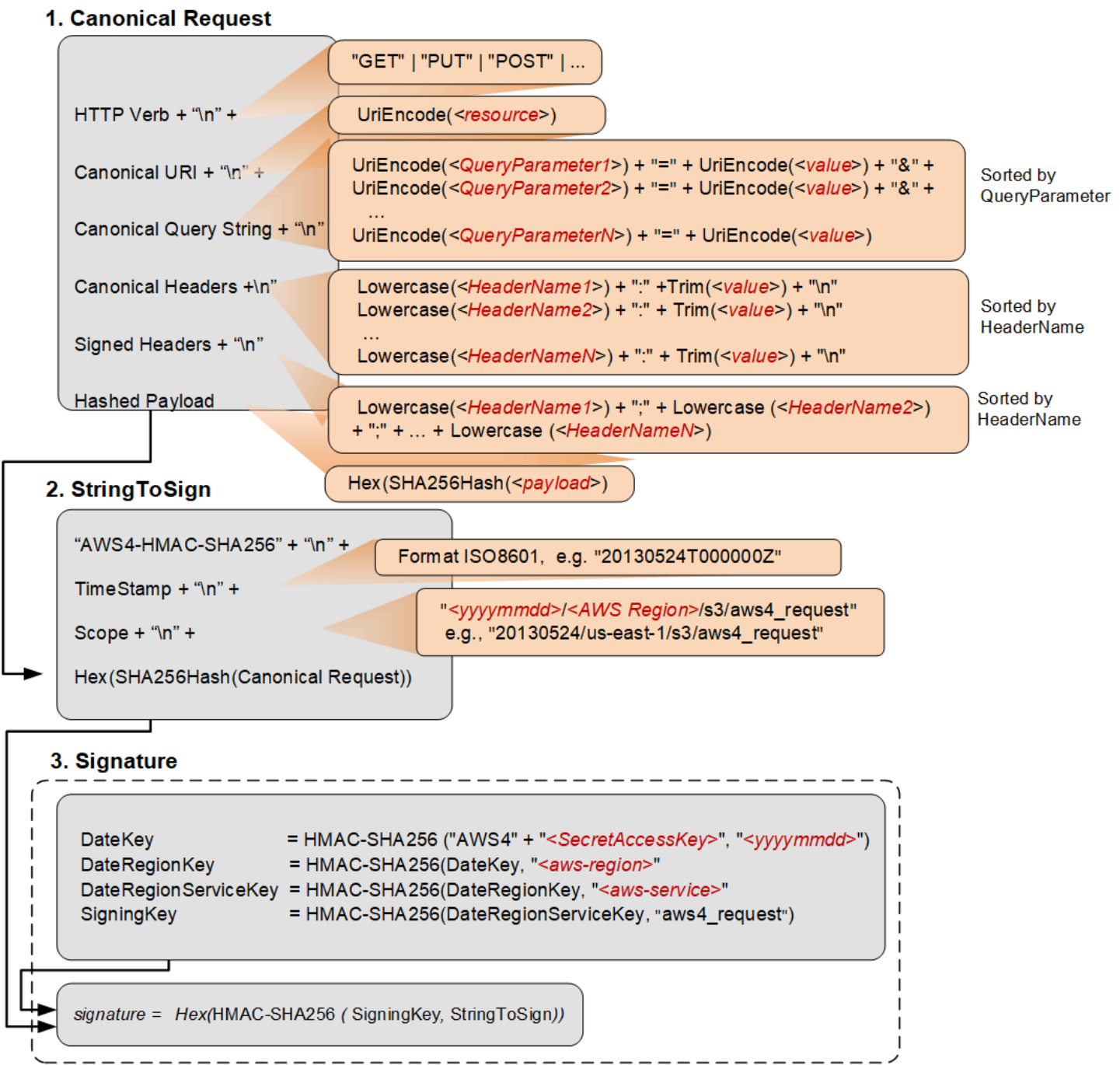
Amazon S3 にリクエストを送信すると、x-amz-content-sha256 ヘッダー値はペイロードが署名されているかどうかを Amazon S3 に通知します。その後、Amazon S3 は検証のためにそれに応じて署名を作成できます。

どちらの場合も、x-amz-content-sha256 ヘッダーの値は既に HashedPayload の一部であるため、x-amz-content-sha256 ヘッダーを正規ヘッダーとして含める必要はありません。

## 署名の計算

署名を計算するには、まず署名する文字列が必要です。次に、署名キーを使用して署名する文字列の HMAC-SHA256 ハッシュを計算します。次の図は、署名用に作成する文字列のさまざまなコンポーネントを含むプロセスを示しています

Amazon S3 は、認証されたリクエストを受信すると、署名を計算し、その署名をリクエスト内で提供された署名と比較します。そのため、署名は Amazon S3 で使用されているものと同じ方法で計算する必要があります。署名のために、リクエストを承認済み形式にするプロセスは正規化と呼ばれます。



次の表では、図に示されている関数について説明します。これらの関数のコードを実装する必要があります。

| 関数          | 説明             |
|-------------|----------------|
| Lowercase() | 文字列を小文字に変換します。 |

| 関数            | 説明                                                   |
|---------------|------------------------------------------------------|
| Hex()         | 16 進数の小文字エンコード。                                      |
| SHA256Hash()  | セキュアハッシュアルゴリズム (SHA) 暗号化ハッシュ関数。                      |
| HMAC-SHA256() | 指定した署名キーで SHA256 アルゴリズムを使用して HMAC を計算します。これが最後の署名です。 |
| Trim()        | 先頭または末尾の空白をすべて削除します。                                 |

| 関数          | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UriEncode() | <p>すべてのバイトが URI でエンコードされます。UriEncode() は、以下のルールを適用する必要があります。</p> <ul style="list-style-type: none"><li>• 非予約文字 (「A」 - 「Z」、 「a」 - 「z」、 「0」 - 「9」、 「-」、「.」、「_」 および 「~」 ) を除くすべてのバイトが URI でエンコードされます。</li><li>• スペース文字は予約文字であるため、「+」ではなく「%20」としてエンコードする必要があります。</li><li>• URI で エンコードされた各バイトは、「%」とそのバイトの 2 桁の 16 進値で構成されています。</li><li>• 16 進値の文字は大文字である必要があります (例: 「%1A」 )。</li><li>• オブジェクトキー名以外では、フォワードスラッシュ文字「/」がエンコードされます。例えば、オブジェクトキー名が photos/Jan/sample.jpg の場合、キー名のフォワードスラッシュはエンコードされません。</li></ul> <div data-bbox="630 1108 662 1150"></div> <div data-bbox="678 1108 1474 1392"><p><b>Important</b></p><p>開発プラットフォームが提供する標準の UriEncode 関数は、実装の違いや基礎となる RFC のあいまいさにより動作しない場合があります。エンコードが確実に機能するように、独自のカスタム UriEncode 関数を作成することをお勧めします。</p></div> <p>Java の UriEncode() 関数の例を次に示します。</p> <pre data-bbox="613 1591 1507 1820">public static String UriEncode(CharSequence input,     boolean encodeSlash) {     StringBuilder result = new StringBui lder();     for (int i = 0; i &lt; input.length(); i++) {         char ch = input.charAt(i);</pre> |

| 関数 | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    | <pre>        if ((ch &gt;= 'A' &amp;&amp; ch &lt;= 'Z')    (ch &gt;= 'a' &amp;&amp; ch &lt;= 'z')    (ch &gt;= '0' &amp;&amp; ch &lt;= '9')    ch == '_'    ch == '-'    ch == '~'    ch == '.') {             result.append(ch);         } else if (ch == '/') {             result.append(encodeSlash ? "%2F" : ch);         } else {             result.append(toHexUTF8(ch));         }     }     return result.toString(); }</pre> |

## タスク 1: 正規リクエストを作成する

このセクションでは、正規リクエストの作成の概要を説明します。

以下は、Amazon S3 が署名の計算に使用する正規リクエスト形式です。署名を一致させるには、次の形式で正規リクエストを作成する必要があります。

```
<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>
```

説明は以下のとおりです。

- **HTTPMethod** は、GET、PUT、HEAD、DELETE などの HTTP メソッドの 1 つです。
- **CanonicalURI** とは、URI の絶対パスコンポーネントの URI エンコード版です。ドメイン名の後の「/」から始まり、文字列の末尾まで、またはクエリ文字列パラメータがある場合は疑問符文字（「?」）までの部分を指します。次の例の URI、/examplebucket/myphoto.jpg は絶対パスであるため、ユーザーは絶対パス内の「/」をエンコードしないでください。



```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

 Note

Amazon S3 へのリクエストにおいて、URI パスを正規化することはありません。例えば、「my-object//example//photo.user」という名前のオブジェクトを持つバケットがあるとします。パスを正規化すると、リクエストのオブジェクト名が「my-object/example/photo.user」に変更されます。これは、そのオブジェクトの正しくないパスです。

- **CanonicalQueryString** は、URI エンコードされたクエリ文字列パラメータを指定します。名前と値を個別に URI エンコードします。また、正規クエリ文字列内のパラメータをキー名のアルファベット順にソートする必要があります。ソートはエンコード後に行われます。次のサンプル URI のクエリ文字列は次のとおりです。prefix=somePrefix&marker=someMarker&max-keys=20

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

正規クエリ文字列は次のとおりです (この例では、読みやすいように改行が追加されています)。

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&" +  
UriEncode("max-keys")+"="+UriEncode("20") + "&" +  
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

リクエストがサブリソースをターゲットにしている場合、対応するクエリパラメータ値は空の文字列("") になります。例えば、次の URI は examplebucket バケット上の ACL サブリソースを特定してします。

```
http://s3.amazonaws.com/examplebucket?acl
```

この場合の CanonicalQueryString は次のとおりです。

```
UriEncode("acl") + "=" + ""
```

URI に「?」が含まれていない場合は、リクエストにクエリ文字列が存在しないため、正規クエリ文字列を空の文字列 ("" ) に設定します。「\n」は引き続き含める必要があります。

- **CanonicalHeaders** は、リクエストヘッダーとその値のリストです。個々のヘッダー名と値のペアは改行文字 (「\n」) で区切られます。ヘッダー名は小文字である必要があります。次の例に示すように、文字列を作成するには、ヘッダー名をアルファベット順にソートする必要があります。

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"  
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"  
...  
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

この例で使用されている関数 Lowercase() と Trim() は、前のセクションで説明されています。

**CanonicalHeaders** リストには以下が含まれている必要があります。

- HTTP host ヘッダー。
- Content-MD5 ヘッダーがリクエスト内に存在する場合は、それを **CanonicalHeaders** リストに追加する必要があります。
- リクエストに含める予定の x-amz-\* ヘッダーも追加する必要があります。例えば、一時的なセキュリティ認証情報を使用している場合は、リクエストに x-amz-security-token を含める必要があります。このヘッダーを **CanonicalHeaders** リストに追加する必要があります。

#### Note

- x-amz-content-sha256 ヘッダーは、すべての AWS 署名バージョン 4 リクエストに必要です。これにより、リクエストペイロードのハッシュが指定されます。ただし、S3 はリクエストで送信されるペイロードのハッシュを計算するときに自動的にその値を使用するため、x-amz-content-sha256 を正規ヘッダーとして含める必要はありません。
- ペイロードがない場合は、空の文字列のハッシュを指定する必要があります。
- S3 でリクエストのハッシュと照合しない場合は、代わりにリテラル文字列 "UNSIGNED-PAYLOAD" を使用できます。

以下は CanonicalHeaders 文字列の例です。ヘッダー名は小文字で、ソートされています。

## Example例 1

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

## Example例 2

```
host:s3.amazonaws.com
x-amz-content-sha256:UNSIGNED-PAYLOAD
x-amz-date:20130708T220855Z
```

## Example例 3

```
host:s3.amazonaws.com
x-amz-date:20130708T220855Z
```

### Note

認可署名を計算するには、ホストヘッダーと x-amz-content-sha256 を除くすべての x-amz-\* ヘッダーが必要です。ただし、データの改ざんを防ぐために、署名計算にすべてのヘッダーを含めることを検討する必要があります。

S3 は受信したリクエストペイロードハッシュを計算するときにその値を使用するため、x-amz-content-sha256 ヘッダーへの署名はオプションです。

- **SignedHeaders** は、アルファベット順にソートされ、セミコロンで区切られた小文字のリクエストヘッダー名のリストです。リスト内のリクエストヘッダーは、CanonicalHeaders 文字列に含めたヘッダーと同じです。前の例の場合、**SignedHeaders** の値は次のようになります。

## Example例 1 と 2

```
host;x-amz-content-sha256;x-amz-date
```

## Example例 3

```
host;x-amz-date
```

- **HashedPayload** は、リクエストペイロードの SHA256 ハッシュの 16 進値です。

```
Hex(SHA256Hash(<payload>))
```

リクエストにペイロードがない場合は、空の文字列のハッシュを次のように計算します。

```
Hex(SHA256Hash(""))
```

ハッシュは次の値を返します。

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

例えば、PUT リクエストを使用してオブジェクトをアップロードする場合、本文にオブジェクトデータを指定します。GET リクエストを使用してオブジェクトを取得する場合、空の文字列ハッシュを計算します。

## タスク 2: 署名文字列を作成する

このセクションでは、署名する文字列の作成の概要を説明します。ステップバイステップの手順については、「AWS 全般のリファレンス」の「[タスク 2: 署名する文字列を作成する](#)」を参照してください。

署名する文字列は、次の文字列を連結したものです。

```
"AWS4-HMAC-SHA256" + "\n" +  
timeStampISO8601Format + "\n" +  
<Scope> + "\n" +  
Hex(SHA256Hash(<CanonicalRequest>))
```

定数文字列 AWS4-HMAC-SHA256 は、使用しているハッシュアルゴリズム HMAC-SHA256 を指定します。timeStamp は、ISO 8601 形式の現在の UTC 時刻です (例: 20130524T000000Z)。

Scope は、生成された署名を特定の日付、AWS リージョン、およびサービスにバインドします。したがって、生成された署名は、特定のリージョンおよび特定のサービスでのみ機能します。署名は、指定された日付から 7 日間有効です。

```
date.Format(<YYYYMMDD>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

Amazon S3 の場合、サービス文字列は s3 です。**region** 文字列のリストについては、「AWS 全般のリファレンス」の「[リージョンとエンドポイント](#)」を参照してください。この表のリージョン列には、有効なリージョン文字列のリストが表示されます。

次のスコープは、生成される署名を us-east-1 リージョンと Amazon S3 に制限します。

```
20130606/us-east-1/s3/aws4_request
```

#### Note

Scope は、次のセクションで説明するように、署名キーの計算に使用するのと同じ日付を使用する必要があります。

### タスク 3: 署名を計算する

AWS 署名バージョン 4 では、リクエストの署名に AWS アクセスキーを使用するのではなく、特定のリージョンおよびサービスにスコープされた署名キーを最初に作成します。署名キーの詳細については、「[リクエストへの署名の概要](#)」を参照してください。

```
DateKey          = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<YYYYMMDD>")
DateRegionKey    = HMAC-SHA256(<DateKey>, "<aws-region>")
DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>, "<aws-service>")
SigningKey       = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")
```

#### Note

一部のユースケースでは、署名キーを最大 7 日間処理できます。詳細については、「[他ユーザーとのオブジェクトの共有](#)」を参照してください。

リージョン文字列のリストについては、「AWS 全般のリファレンス」の「[リージョンとエンドポイント](#)」を参照してください。

署名キーを使用すると、AWS 認証情報を 1 つの安全な場所に保持できます。例えば、Amazon S3 と通信するサーバーが複数ある場合は、署名キーをそれらのサーバーと共有します。各サーバーにシークレットアクセスキーのコピーを保持する必要はありません。署名キーは最大 7 日間有効です。したがって、署名キーを計算するたびに、署名キーをサーバーと共有する必要があります。詳細については、「[リクエストの認証 \(AWS 署名バージョン 4\)](#)」を参照してください。

最後の署名は、署名キーをキーとして使用し、署名する文字列に対して HMAC-SHA256 を適用して得られるハッシュ値です。

```
HMAC-SHA256(SigningKey, StringToSign)
```

署名を作成するステップバイステップの手順については、「AWS 全般のリファレンス」の「[タスク 3: 署名を作成する](#)」を参照してください。

例: 署名計算

このセクションの例を参考にして、コード内の署名計算を確認できます。例に示す計算では、次のデータを使用します。

- アクセスキーの例。

| パラメータ              | 値                                        |
|--------------------|------------------------------------------|
| AWSAccessKeyId     | AKIAIOSFODNN7EXAMPLE                     |
| AWSSecretAccessKey | wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY |

- リクエストのタイムスタンプ 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT)。
- バケット名 examplebucket。
- バケットは米国東部 (バージニア北部) リージョンにあると見なされます。認証情報 Scope と Signing Key 計算では、us-east-1 をリージョン指定子として使用します。リージョンの詳細については、「AWS 全般のリファレンス」の「[リージョンとエンドポイント](#)」を参照してください。
- パススタイルまたは仮想ホスト形式のリクエストのいずれかを使用できます。次の例は、仮想ホスト形式のリクエストに署名する方法を示しています。例えば、次のようになります。

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[バケットの仮想ホスティング](#)」を参照してください。

## 例: GET Object

次の例では、examplebucket からオブジェクト (test.txt) の最初の 10 バイトを取得します。API アクションの詳細については、「[GetObject](#)」を参照してください。

```
GET /test.txt HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
Range: bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date: 20130524T000000Z
```

この GET リクエストは本文コンテンツを提供しないため、x-amz-content-sha256 の値は空のリクエスト本文のハッシュまたはリテラル文字列 "UNSIGNED-PAYLOAD" のいずれかになります。次の手順は、空の文字列のハッシュを使用して署名を計算し、Authorization ヘッダーを構成する方法を示しています。

### 1. StringToSign

#### a. CanonicalRequest

```
GET
/test.txt

host:examplebucket.s3.amazonaws.com
range:bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;range;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

正規リクエスト文字列の最後の行は、空のリクエスト本文のハッシュです。リクエストにクエリパラメータが含まれていないため、3行目は空です。

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
7344ae5b7ee6c3e7e6b0fe0640412a37625d1fbffff95c48bbb2dc43964946972
```

## 2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

## 3. Signature

```
f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

## 4. Authorization ヘッダー

その結果得られる Authorization ヘッダーは以下のようになります。

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;range;x-amz-content-sha256;x-amz-
date,Signature=f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

### 例: PUT Object

この PUT リクエストの例では、examplebucket にオブジェクト (test\$file.text) を作成します。例では、次のように想定しています。

- x-amz-storage-class リクエストヘッダーを追加して、ストレージクラスとして REDUCED\_REDUNDANCY をリクエストしています。ストレージクラスの詳細については、



「Amazon Simple Storage Service ユーザーガイド」の「[ストレージクラス](#)」を参照してください。

- アップロードされたファイルのコンテンツは、"Welcome to Amazon S3." という文字列です。リクエストの x-amz-content-sha256 の値はこの文字列に基づいています。

API アクションの詳細については、「[PutObject](#)」を参照してください。

```
PUT test$file.text HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Fri, 24 May 2013 00:00:00 GMT
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
x-amz-content-sha256: 44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
```

*<Payload>*

次の手順は、署名の計算を示しています。

## 1. StringToSign

### a. CanonicalRequest

```
PUT
/test%24file.text

date:Fri, 24 May 2013 00:00:00 GMT
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
x-amz-date:20130524T000000Z
x-amz-storage-class:REDUCED_REDUNDANCY

date;host;x-amz-content-sha256;x-amz-date;x-amz-storage-class
44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
```

リクエストにクエリパラメータがないため、正規リクエストの 3 行目は空です。x-amz-content-sha256 正規ヘッダーは、ペイロードハッシュがリクエストの末尾に既に提供されているため、オプションで署名できます。最後の行は本文のハッシュで、HTTP リクエ

ストで S3 に送信される `x-amz-content-sha256` header 値と同じである必要があります。

## b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
9e0e90d9c76de8fa5b200d8c849cd5b8dc7a3be3951ddb7f6a76b4158342019d
```

## 2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

## 3. Signature

```
98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

## 4. Authorization ヘッダー

その結果得られる Authorization ヘッダーは以下のようになります。

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=date;host;x-amz-content-sha256;x-amz-date;x-amz-storage-
class,Signature=98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

## 例: GET Bucket Lifecycle

次の GET リクエストは、`examplebucket` のライフサイクル設定を取得します。API アクションの詳細については、「[GetBucketLifecycleConfiguration](#)」を参照してください。

```
GET ?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
```

```
x-amz-date: 20130524T000000Z
```

```
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

リクエストは本文コンテンツを提供しないため、x-amz-content-sha256 ヘッダー値は空のリクエスト本文のハッシュです。次の手順は、署名の計算を示しています。

## 1. StringToSign

### a. CanonicalRequest

```
GET
/
lifecycle=
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

正規リクエストの最後の行は、空のリクエスト本文のハッシュです。

### b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
9766c798316ff2757b517bc739a67f6213b4ab36dd5da2f94eaebf79c77395ca
```

## 2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

## 3. Signature

```
fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

#### 4. Authorization ヘッダー

その結果得られる Authorization ヘッダーは以下ようになります。

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

例: バケットの取得 (オブジェクトの一覧表示)

次の例では、examplebucket バケットからオブジェクトのリストを取得します。API アクションの詳細については、「[ListObjects](#)」を参照してください。

```
GET ?max-keys=2&prefix=J HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

リクエストは本文を提供しないため、x-amz-content-sha256 の値は空のリクエスト本文のハッシュです。次の手順は、署名の計算を示しています。

##### 1. StringToSign

###### a. CanonicalRequest

```
GET
/
max-keys=2&prefix=J
host:examplebucket.s3.amazonaws.com
x-amz-content-
sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
```

```
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

正規文字列の最後の行は、空のリクエスト本文のハッシュです。

## b. StringToSign

```
AWS4-HMAC-SHA256  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
df57d21db20da04d7fa30298dd4488ba3a2b47ca3a489c74750e0f1e7df1b9b7
```

## 2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +  
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

## 3. Signature

```
34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

## 4. Authorization ヘッダー

その結果得られる Authorization ヘッダーは以下のようになります。

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/  
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-  
date,Signature=34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

## Authorization ヘッダーの署名計算: 複数チャンクでのペイロードの転送 (チャンクアップロード) (AWS 署名バージョン 4)

**概要** で説明されているように、Authorization ヘッダーを使用してリクエストを認証する場合、ペイロードをチャンクでアップロードするオプションがあります。固定サイズまたは可変サイズのチャンクでデータを送信できます。このセクションでは、チャンクアップロードの署名計算プロセス、チャンクボディの作成方法、チャンクを最初にアップロードした後に後続のチャンクで署名を送信する遅延署名の仕組みについて説明します。サンプルセクション ([例: PUT Object](#) を参照) には、コードを検証するためのテストスイートとして使用できる署名計算と結果として生成される Authorization ヘッダーが表示されます。

### Note

一連のチャンクとしてデータを転送する場合は、次のいずれかを実行する必要があります。

- Content-Length HTTP ヘッダーを使用して、コンテンツの合計長 (オブジェクト長のバイト数と各チャンクのメタデータの合計) を明示的に指定します。これを行うには、リクエストを開始する前に、各チャンクで送信するメタデータを含むペイロードの合計長を事前に計算する必要があります。
- Transfer-Encoding HTTP ヘッダーを指定します。Transfer-Encoding ヘッダーを含め、identity 以外の値を指定する場合は、Content-Length ヘッダーを省略する必要があります。

すべてのリクエストについて、オブジェクトのサイズをバイト単位で指定する x-amz-decoded-content-length ヘッダーを含める必要があります。

各チャンクの署名の計算には、前のチャンクの署名が含まれます。最初に、ヘッダーのみを使用してシード署名を作成します。シード署名は、最初のチャンクの署名計算に使用します。後続の各チャンクについて、前のチャンクの署名を含むチャンク署名を作成します。このように、チャンク署名は互いに連結されます。つまり、チャンク  $n$  の署名は  $F(\text{チャンク } n, \text{署名}(\text{チャンク } n-1))$  という関数になります。連鎖により、チャンクを正しい順序で送信できます。

チャンクアップロードを実行するには、以下を実行します。

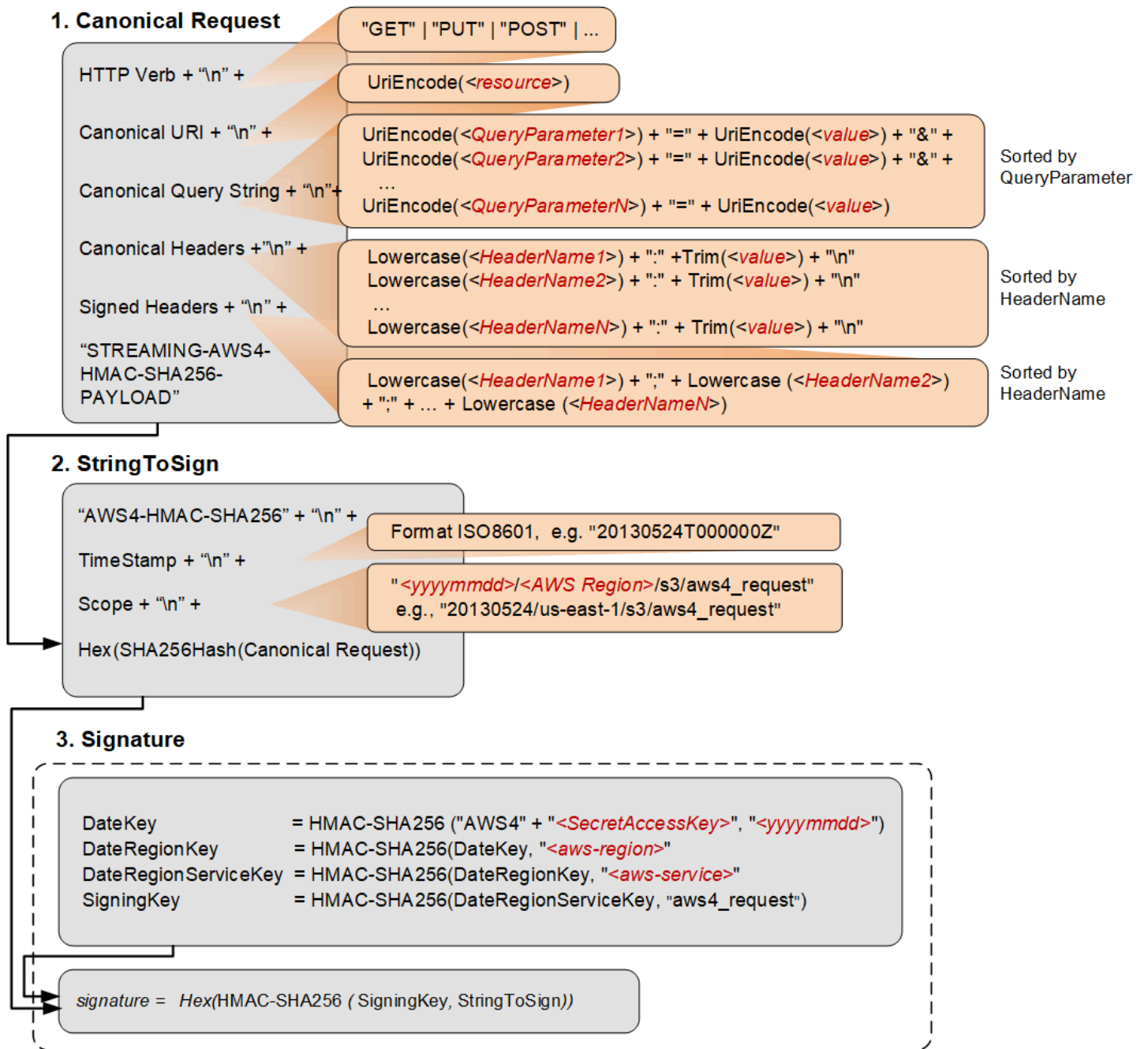
1. ペイロードのチャンクサイズを決定します。これは、コードを記述するときに必要なになります。

チャンクサイズは 8 KB 以上である必要があります。パフォーマンスを向上させるため、64 KB 以上のチャンクサイズをお勧めします。このチャンクサイズは、最後のチャンクを除くすべてのチャンクに適用されます。送信する最後のチャンクは 8 KB 未満でも構いません。ペイロードが小さく、1 つのチャンクに収まる場合、8 KB 未満にすることができます。

2. 最初のチャンクに含めるシード署名を作成します。詳細については、「[シード署名の計算](#)」を参照してください。
3. 最初のチャンクを作成してストリーミングします。詳細については、「[チャンクボディの定義](#)」を参照してください。
4. 後続のチャンクごとに、署名対象文字列に前の署名を含むチャンク署名を計算し、チャンクを作成して送信します。詳細については、「[チャンクボディの定義](#)」を参照してください。
5. 最後の追加チャンクを送信します。これは、構築内の他のチャンクと同じですが、データバイト数はゼロです。詳細については、「[チャンクボディの定義](#)」を参照してください。

## シード署名の計算

次の図は、シード署名を計算するプロセスを示しています。



次の表では、図に示されている関数について説明します。これらの関数のコードを実装する必要があります。

| 関数          | 説明              |
|-------------|-----------------|
| Lowercase() | 文字列を小文字に変換します。  |
| Hex()       | 16 進数の小文字エンコード。 |



| 関数            | 説明                                                   |
|---------------|------------------------------------------------------|
| SHA256Hash()  | セキュアハッシュアルゴリズム (SHA) 暗号化ハッシュ関数。                      |
| HMAC-SHA256() | 指定した署名キーで SHA256 アルゴリズムを使用して HMAC を計算します。これが最後の署名です。 |
| Trim()        | 先頭または末尾の空白をすべて削除します。                                 |

| 関数          | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UriEncode() | <p>すべてのバイトが URI でエンコードされます。UriEncode() は、以下のルールを適用する必要があります。</p> <ul style="list-style-type: none"><li>• 非予約文字 (「A」 - 「Z」、 「a」 - 「z」、 「0」 - 「9」、 「-」、「.」、「_」 および 「~」 ) を除くすべてのバイトが URI でエンコードされます。</li><li>• スペース文字は予約文字であるため、「+」ではなく「%20」としてエンコードする必要があります。</li><li>• URI で エンコードされた各バイトは、「%」とそのバイトの 2 桁の 16 進値で構成されています。</li><li>• 16 進値の文字は大文字である必要があります (例: 「%1A」 )。</li><li>• オブジェクトキー名以外では、フォワードスラッシュ文字「/」がエンコードされます。例えば、オブジェクトキー名が photos/Jan/sample.jpg の場合、キー名のフォワードスラッシュはエンコードされません。</li></ul> <div data-bbox="630 1108 662 1150"></div> <div data-bbox="678 1108 812 1150">Important</div> <p>開発プラットフォームが提供する標準の UriEncode 関数は、実装の違いや基礎となる RFC のあいまいさにより動作しない場合があります。エンコードが確実に機能するように、独自のカスタム UriEncode 関数を作成することをお勧めします。</p> <p>Java の UriEncode() 関数の例を次に示します。</p> <pre data-bbox="615 1602 1446 1829">public static String UriEncode(CharSequence input,     boolean encodeSlash) {     StringBuilder result = new StringBui lder();     for (int i = 0; i &lt; input.length(); i++) {         char ch = input.charAt(i);</pre> |

| 関数 | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    | <pre>         if ((ch &gt;= 'A' &amp;&amp; ch &lt;= 'Z')    (ch &gt;= 'a' &amp;&amp; ch &lt;= 'z')    (ch &gt;= '0' &amp;&amp; ch &lt;= '9')    ch == '_'    ch == '-'    ch == '~'    ch == '.') {             result.append(ch);         } else if (ch == '/') {             result.append(encodeSlash ? "%2F" : ch);         } else {             result.append(toHexUTF8(ch));         }     }     return result.toString(); } </pre> |

署名プロセスについては、「[Authorization ヘッダーの署名計算: 単一チャンクでのペイロードの転送 \(AWS 署名バージョン 4\)](#)」を参照してください。プロセスは同じですが、CanonicalRequest の作成は次のように異なります。

- 追加するリクエストヘッダーに加えて、次のヘッダーを含める必要があります。

| ヘッダー                 | 説明                                                                                                                   |
|----------------------|----------------------------------------------------------------------------------------------------------------------|
| x-amz-content-sha256 | このヘッダーは、すべての AWS 署名バージョン 4 リクエストに必要です。値を STREAMING-AWS4-HMAC-SHA256-PAYLOAD に設定して、署名がヘッダーのみを対象とし、ペイロードが含まれないことを示します。 |

| ヘッダー                         | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Content-Encoding             | <p>値を <code>aws-chunked</code> に設定します。</p> <p>Amazon S3 は、複数のコンテンツエンコーディング値をサポートしています。署名バージョン 4 ストリーミング API を使用する場合は、カスタムのコンテンツエンコードを指定できます。</p> <p>例えば、次のようになります。</p> <div data-bbox="500 632 1507 711"><pre>Content-Encoding : aws-chunked,gzip</pre></div> <p>Amazon S3 は、<code>content-encoding</code> ヘッダーに <code>aws-chunked</code> 値を含めずに、生成されたオブジェクトを保存します。<code>content-encoding</code> ヘッダーに渡す値が <code>aws-chunked</code> のみである場合、S3 は <code>content-encoding</code> ヘッダーを空と見なし、オブジェクトの取得時にこのヘッダーを返しません。</p> |
| x-amz-decoded-content-length | <p>メタデータをカウントせずに、チャンクするデータの長さをバイト単位で設定します。例えば、4 GB のファイルをアップロードする場合は、値を 4294967296 に設定します。これは、アップロードするオブジェクト (Amazon S3 に保存するデータ) の raw サイズです。</p>                                                                                                                                                                                                                                                                                                                                                                     |
| Content-Length               | <p>この値を、送信される HTTP 本文の実際のサイズに設定します。このサイズには、データ (<code>x-amz-decoded-content-length</code> に設定された値) に加えて、チャンクメタデータの長さも含まれます。各チャンクには、前のチャンクの署名などのメタデータがあります。チャンクの計算については、次のセクションで説明します。Transfer-Encoding ヘッダーを含め、<code>identity</code> 以外の値を指定する場合は、Content-Length ヘッダーを含めないでください。</p>                                                                                                                                                                                                                                   |

シード署名と共に最初のチャンクを送信します。次のセクションで説明するように、チャンクを作成する必要があります。

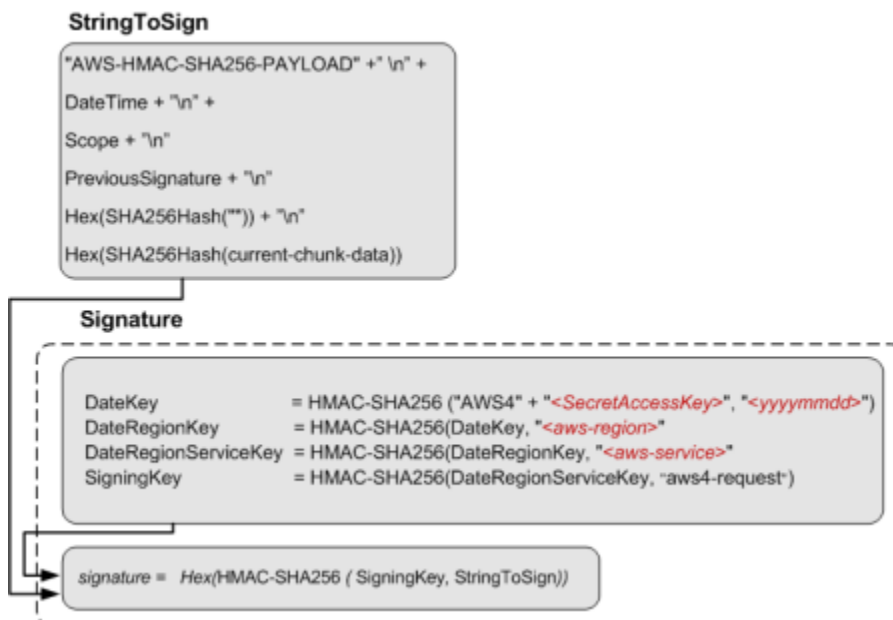
## チャンクボディの定義

すべてのチャンクにはメタデータが含まれています。各チャンクは、次の構造に従う必要があります。

```
string(IntHexBase(chunk-size)) + ";chunk-signature=" + signature + \r\n + chunk-data + \r\n
```

説明は以下のとおりです。

- `IntHexBase()` は、整数のチャンクサイズを 16 進数に変換するために記述する関数です。例えば、チャンクサイズが 65536 の場合、16 進数の文字列は「 10000」となります。
- *chunk-size* は、メタデータを含まないチャンクデータのサイズをバイト単位で表したものです。例えば、65 KB のオブジェクトを 64 KB のチャンクサイズでアップロードする場合、データは 3 つのチャンクに分けてアップロードされます。つまり 1 つ目は 64 KB、2 つ目は 1 KB、最後のチャンクは 0 バイトです。
- *signature* チャンクごとに、次の署名する文字列を使用して署名を計算します。最初のチャンクでは、シード署名を前の署名として使用します。



送信する最後のチャンクデータのサイズは 0 ですが、チャンクボディには、前のチャンクの署名を含むメタデータが含まれます。

## 例: PUT Object

このセクションの例を参考にして、コード内の署名計算を確認できます。例を確認する前に、次の点に注意してください。

- これらの例の署名計算では、次のサンプルのセキュリティ認証情報を使用します。

| パラメータ              | 値                                        |
|--------------------|------------------------------------------|
| AWSAccessKeyId     | AKIAIOSFODNN7EXAMPLE                     |
| AWSSecretAccessKey | wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY |

- すべての例では、リクエストのタイムスタンプとして 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT) を使用します。
- すべての例では、バケット名として examplebucket を使用します。
- バケットは米国東部 (バージニア北部) リージョンにあると見なされ、認証情報の Scope と Signing Key の計算ではリージョン指定子として us-east-1 が使用されます。詳細については、Amazon Web Services 全般のリファレンスの [リージョンとエンドポイント](#) を参照してください。
- パススタイルまたは仮想ホスト形式のリクエストのいずれかを使用できます。次の例では、仮想ホスト形式のリクエストを使用します。次に例を示します。

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[バケットの仮想ホスティング](#)」を参照してください。

次の例では、オブジェクトをアップロードする PUT リクエストを送信します。署名の計算では、次のことを前提としています。

- 65 KB のテキストファイルをアップロードしようとしています。ファイルの内容は文字「a」で構成される 1 文字の文字列です。
- チャンクサイズは 64 KB です。その結果、ペイロードは 3 つのチャンク (64 KB、1 KB、およびチャンクデータが 0 バイトの最後のチャンク) に分けてアップロードされます。
- 生成されるオブジェクトのキー名は chunkObject.txt です。
- x-amz-storage-class リクエストヘッダーを追加して、ストレージクラスとして REDUCED\_REDUNDANCY をリクエストしています。

API アクションの詳細については、「[PutObject](#)」を参照してください。一般的なリクエストの構文は次のとおりです。

```
PUT /examplebucket/chunkObject.txt HTTP/1.1
Host: s3.amazonaws.com
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
Authorization: SignatureToBeCalculated
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 66560
Content-Length: 66824
<Payload>
```

次の手順は、署名の計算を示しています。

1. シード署名 – 署名する文字列を作成する
  - a. CanonicalRequest

```
PUT
/examplebucket/chunkObject.txt

content-encoding:aws-chunked
content-length:66824
host:s3.amazonaws.com
x-amz-content-sha256:STREAMING-AWS4-HMAC-SHA256-PAYLOAD
x-amz-date:20130524T000000Z
x-amz-decoded-content-length:66560
x-amz-storage-class:REDUCED_REDUNDANCY
```

```
content-encoding;content-length;host;x-amz-content-sha256;x-amz-date;x-amz-
decoded-content-length;x-amz-storage-class
STREAMING-AWS4-HMAC-SHA256-PAYLOAD
```

リクエストにクエリパラメータがないため、正規リクエストの 3 行目は空です。最後の行は、ハッシュされたペイロードの値として提供される定数文字列で、x-amz-content-sha256 header の値と同じである必要があります。

## b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
cee3fed04b70f867d036f722359b0b1f2f0e5dc0efadbc082b76c4c60e316455
```

### Note

署名する文字列の各行については、シード署名の計算を説明する図を参照してください。

## 2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

## 3. シード署名

```
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

## 4. Authorization ヘッダー

その結果得られる Authorization ヘッダーは以下のようになります。

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=content-encoding;content-length;host;x-amz-
```



```
content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-
class,Signature=4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

## 5. チャンク 1: (65536 バイト、文字「a」の値 97)

### a. 署名するチャンク文字列。

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

#### Note

署名する文字列の各行の詳細については、署名する文字列のさまざまなコンポーネントを示す前述の図を参照してください (例えば、最後の 3 行は、previous-signature、hash("")、および hash(current-chunk-data) です)。

### b. チャンク署名。

```
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
```

### c. 送信されたチャンクデータ。

```
10000;chunk-
signature=ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
<65536-bytes>
```

## 6. チャンク 2: (1024 バイト、文字「a」の値 97)

### a. 署名するチャンク文字列。

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

```
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

- b. チャンク署名。

```
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
```

- c. 送信されたチャンクデータ。

```
400; chunk-  
signature=0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
<1024 bytes>
```

## 7. チャンク 3: (0 バイトのデータ)

- a. 署名するチャンク文字列。

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

- b. チャンク署名。

```
b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

- c. 送信されたチャンクデータ。

```
0; chunk-  
signature=b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

## 末尾ヘッダーの署名計算 (チャンクアップロード) (AWS 署名バージョン 4)

Authorization ヘッダーを使用してリクエストを認証する場合、ペイロードをチャンクでアップロードすることもできます。オブジェクトのデータをチャンクで送信する場合、末尾ヘッダーを含めることもできます。(詳しくは、[Authorization ヘッダーの署名計算: 複数チャンクでのペイロードの転送 \(チャンクアップロード\) \(AWS 署名バージョン 4\)](#) を参照してください)。このセクションでは、複数のチャンクアップロードの最後に末尾ヘッダーを含めるために必要な手順について説明します。

### Important

末尾ヘッダーを含める場合は、最初のヘッダーに以下を送信する必要があります。

- トレーラーが含まれることを示す適切な値に `x-amz-content-sha256` を設定する必要があります。`x-amz-content-sha256` に指定可能な値については、「[リクエストの認証: Authorization ヘッダーの使用 \(AWS 署名バージョン 4\)](#)」を参照してください。
- 末尾ヘッダーに含める内容を指定するように `x-amz-trailer` を設定する必要があります。

末尾ヘッダーは、チャンクがアップロードされた後にのみ送信されます。それ以前のチャンクは通常どおり送信され、前のセクションで説明したように署名されます。これには、ペイロードが 0 バイトの最後のチャンクの送信が含まれます。末尾ヘッダーは独自のチャンクとして含まれ、ペイロードが 0 バイトの最後のチャンクの後に送信されます。例えば、データが 100 KB のチャンクで終了した場合、以下を送信します。

- 以前のデータチャンク
- オブジェクトの最後の 100 KB のチャンク
- オブジェクトの終了を示す 0 バイトのチャンク
- 末尾ヘッダーチャンク

### 例: 署名計算の確認

このセクションの例を参考にして、コード内の署名計算を確認できます。例を確認する前に、次の点に注意してください。

- これらの例の署名計算では、次のサンプルのセキュリティ認証情報を使用します。

| パラメータ             | 値                                        |
|-------------------|------------------------------------------|
| AWSAccessKeyId    | AKIAIOSFODNN7EXAMPLE                     |
| AWSecretAccessKey | wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY |

- すべての例では、リクエストのタイムスタンプとして 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT) を使用します。
- すべての例では、バケット名として examplebucket を使用します。
- バケットは米国東部 (バージニア北部) リージョンにあると見なされ、認証情報の Scope と Signing Key の計算ではリージョン指定子として us-east-1 が使用されます。詳細については、「Amazon Web Services General Reference」(Amazon ウェブサービス全般のリファレンス) の「[Regions and endpoints](#)」(リージョンとエンドポイント) をご参照ください。
- パススタイルまたは仮想ホスト形式のリクエストのいずれかを使用できます。次の例では、仮想ホスト形式のリクエストを使用します。次に例を示します。

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[バケットの仮想ホスティング](#)」を参照してください。

次の例では、オブジェクトをアップロードする PUT リクエストを送信します。署名の計算では、次のことを前提としています。

- 65 KB のテキストファイルをアップロードしようとしています。ファイルの内容は文字「a」で構成される 1 文字の文字列です。
- チャンクサイズは 64 KB です。その結果、ペイロードは 3 つのチャンク (64 KB、1 KB、およびチャンクデータが 0 バイトの最後のチャンク) に分けてアップロードされます。
- 生成されるオブジェクトのキー名は chunkObject.txt です。
- x-amz-storage-class リクエストヘッダーを追加して、ストレージクラスとして REDUCED\_REDUNDANCY をリクエストしています。
- 転送には、末尾ヘッダーとして CRC32C チェックサム値が含まれています。

API アクションの詳細については、「[PutObject](#)」を参照してください。一般的なリクエストの構文は次のとおりです。

```
PUT /examplebucket/chunkObject.txt HTTP/1.1
Host: s3.amazonaws.com
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
Authorization: SignatureToBeCalculated
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 66560
x-amz-trailer: x-amz-checksum-crc32c
Content-Length: 66946
<Payload>
```

次の手順は、署名の計算を示しています。

## 1. シード署名 – 署名する文字列を作成する

### a. CanonicalRequest

```
PUT
/examplebucket/chunkObject.txt

content-encoding:aws-chunked
host:s3.amazonaws.com
x-amz-content-sha256:STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
x-amz-date:20130524T000000Z
x-amz-decoded-content-length:66560
x-amz-storage-class:REDUCED_REDUNDANCY
x-amz-trailer:x-amz-checksum-crc32c

content-encoding;host;x-amz-content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-class;x-amz-trailer
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
```

リクエストにクエリパラメータがないため、正規リクエストの 3 行目は空です。最後の行は、ハッシュされたペイロードの値として提供される定数文字列で、x-amz-content-sha256 header の値と同じである必要があります。

## b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
44d48b8c2f70eae815a0198cc73d7a546a73a93359c070abbaa5e6c7de112559
```

### Note

署名する文字列の各行については、シード署名の計算を説明する図を参照してください。

## 2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

## 3. シード署名

```
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

## 4. Authorization ヘッダー

その結果得られる Authorization ヘッダーは以下ようになります。

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=content-encoding;content-length;host;x-amz-
content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-
class,Signature=106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

## 5. チャンク 1: (65536 バイト、文字「a」の値 97)

### a. 署名するチャンク文字列。

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

 Note

署名する文字列の各行の詳細については、署名する文字列のさまざまなコンポーネントを示す前のトピック ([シード署名の計算](#)) の図を参照してください。例えば、最後の 3 行は以下で構成されます。

- *previous-signature*
- hash("")
- hash(*current-chunk-data*)

b. チャンク署名。

```
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
```

c. 送信されたチャンクデータ。

```
10000;chunk-
signature=b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
<65536-bytes>
```

6. チャンク 2: (1024 バイト、文字「a」の値 97)

a. 署名するチャンク文字列。

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. チャンク署名。

```
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
```

- c. 送信されたチャンクデータ。

```
400; chunk-  
signature=1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7  
<1024-bytes>
```

## 7. チャンク 3: (0 バイトのデータ)

- a. 署名するチャンク文字列。

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

- b. チャンク署名。

```
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

- c. 送信されたチャンクデータ。

```
0; chunk-  
signature=2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

## 8. チャンク 4: 末尾ヘッダー

- a. 署名するトレーラーチャンク文字列。

```
AWS4-HMAC-SHA256-TRAILER  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992  
1e376db7e1a34a8ef1c4bcee131a2d60a1cb62503747488624e10995f448d774
```



 Note

最後の 2 行は、previous-signature (0 バイトのデータチャンク署名) と、hash(*trailing-checksum-header-name:base64-encoded-trailing-checksum-value*\n) です。

ハッシュは空白なしで次のように計算されます。

- 末尾チェックサムヘッダー名
- コロン (:) )
- base64 でエンコードされた末尾のチェックサム値
- 改行文字 (\n)。

この例では、crc32c ハッシュアルゴリズムを使用し、base64 でエンコードされたチェックサム値 s008/Q== を使用して、次のように計算を表すことができます。hash('x-amz-checksum-crc32c:s008/Q==\n')。

## b. チャンク署名。

```
d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

## c. 送信されたチャンクデータ。

```
x-amz-checksum-crc32c:s008/Q==  
x-amz-trailer-  
signature:d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

## リクエストの認証: クエリパラメータの使用 (AWS 署名バージョン 4)

認証の概要 ([認証方法](#) を参照) で説明されているように、クエリ文字列パラメータを使用して認証情報を指定できます。リクエストを完全に URL で表現したい場合、クエリパラメータを使用してリクエストを認証するのが便利です。このメソッドは、URL の事前署名とも呼ばれます。

署名付き URL のユースケースシナリオでは、Amazon S3 リソースへの一時的なアクセスを許可できます。例えば、署名付き URL をウェブサイト埋め込んだり、コマンドラインクライアント (Curl など) で使用してオブジェクトをダウンロードしたりすることができます。

 Note

AWS CLI を使用して署名付き URL を作成することもできます。詳細については、AWS CLI コマンドリファレンスの「[presign](#)」を参照してください。

署名付き URL の例を次に示します。

```
https://examplebucket.s3.amazonaws.com/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=<your-access-key-id>/20130721/us-east-1/s3/aws4_request
&X-Amz-Date=20130721T201207Z
&X-Amz-Expires=86400
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

例の URL では、以下の点に注意してください。

- 読みやすくするために、ラインフィードが追加されています。
- URL 内の X-Amz-Credential 値には、読みやすさのためにのみ挿入された「/」文字が表示されています。実際には、%2F としてエンコードする必要があります。例えば、次のようになります。

```
&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request
```

次の表では、認証情報を提供する URL 内のクエリパラメータについて説明します。

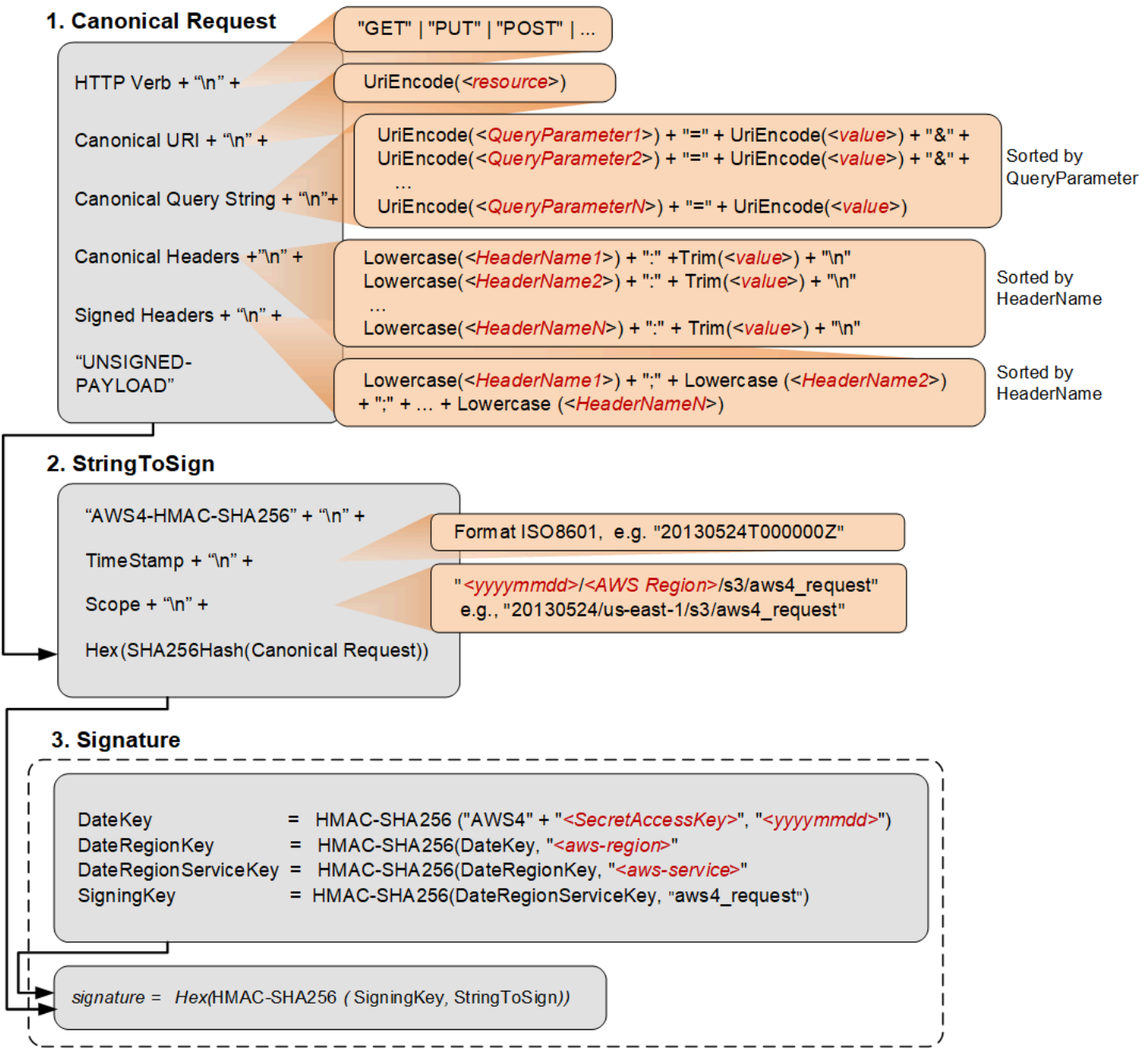
| クエリ文字列パラメータ名    | 値の例                                  |
|-----------------|--------------------------------------|
| X-Amz-Algorithm | AWS 署名のバージョンと署名の計算に使用したアルゴリズムを識別します。 |

| クエリ文字列パラメータ名     | 値の例                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | <p>AWS 署名バージョン 4 では、このパラメータ値を AWS4-HMAC-SHA256 に設定します。この文字列は、AWS 署名バージョン 4 (AWS4) および HMAC-SHA256 アルゴリズム (HMAC-SHA256) を識別します。</p>                                                                                                                                                                                                                                                                                                                                                                                                                             |
| X-Amz-Credential | <p>このパラメータは、アクセスキー ID の他に、署名が有効なスコープ (AWS リージョンおよびサービス) も指定します。この値は、次のセクションで説明する署名計算で使用するスコープと一致する必要があります。このパラメータ値の一般的な形式は次のとおりです。</p> <div data-bbox="602 737 1507 848"><pre>&lt;your-access-key-id&gt; /&lt;date&gt;/&lt;AWS #####&gt;/&lt;AWS-service&gt; /aws4_request</pre></div> <p>例えば、次のようになります。</p> <div data-bbox="602 989 1507 1108"><pre>AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request</pre></div> <p>Amazon S3 の場合、<i>AWS-service</i> 文字列は s3 です。S3 AWS-region 文字列のリストについては、「AWS 全般のリリース」の「<a href="#">リージョンとエンドポイント</a>」を参照してください。</p> |
| X-Amz-Date       | <p>日付と時刻の形式は ISO 8601 規格に準拠している必要があるため、「<i>yyyyMMddTHH:mm:ssZ</i>」形式でフォーマットする必要があります。例えば、日付と時刻が「08/01/2016 15:32:41.982-700」の場合、まず UTC (協定世界時) に変換してから「20160801T223241Z」として送信する必要があります。</p>                                                                                                                                                                                                                                                                                                                                                                    |

| クエリ文字列パラメータ名         | 値の例                                                                                                                                                                                                                                                                                                                                      |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| X-Amz-Expires        | <p>生成された署名付き URL が有効な期間を秒単位で指定します。例えば、86400 (24 時間) と指定します。この値は整数です。設定可能な最小値は 1、最大値は 604800 (7 日間) です。</p> <p>署名計算に使用する署名キーは最大 7 日間有効であるため、署名付き URL は最大 7 日間有効です。</p>                                                                                                                                                                    |
| X-Amz-SignedHeaders  | <p>署名の計算に使用したヘッダーを一覧表示します。署名計算には次のヘッダーが必要です。</p> <ul style="list-style-type: none"><li>• HTTP host ヘッダー。</li><li>• リクエストに追加する予定のすべての x-amz-* ヘッダー。</li></ul> <div><p> <b>Note</b></p><p>セキュリティの強化のために、リクエストに含める予定のすべてのリクエストヘッダーに署名する必要があります。</p></div> |
| X-Amz-Signature      | <p>リクエストを認証するための署名を指定します。この署名は、Amazon S3 が計算する署名と一致する必要があります。一致しない場合、Amazon S3 はリクエストを拒否します。</p> <p>例: 733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7</p> <p>署名計算については、次のセクションで説明します。</p>                                                                                                                              |
| X-Amz-Security-Token | <p>STS サービスから取得した認証情報を使用する場合のオプションの認証情報パラメータ。</p>                                                                                                                                                                                                                                                                                        |

# 署名の計算

次の図は、署名計算プロセスを示しています。



次の表では、図に示されている関数について説明します。これらの関数のコードを実装する必要があります。

| 関数            | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Lowercase()   | 文字列を小文字に変換します。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Hex()         | 16 進数の小文字エンコード。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| SHA256Hash()  | セキュアハッシュアルゴリズム (SHA) 暗号化ハッシュ関数。                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| HMAC-SHA256() | 指定した署名キーで SHA256 アルゴリズムを使用して HMAC を計算します。これが最後の署名です。                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Trim()        | 先頭または末尾の空白をすべて削除します。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| UriEncode()   | <p>すべてのバイトが URI でエンコードされます。UriEncode() は、以下のルールを適用する必要があります。</p> <ul style="list-style-type: none"><li>• 非予約文字 (「A」 - 「Z」、 「a」 - 「z」、 「0」 - 「9」、 「-」、「.」、「_」 および 「~」 ) を除くすべてのバイトが URI でエンコードされます。</li><li>• スペース文字は予約文字であるため、「+」ではなく「%20」としてエンコードする必要があります。</li><li>• URI で エンコードされた各バイトは、「%」とそのバイトの 2 桁の 16 進値で構成されています。</li><li>• 16 進値の文字は大文字である必要があります (例: 「%1A」 )。</li><li>• オブジェクトキー名以外では、フォワードスラッシュ文字「/」がエンコードされます。例えば、オブジェクトキー名が photos/Jan/sample.jpg の場合、キー名のフォワードスラッシュはエンコードされません。</li></ul> |

 **Important**

開発プラットフォームが提供する標準の UriEncode 関数は、実装の違いや基礎となる RFC のあいまいさにより動作しない場合があります。エンコードが確実に機能

| 関数 | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|    | <p>するように、独自のカスタム UriEncode 関数を作成することをお勧めします。</p> <p>Java の UriEncode() 関数の例を次に示します。</p> <pre> public static String UriEncode(CharSequence input,     boolean encodeSlash) {     StringBuilder result = new StringBui lder();     for (int i = 0; i &lt; input.length(); i++) {         char ch = input.charAt(i);         if ((ch &gt;= 'A' &amp;&amp; ch &lt;= 'Z')    (ch &gt;= 'a' &amp;&amp; ch &lt;= 'z')    (ch &gt;= '0' &amp;&amp; ch &lt;= '9')    ch == '_'    ch == '-'    ch == '~'    ch == '.') {             result.append(ch);         } else if (ch == '/') {             result.append(encodeSlash ? "%2F" : ch);         } else {             result.append(toHexUTF8(ch));         }     }     return result.toString(); } </pre> |

署名プロセスの詳細 (正規リクエストの作成、署名する文字列、署名計算の詳細) については、  
[「Authorization ヘッダーの署名計算: 単一チャンクでのペイロードの転送 \(AWS 署名バージョン 4\)」](#)  
を参照してください。このプロセスは一般的には同じですが、署名付き URL での CanonicalRequest  
の作成方法は、次のように異なります。

- 署名付き URL を作成する際、URL が任意のペイロードのアップロードに使用されるため、ペイロードの内容を事前に特定できません。そのため、正規リクエストにはペイロードハッシュを含めません。代わりに、定数文字列 UNSIGNED-PAYLOAD を使用します。
- 正規クエリ文字列には、X-Amz-Signature を除く前述のテーブルのすべてのクエリパラメータを含める必要があります。

- S3 では、STS サービスから取得された認証情報を使用する場合は、URL に X-Amz-Security-Token クエリパラメータを含める必要があります。
- 正規ヘッダーには HTTP host ヘッダーを含める必要があります。いずれかの x-amz-\* ヘッダーを含める場合は、署名計算のためにこれらのヘッダーも追加する必要があります。必要に応じて、リクエストに含める予定の他のすべてのヘッダーを追加できます。セキュリティを強化するには、できるだけ多くのヘッダーに署名する必要があります。署名付きヘッダーを追加し、それが署名付きクエリパラメータでも指定されている場合、それぞれの値が異なる場合、入力が競合しているため InvalidRequest エラーが発生します。

## 例

examplebucket バケットにオブジェクト test.txt があるとします。署名付き URL を作成して、このオブジェクトを 24 時間 (86,400 秒間) 他のユーザーと共有したいと考えています。

```
https://examplebucket.s3.amazonaws.com/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request
&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

次のステップでは、まず署名の計算を行い、次に署名付き URL の構築について説明します。この例では、さらに次を前提としています。

- リクエストのタイムスタンプは Fri, 24 May 2013 00:00:00 GMT です。
- バケットは米国東部 (バージニア北部) リージョンにあり、認証情報 Scope と Signing Key の計算ではリージョン指定子として us-east-1 が使用されます。リージョンの詳細については、[https://docs.aws.amazon.com/general/latest/gr/rande.html#s3\\_region](https://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region) の「AWS 全般のリファレンスリージョンとエンドポイント」を参照してください。

この例をテストケースとして使用して、お客様のコードで計算された署名を検証できます。ただし、同じバケット名、オブジェクトキー、タイムスタンプ、および次の認証情報の例を使用する必要があります。



| パラメータ              | 値                                        |
|--------------------|------------------------------------------|
| AWSAccessKeyId     | AKIAIOSFODNN7EXAMPLE                     |
| AWSSecretAccessKey | wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY |

## 1. StringToSign

### a. CanonicalRequest

```
GET
/test.txt
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE
%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
host:examplebucket.s3.amazonaws.com

host
UNSIGNED-PAYLOAD
```

### b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
3bfa292879f6447bbcda7001decf97f4a54dc650c8942174ae0a9121cf58ad04
```

## 2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

## 3. Signature



認証を必要とするリクエストを行う場合、AWS SDK を使用していないのであれば、リクエストの認証情報を提供する署名を計算するコードを記述する必要があります。AWS Signature Version 4 ([リクエストの認証 \(AWS 署名バージョン 4\)](#)) を参照) での署名計算は複雑な作業になる場合があるため、可能な限り AWS SDK を使用することをお勧めします。

このセクションでは、Java および C# で記述された署名計算の例を示します。コードサンプルは次のリクエストを送信し、HTTP Authorization ヘッダーを使用して認証情報を提供します。

- PUT オブジェクト – それぞれの例は、ペイロード全体を一度にアップロードする方法と、ペイロードをチャンク単位でアップロードする方法の両方を示しています。認証に Authorization ヘッダーを使用する方法については、「[リクエストの認証: Authorization ヘッダーの使用 \(AWS 署名バージョン 4\)](#)」を参照してください。
- GET オブジェクト – この例では、オブジェクトを取得するための署名付き URL を生成します。クエリパラメータは、署名およびその他の認証情報を提供します。ユーザーは、署名付き URL をブラウザに貼り付けてオブジェクトを取得するか、URL を使用してクリック可能なリンクを作成できます。認証にクエリパラメータを使用する方法については、「[リクエストの認証: クエリパラメータの使用 \(AWS 署名バージョン 4\)](#)」を参照してください。

このセクションの残りの部分では、Java および C# の例について説明します。トピックには、サンプルのダウンロードおよび実行の手順が含まれています。

## Java を使用した署名計算の例 (AWS 署名バージョン 4)

署名計算を示す Java のサンプルは、<https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AWSS3SigV4JavaSamples.zip> からダウンロードできます。これらのサンプルでは AWS SDK for Java v1 を使用していますが、新しいアプリケーションには [AWS SDK for Java v2](#) を使用することをお勧めします。RunAllSamples.java の main() 関数は、サンプルリクエストを実行して、オブジェクトの作成、オブジェクトの取得、オブジェクトの署名付き URL の作成を行います。このサンプルは、コードで指定されたテキスト文字列からオブジェクトを作成します。

```
PutS3ObjectSample.putS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
GetS3ObjectSample.getS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
PresignedUrlSample.getPresignedUrlToS3Object(bucketName, regionName, awsAccessKey,
awsSecretKey);
PutS3ObjectChunkedSample.putS3ObjectChunked(bucketName, regionName, awsAccessKey,
awsSecretKey);
```

## Linux ベースのコンピュータで例をテストするには

以下の手順は、Linux オペレーティングシステムを対象としています。

1. ターミナルで、AWS3SigV4JavaSamples.zip が含まれるディレクトリに移動します。
2. .zip ファイルを展開します。
3. テキストエディタで、./com/amazonaws/services/s3/samples/RunAllSamples.java ファイルを開きます。次の情報を使用してコードを更新します。
  - 新しいオブジェクトを作成できるバケットの名前。

### Note

この例では、仮想ホスト形式のリクエストを使用してバケットにアクセスします。潜在的なエラーを回避するため、「Amazon Simple Storage Service ユーザーガイド」の「[バケットの制約と制限](#)」で説明されているように、バケット名がバケット命名規則に準拠していることを確認してください。

- バケットが存在する AWS リージョン。

バケットが米国東部 (バージニア北部) リージョンにある場合は、us-east-1 を使用してリージョンを指定します。その他の AWS リージョンのリストについては、「AWS 全般のリファレンス」の「[Amazon Simple Storage Service \(S3\)](#)」を参照してください。

4. ソースコードをコンパイルし、コンパイルされたクラスを bin/ ディレクトリに保存します。

```
javac -d bin -source 6 -verbose com
```

5. ディレクトリを bin/ に変更し、RunAllSamples を実行します。

```
java com.amazonaws.services.s3.sample.RunAllSamples
```

このコードは、main() のすべてのメソッドを実行します。各リクエストについて、出力には正規リクエスト、署名する文字列、署名が表示されます。

## C# を使用した署名計算の例 (AWS 署名バージョン 4)

署名計算を示す C# のサンプルは、[https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4\\_Samples\\_CSharp.zip](https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip) からダウンロードできます。これらのサンプルは AWS SDK for .NET v1 を使用していますが、新しいアプリケーションには [AWS SDK for .NET v3](#) を使用することをお勧めします。Program.cs の main() 関数は、サンプルリクエストを実行して、オブジェクトの作成、オブジェクトの取得、オブジェクトの署名付き URL の作成を行います。署名計算のコードは \Signers フォルダにあります。

```
PutS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n*****");
PutS3ObjectChunkedSample.Run(awsRegion, bucketName, "MySampleFileChunked.txt");

Console.WriteLine("\n\n*****");
GetS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n*****");
PresignedUrlSample.Run(awsRegion, bucketName, "MySampleFile.txt");
```

Microsoft Visual Studio 2010 以降で例をテストするには

1. .zip ファイルを展開します。
2. Visual Studio を起動し、.sln ファイルを開きます。
3. 有効なセキュリティ認証情報を使用して App.config ファイルを更新します。
4. 次のようにコードを更新します。

- Program.cs で、バケット名とバケットが存在する AWS リージョンを指定します。このサンプルは、このバケット内にオブジェクトを作成します。

5. コードを実行します。
6. オブジェクトが作成されたことを確認するには、プログラムが作成した署名付き URL をコピーし、ブラウザのウィンドウに貼り付けます。

# リクエストの認証: POST (AWS 署名バージョン 4) を使用したブラウザベースのアップロード

Amazon S3 は HTTP POST リクエストをサポートしているため、ユーザーはコンテンツを Amazon S3 に直接アップロードできます。Amazon S3 にデータを保存するために HTTP POST を使用してコンテンツをアップロードすると、簡単にアップロードでき、アップロードのレイテンシーが低減されます。このセクションでは、HTTP POST リクエストを認証する方法について説明します。HTTP POST リクエスト、フォームの作成方法、POST ポリシーの作成方法、および例の詳細については、「[POST \(AWS 署名バージョン 4\) を使用したブラウザベースのアップロード](#)」を参照してください。

HTTP POST リクエストを認証するには、次の手順を実行します。

1. Amazon S3 がリクエストを受信したときに、署名を再計算するために使用できる署名および関連情報を提供するために、フォームには次のフィールドが含まれている必要があります。

| 要素名              | 説明                                                                                                                                                                                                                |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| policy           | リクエストで許可されているものについて説明する Base64 でエンコードされたセキュリティポリシー。署名の計算において、このポリシーが署名対象となる文字列です。Amazon S3 は署名を再計算できるように、このポリシーを取得する必要があります。                                                                                      |
| x-amz-algorithm  | 使用される署名アルゴリズムです。AWS 署名バージョン 4 の場合、値は AWS4-HMAC-SHA256 です。                                                                                                                                                         |
| x-amz-credential | アクセスキー ID に加えて、署名計算用の署名キーの計算に使用したスコープ情報を提供します。<br><br>これは、次のような形式の文字列です。<br><br><code>&lt;your-access-key-id&gt; /&lt;date&gt;/&lt;aws-region&gt; /&lt;aws-service&gt; /aws4_request</code><br><br>例えば、次のようになります。 |

| 要素名             | 説明                                                                                                                                                                                                         |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 | <p>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .</p> <p>Amazon S3 の場合、aws-service 文字列は s3 です。Amazon S3 aws-region 文字列のリストについては、「AWS 全般のリファレンス」の「<a href="#">リージョンとエンドポイント</a>」を参照してください。</p> |
| x-amz-date      | <p>ISO8601 形式の日付値です。例えば、20130728T000000Z。</p> <p>署名キーの作成に使用したのと同じ日付です。これは、署名したポリシー (x-amz-date ) で指定した値と同じである必要があります。</p>                                                                                  |
| x-amz-signature | <p>(AWS 署名バージョン 4) セキュリティポリシーの HMAC-SHA256 ハッシュ。</p> <p>署名のオプションの詳細については、「AWS 全般のリファレンス」の「<a href="#">HTTP リクエストに署名を追加する</a>」を参照してください。</p>                                                                |

## 2. POST ポリシーには以下の要素が含まれます。

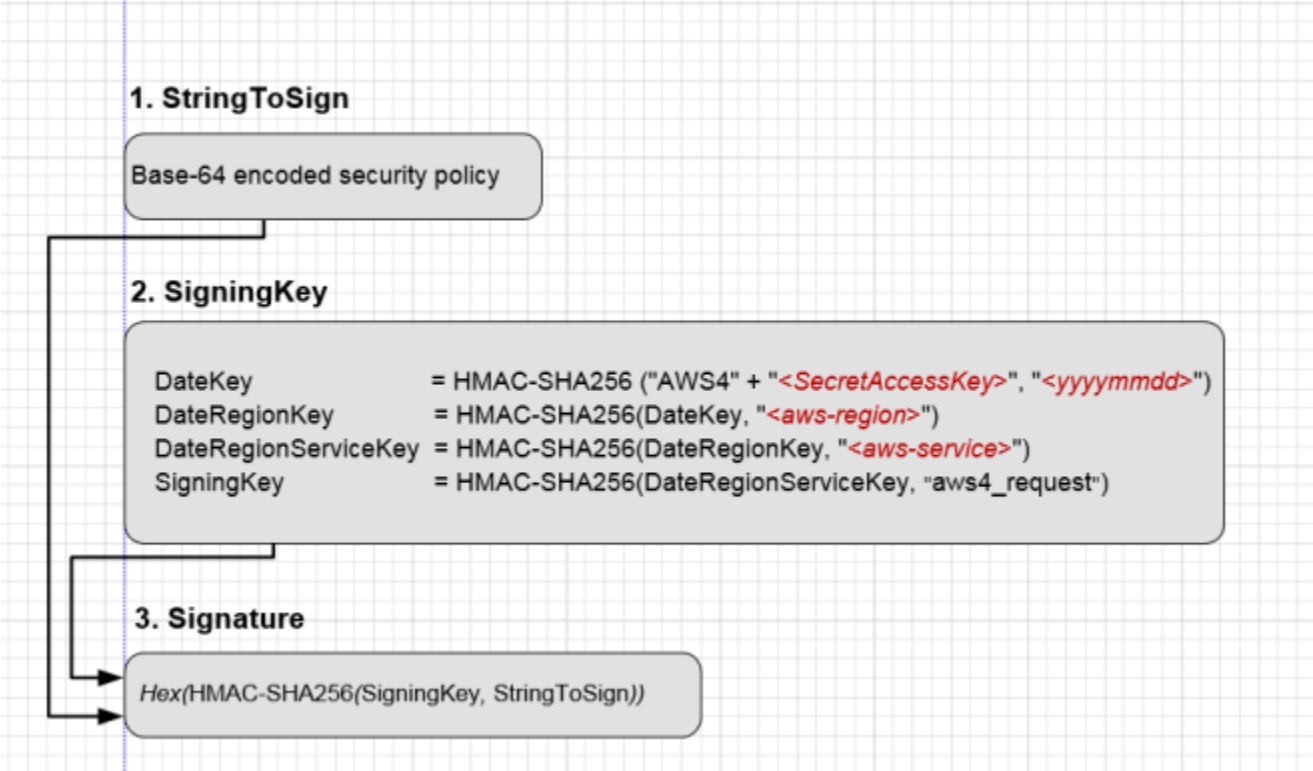
| 要素名              | 説明                                                                              |
|------------------|---------------------------------------------------------------------------------|
| x-amz-algorithm  | <p>署名の計算に使用した署名アルゴリズム。AWS 署名バージョン 4 の場合、値は AWS4-HMAC-SHA256 です。</p>             |
| x-amz-credential | <p>アクセスキー ID に加えて、署名計算用の署名キーの計算に使用したスコープ情報を提供します。</p> <p>これは、次のような形式の文字列です。</p> |

| 要素名        | 説明                                                                                                                                                                                               |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            | <div>&lt;your-access-key-id&gt; /&lt;date&gt;/&lt;aws-region&gt; /&lt;aws-service&gt; /aws4_request</div> <div>例えば、</div> <div>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .</div> |
| x-amz-date | ISO8601 形式の文字列で指定された日付値。例えば、「20130728T000000Z」。日付は、署名計算用の署名キーの作成に使用した日付と同じである必要があります。                                                                                                            |

3. 署名の計算において、POST ポリシーが署名する文字列です。

## 署名の計算

次の図は、署名計算プロセスを示しています。





## 署名を計算するには

1. UTF-8 エンコーディングを使用してポリシーを作成します。
2. UTF-8 エンコードされたポリシーを Base64 に変換します。その結果が、署名する文字列です。
3. 署名する文字列の HMAC-SHA256 ハッシュとして署名を作成します。署名キーをハッシュ関数のキーとして提供します。
4. 16 進エンコードを使用して署名をエンコードします。

HTML フォーム、セキュリティポリシー、および例の作成の詳細については、次のサブトピックを参照してください。

- [HTML フォームの作成 \(AWS 署名バージョン 4 を使用\)](#)
- [POST ポリシー](#)
- [例: HTTP POST \(AWS 署名バージョン 4\) を使用したブラウザベースのアップロード](#)

## Amazon S3 署名バージョン 4 の認証固有のポリシーキー

次の表は、Amazon S3 ポリシーで利用できる Amazon S3 署名バージョン 4 認証に関連するポリシーキーを示しています。これらの条件をバケットポリシーに追加して、Signature Version 4 を使用してリクエストが認証された際の特定の動作を適用できます。エンドポイントポリシーの例については、「[署名バージョン 4 に関連する条件キーを使用するバケットポリシーの例](#)」を参照してください。

| 適用できるキー             | 説明                                                                                                                                                    |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| s3:signatureversion | 認証されたリクエストでサポートされる AWS 署名のバージョンを識別します。認証されたリクエストについて、Amazon S3 は署名バージョン 4 と署名バージョン 2 の両方をサポートしています。この条件をバケットポリシーに追加して、特定の署名バージョンを要求できます。<br><br>有効な値: |

| 適用できるキー     | 説明                                                                                                                                                                                                                                                                    |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <p>"AWS" は署名バージョン 2 を識別します</p> <p>"AWS4-HMAC-SHA256" は署名バージョン 4 を識別します</p>                                                                                                                                                                                            |
| s3:authType | <p>Amazon S3 は、さまざまな認証方式をサポートしています (<a href="#">リクエストの認証 (AWS 署名バージョン 4)</a> を参照)。必要に応じてこの条件キーを使用し、受信するリクエストが特定の認証方法を使用するように制限することができます。例えば、リクエスト認証のために HTTP Authorization ヘッダーのみの使用を指定できます。</p> <p>有効な値:</p> <p>REST-HEADER</p> <p>REST-QUERY-STRING</p> <p>POST</p> |

| 適用できるキー                      | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>s3:signatureAge</code> | <p>認証されたリクエストの中で署名が有効である時間長 (ミリ秒単位)。</p> <p>この条件は、以下に適用されます。</p> <ul style="list-style-type: none"><li>• 署名付き URL – 最も制限の厳しい条件が優先されます。詳細については、「<a href="#">署名付き URL の使用</a>」を参照してください。</li><li>• 署名付き POST – 署名付き POST を使用してファイルを S3 に直接アップロードします。詳細については、「<a href="#">Amazon S3 POST ポリシー</a>」を参照してください。</li></ul> <p>署名バージョン 2 では、この値は常に 0 に設定されます。</p> <p>Signature Version 4 では、署名キーは最大 7 日間有効です。したがって、署名の有効期間も最大 7 日間となります。この条件を使用すると、署名の有効期間をさらに制限できます。詳細については、「<a href="#">リクエストへの署名の概要</a>」を参照してください。</p> <p>値の例: 100</p> |

| 適用できるキー                 | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| s3:x-amz-content-sha256 | <p>この条件キーを使用すると、バケット内にある署名されていないコンテンツを許可しません。</p> <p>Authorization ヘッダを使用するリクエストに対し Signature Version 4 を使用する場合、署名計算に x-amz-content-sha256 ヘッダーを追加した後、その値をハッシュペイロードに設定します。この条件キーは、クエリ文字列パラメータとして x-amz-content-sha256 ヘッダーをサポートしていないことに注意してください。</p> <p>この条件キーをバケットポリシーで使用する、ペイロードに署名がないアップロードを、すべて拒否することができます。例えば、リクエスト認証において、Authorization ヘッダーを使用しているものの、ペイロードに署名していないアップロードを拒否することができます。詳細については、「<a href="#">Authorization ヘッダーの署名計算: 単一チャンクでのペイロードの転送 (AWS 署名バージョン 4)</a>」を参照してください。</p> <p>有効な値: UNSIGNED-PAYLOAD</p> |

## 署名バージョン 4 に関連する条件キーを使用するバケットポリシーの例

次のバケットポリシーは、署名が作成されてから 10分以上経過している場合、examplebucket 内のオブジェクトに対する Amazon S3 の署名付き URL リクエストを拒否します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```

    {
      "Sid": "Deny a presigned URL request if the signature is more than 10
min old",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket3/*",
      "Condition": {
        "NumericGreaterThan": {
          "s3:signatureAge": 600000
        }
      }
    }
  ]
}

```

次のバケットポリシーでは、リクエスト認証に Authorization ヘッダーを使用しているリクエストのみを許可します。POST リクエストまたは署名付き URL リクエストはすべて拒否されます。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow only requests that use Authorization header for
request authentication. Deny POST or presigned URL requests.",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket3/*",
      "Condition": {
        "StringNotEquals": {
          "s3:authType": "REST-HEADER"
        }
      }
    }
  ]
}

```

次のバケットポリシーは、リクエスト認証に署名付き URL を使用するリクエストを拒否します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyUploadsUsingPresignedURL",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
      "Condition": {
        "StringEquals": {
          "s3:authType": "REST-query-string"
        }
      }
    }
  ]
}
```

次のバケットポリシーは、署名されていないペイロードを含むすべてのアップロードを拒否します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Deny uploads with unsigned payloads that use the
Authorization header.",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket3/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-content-sha256": "UNSIGNED-PAYLOAD"
        }
      }
    }
  ]
}
```

```
    }  
  ]  
}
```

# POST (AWS 署名バージョン 4) を使用したブラウザベースのアップロード

このセクションでは、HTTP POST リクエストを使用してブラウザを介して Amazon S3 に直接ファイルをアップロードする方法について説明します。また、ブラウザベースの Amazon S3 へのファイルアップロードに AWS Amplify JavaScript ライブラリを使用する方法に関する情報も含まれています。

## トピック

- [POST Object](#)
- [POST Object restore](#)
- [HTTP POST を使用したブラウザベースのアップロード](#)
- [署名の計算](#)
- [HTML フォームの作成 \(AWS 署名バージョン 4 を使用\)](#)
- [POST ポリシー](#)
- [例: HTTP POST \(AWS 署名バージョン 4\) を使用したブラウザベースのアップロード](#)
- [AWS Amplify ライブラリを使用した Amazon S3 へのブラウザベースのアップロード](#)



# POST Object

## 説明

POST オペレーションは、HTML フォームを使用して指定されたバケットにオブジェクトを追加します。POST は、バケットにオブジェクトを配置する方法としてブラウザベースのアップロードを可能にする PUT の代替形式です。HTTP ヘッダーを介して PUT に渡されるパラメータは、代わりにマルチパート/フォームデータエンコードされたメッセージ本文のフォームフィールドとして POST に渡されます。オブジェクトをバケットに追加するには、バケットへの WRITE アクセス権が必要です。Amazon S3 は部分的なオブジェクトを保存しません。成功したレスポンスを受け取った場合は、オブジェクト全体が保存されたことを確信できます。

Amazon S3 は分散システムです。バケットのバージョニングを有効にしていない場合、Amazon S3 が同じオブジェクトに対して複数の書き込みリクエストを同時に受信すると、最後に書き込まれたバージョンのオブジェクトのみが保存されます。

ネットワークのトラバース中にデータが破損しないようにするには、Content-MD5 フォームフィールドを使用します。このフォームフィールドを使用すると、Amazon S3 は指定された MD5 値に対してオブジェクトをチェックします。一致しない場合、Amazon S3 はエラーを返します。さらに、オブジェクトを Amazon S3 に投稿しながら MD5 値を計算し、返された ETag を計算された MD5 値と比較できます。ETag は、オブジェクトのコンテンツに加えた変更のみを反映し、メタデータに加えた変更は反映しません。

### Note

リクエスト本文を送信する前にリクエストヘッダーを送信するようにアプリケーションを設定するには、HTTP ステータスコード 100 (続行) を使用します。POST オペレーションの場合、このステータスコードを使用すると、メッセージがヘッダーに基づいて拒否された場合 (認証の失敗やリダイレクトなど)、メッセージ本文の送信を回避できます。HTTP ステータスコード 100 (続行) の詳細については、<http://www.ietf.org/rfc/rfc2616.txt> のセクション 8.2.3 を参照してください。

Amazon S3 は、S3 バケットにコピーされたすべての新しいオブジェクトを自動的に暗号化します。アップロードされたオブジェクトの暗号化設定は、送信先バケットのデフォルトの暗号化設定によって異なります。デフォルトでは、すべてのバケットには、Amazon S3 マネージドキーによるサーバー側の暗号化 (SSE-S3) を使用するデフォルトの暗号化設定が指定されています。

送信先バケットに、AWS Key Management Service (AWS KMS) キーを使用するサーバー側の暗号化 (SSE-KMS)、AWS KMS キーを使用する二層式サーバー側の暗号化 (DSSE-KMS)、または顧客提供の暗号化キーを使用する暗号化 (SSE-C) が適用されている場合、Amazon S3 は対応する KMS キーまたは顧客提供のキーを使用して、アップロードされたオブジェクトを暗号化します。オブジェクトをアップロードするときに、アップロードされたオブジェクトの暗号化設定を変更する場合は、サーバー側の暗号化のタイプを指定できます。SSE-S3、SSE-KMS、DSSE-KMS、または SSE-C を設定できます。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[サーバー側の暗号化を使用したデータの保護](#)」を参照してください。

#### Important

リクエストを作成するときは、file フィールドがフォームの最後のフィールドであることを確認してください。

## バージョニング

バケットのバージョニングを有効にすると、POST は追加されるオブジェクトに対して一意のバージョン ID を自動的に生成します。Amazon S3 は、x-amz-version-id レスポンスヘッダーを使用してこの ID をレスポンスに返します。

バケットのバージョニングを停止した場合、Amazon S3 は常にバケットに保存されているオブジェクトのバージョン ID として null を使用します。

バケットのバージョニング状態を返す方法の詳細については、「[GetBucketVersioning](#)」を参照してください。

Amazon S3 は分散システムです。バケットのバージョニングが有効な場合、Amazon S3 が同じオブジェクトに対して複数の書き込みリクエストを同時に受信すると、すべてのバージョンのオブジェクトが保存されます。

バージョニングを使用するサンプルリクエストを確認するには、「[サンプルリクエスト](#)」を参照してください。

## リクエスト

### 構文

POST / HTTP/1.1

Host: *destinationBucket*.s3.amazonaws.com

```
User-Agent: browser_data
Accept: file_types
Accept-Language: Regions
Accept-Encoding: encoding
Accept-Charset: character_set
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: length

--9431149156168
Content-Disposition: form-data; name="key"

acl
--9431149156168
Content-Disposition: form-data; name="tagging"

<Tagging><TagSet><Tag><Key>Tag Name</Key><Value>Tag Value</Value></Tag></TagSet></Tagging>
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

success_redirect
--9431149156168
Content-Disposition: form-data; name="Content-Type"

content_type
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

uuid
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

metadata
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

access-key-id
--9431149156168
Content-Disposition: form-data; name="Policy"

encoded_policy
--9431149156168
```

```
Content-Disposition: form-data; name="Signature"

signature=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

file_content
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

## リクエストパラメータ

オペレーションの実装では、リクエストパラメータを使用しません。

## フォームのフィールド

このオペレーションでは、次のフォームフィールドを使用できます。

| 名前             | 説明                                                                                                                                                                                 | 必須   |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| AWSAccessKeyId | <p>ポリシー内の一連の制約を満たすリクエストに対して匿名ユーザーアクセスを付与するバケット所有者の AWS アクセスキー ID。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> <p>制約: ポリシードキュメントがリクエストに含まれる場合に必要です。</p>                                    | 条件付き |
| acl            | <p>指定された Amazon S3 のアクセスコントロールリスト (ACL)。指定された ACL が有効でない場合、エラーが生成されます。ACL の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">アクセスコントロールリスト (ACL) の概要</a>」を参照してください。</p> | いいえ  |

| 名前                                                                              | 説明                                                                                                                                                                        | 必須  |
|---------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|                                                                                 | タイプ: 文字列<br><br>デフォルト: プライベート<br><br>有効な値: private   public-read   public-read-write   aws-exec-read   authenticated-read   bucket-owner-read   bucket-owner-full-control |     |
| Cache-Control , Content-Type , Content-Disposition , Content-Encoding , Expires | REST 固有ヘッダー。詳細については、「 <a href="#">PutObject</a> 」を参照してください。<br><br>タイプ: 文字列<br><br>デフォルト: なし                                                                              | いいえ |
| file                                                                            | ファイルまたはテキストコンテンツ。<br><br>ファイルまたはテキストコンテンツはフォームの最後のフィールドでなければなりません。<br><br>複数のファイルを一度にアップロードすることはできません。<br><br>タイプ: ファイルまたはテキストコンテンツ<br><br>デフォルト: なし                      | はい  |

| 名前     | 説明                                                                                                                                                                                                                                                                                                                                                                                    | 必須   |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| key    | <p>アップロードされたキーの名前。</p> <p>ユーザーによって指定されたファイル名を使用するには、<code>\${filename}</code> 変数を使用します。例えば、Mary というユーザーが <code>example.jpg</code> というファイルをアップロードし、<code>/user/mary/\${filename}</code> を指定した場合、キー名は <code>/user/mary/example.jpg</code> になります。</p> <p>詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">オブジェクトメタデータの使用</a>」を参照してください。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> | はい   |
| policy | <p>リクエストで許可されているものについて説明するセキュリティポリシー。セキュリティポリシーがないリクエストは匿名と見なされ、誰でも書き込むことができるバケットでのみ動作します。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">HTML フォーム</a>」と「<a href="#">アップロード例</a>」を参照してください。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> <p>制約: バケットがパブリックに書き込み可能でない場合は、セキュリティポリシーが必要です。</p>                                                                                    | 条件付き |

| 名前                                 | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 必須  |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| success_action_redirect , redirect | <p>アップロードが成功したときにクライアントがリダイレクトされる URL。</p> <p>success_action_redirect が指定されていない場合、Amazon S3 は success_action_status フィールドで指定されている空のドキュメントタイプを返します。</p> <p>Amazon S3 が URL を解釈できない場合、フィールドが存在しないかのように動作します。</p> <p>アップロードが失敗した場合、Amazon S3 はエラーを表示し、ユーザーを URL にリダイレクトしません。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> <div><div> Note</div><div>redirect フィールド名は廃止されています。redirect フィールド名のサポートは削除される予定です。</div></div> | いいえ |

| 名前                    | 説明                                                                                                                                                                                                                                                                                                                                                                                              | 必須  |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| success_action_status | <p>success_action_redirect を指定しない場合、アップロードが成功するとステータスコードがクライアントに返されます。</p> <p>このフィールドは、200、201、または 204 (デフォルト) の値を受け入れます。</p> <p>値が 200 または 204 に設定されている場合、Amazon S3 は空のドキュメントとステータスコード 200 または 204 を返します。</p> <p>値が 201 に設定されている場合、Amazon S3 は XML ドキュメントとステータスコード 201 を返します。</p> <p>値が設定されていないか有効でない値が設定されている場合は、Amazon S3 は空のドキュメントとステータスコード 204 を返します。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> | いいえ |



| 名前      | 説明                                                                                                                                                                                                                                                                                                                                                                                                              | 必須  |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| tagging | <p>オブジェクトに追加する指定されたタグのセット。タグを追加するには、次のエンコーディングスキームを使用します。</p> <div><pre>&lt;Tagging&gt;   &lt;TagSet&gt;     &lt;Tag&gt;       &lt;Key&gt;TagName&lt;/Key&gt;       &lt;Value&gt;TagValue&lt;/Value&gt;     &lt;/Tag&gt;     ...   &lt;/TagSet&gt; &lt;/Tagging&gt;</pre></div> <p>詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">オブジェクトのタグ付け</a>」を参照してください。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> | いいえ |

| 名前                  | 説明                                                                                                                                                                                                                                                                                                                                                                                                  | 必須  |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| x-amz-storage-class | <p>オブジェクトの保存に使われるストレージクラス。クラスを指定しない場合、Amazon S3 はデフォルトのストレージクラス STANDARD を使用します。Amazon S3 は他のストレージクラスをサポートしています。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">ストレージクラス</a>」を参照してください。</p> <p>タイプ: 文字列</p> <p>デフォルト: STANDARD</p> <p>有効な値: REDUCED_REDUNDANCY EXPRESS_ONEZONE   DEEP_ARCHIVE   GLACIER   GLACIER_IR   INTELLIGENT_TIERING   ONEZONE_IA   STANDARD   STANDARD_IA  </p> | いいえ |
| x-amz-meta-*        | <p>このプレフィックスで始まるヘッダーはユーザー定義メタデータです。それぞれが保存され、キーと値のペアのセットとして返されます。Amazon S3 はユーザー定義メタデータを検証または解釈しません。詳細については、「<a href="#">PutObject</a>」を参照してください。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p>                                                                                                                                                                                                              | いいえ |

| 名前                   | 説明                                                                                                                                                                 | 必須   |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| x-amz-security-token | <p>Amazon DevPay のセキュリティトークン。</p> <p>Amazon DevPay を使用する各リクエストでは、製品トークン用とユーザートークン用の 2 つの x-amz-security-token フォームフィールドが必要です。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> | いいえ  |
| x-amz-signature      | <p>(AWS 署名バージョン 4) セキュリティポリシーの HMAC-SHA256 ハッシュ。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p>                                                                              | 条件付き |

| 名前                              | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 必須  |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| x-amz-website-redirect-location | <p>バケットがウェブサイトとして設定されている場合、このフィールドはこのオブジェクトに対するリクエストを、同じバケット内の別のオブジェクトか外部 URL にリダイレクトします。Amazon S3 はこのヘッダー値をオブジェクトメタデータに保存します。オブジェクトメタデータの詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">オブジェクトキーとメタデータ</a>」を参照してください。</p> <p>次の例では、リクエストヘッダーは同じバケット内のオブジェクト (anotherPage.html ) へのリダイレクトを設定します。</p> <pre>x-amz-website-redirect-location: /anotherPage.html</pre> <p>次の例では、リクエストヘッダーによって、オブジェクトのリダイレクト先が別のウェブサイトに設定されています。</p> <pre>x-amz-website-redirect-location: http://www.example.com/</pre> <p>Amazon S3 でのウェブサイトのホスティングに関する詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">Amazon S3 でのウェブサイトのホスティング</a>」および「<a href="#">ウェブサイトページのリダイレクトを設定する方法</a>」を参照してください。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> | いいえ |

| 名前 | 説明                                                                       | 必須 |
|----|--------------------------------------------------------------------------|----|
|    | 制約: 値には /、http://、または https:// のプレフィックスを付ける必要があります。値の長さは 2 KB に制限されています。 |    |

## 追加のチェックサムリクエストフォームフィールド

オブジェクトをアップロードするときに、データの整合性を検証するために使用するさまざまなチェックサムを指定できます。Amazon S3 で使用する追加のチェックサムアルゴリズムを 1 つ指定できます。追加のチェックサム値の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[オブジェクトの整合性の確認](#)」を参照してください。

| 名前                       | 説明                                                                                                                                                           | 必須   |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| x-amz-checksum-algorithm | オブジェクトのチェックサムを作成するために使用されるアルゴリズムを示します。値を指定する場合は、一致するチェックサムヘッダーを含める必要があります。そうしないと、リクエストは 400 エラーを生成します。<br><br>指定できる値には、CRC32、CRC32C、SHA1、および SHA256 などがあります。 | いいえ  |
| x-amz-checksum-crc32     | オブジェクトの base64 エンコードされた 32 ビット CRC32 チェックサムを指定します。<br><br>このパラメータは、x-amz-checksum-algorithm の値が CRC32 の場合に必要です。                                              | 条件付き |
| x-amz-checksum-crc32c    | オブジェクトの base64 エンコードされた 32 ビット CRC32C チェックサムを指定します。<br><br>このパラメータは、x-amz-checksum-algorithm の値が CRC32C の場合に必要です。                                            | 条件付き |
| x-amz-checksum-sha1      | オブジェクトの base64 エンコードされた 160 ビット SHA-1 ダイジェストを指定します。                                                                                                          | 条件付き |

| 名前                    | 説明                                                                                                                  | 必須   |
|-----------------------|---------------------------------------------------------------------------------------------------------------------|------|
|                       | このパラメータは、x-amz-checksum-algorithm の値が SHA1 の場合に必要です。                                                                |      |
| x-amz-checksum-sha256 | オブジェクトの base64 エンコードされた 256 ビット SHA-256 ダイジェストを指定します。<br><br>このパラメータは、x-amz-checksum-algorithm の値が SHA256 の場合に必要です。 | 条件付き |

## サーバー側の暗号化固有のリクエストフォームフィールド

サーバー側の暗号化は、保存時のデータ暗号化です。Amazon S3 は、AWS データセンターのディスクに書き込まれるときにデータを暗号化し、お客様がデータにアクセスするときに復号します。オブジェクトをアップロードする際には、そのオブジェクトの暗号化に Amazon S3 が使用するサーバー側の暗号化の種類を指定できます。

サーバー側の暗号化には 4 つのタイプがあります。

- Amazon S3 マネージドキーを使用したサーバー側の暗号化 (SSE-S3) – 2022 年 5 月以降、すべての Amazon S3 バケットでは、デフォルトで暗号化が設定されています。サーバー側の暗号化のデフォルトのオプションは、SSE-S3 を使用しています。一意のキーで各オブジェクトを暗号化します。追加の保護措置として、SSE-S3 は定期的にローテーションされるルートキーを使ってキーそのものを暗号化します。SSE-S3 は、利用可能な最強のブロック暗号の 1 つである 256 ビットの高度暗号化規格 (AES-256) を使用して、データを暗号化します。
- AWS KMS キーを使用したサーバー側の暗号化 (SSE-KMS) – SSE-KMS は、AWS KMS サービスと Amazon S3 の統合によって提供されます。AWS KMS を使用すると、キーをより細かく制御できます。例えば、個別のキーを表示し、制御ポリシーを編集して、AWS CloudTrail のキーに従うことができます。さらに、カスターマネージドキーを作成および管理したり、ユーザー、サービス、およびリージョンに固有の AWS マネージドキーを使用したりできます。
- AWS KMS キーを使用した二層式サーバー側暗号化 (DSSE-KMS) – AWS KMS keys キーを使用した二層式サーバー側暗号化 (DSSE-KMS) は、SSE-KMS と似ていますが、1 つのレイヤーの代わりに 2 つの個別のオブジェクトレベルの暗号化レイヤーを適用します。
- 顧客提供キーによるサーバー側の暗号化 (SSE-C) – SSE-Cでは、お客様が暗号化キーを管理し、Amazon S3 はディスクに書き込む際の暗号化とオブジェクトにアクセスする際の復号を管理します。

 Note

汎用バケットに対してお客様が用意したキー (SSE-C) によるサーバー側の暗号化がブロックされている場合、バケットに新しいデータを書き込むときに SSE-C リクエストヘッダーを指定すると、HTTP 403 Access Denied エラーが発生します。詳細については、[「汎用バケットの SSE-C のブロックまたはブロック解除」](#)を参照してください。

詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[サーバー側の暗号化を使用したデータの保護](#)」を参照してください。

使用するサーバー側の暗号化のタイプに応じて、次のフォームフィールドを指定します。

- SSE-S3、SSE-KMS、または DSSE-KMS を使用する – これらのタイプのサーバー側の暗号化を使用する場合は、リクエストで次のフォームフィールドを指定します。

| 名前                                          | 説明                                                                                                                                                                                                                   | 必須                                              |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| x-amz-server-side-encryption                | <p>Amazon S3 がオブジェクトを作成するときに使用するサーバー側の暗号化アルゴリズムを指定します。SSE-S3 を使用するには、AES256 を指定します。SSE-KMS を使用するには、aws:kms を指定します。DSSE-KMS を使用するには、aws:kms:dsse を指定します。</p> <p>タイプ: 文字列</p> <p>有効な値: aws:kms、AES256、aws:kms:dsse</p> | はい                                              |
| x-amz-server-side-encryption-aws-kms-key-id | <p>x-amz-server-side-encryption ヘッダーの有効な値が aws:kms または aws:kms:dsse の場合、このヘッダーはオブジェクトの暗号化に使用された AWS KMS key の ID を指定します。</p> <p>タイプ: 文字列</p>                                                                         | はい(x-amz-server-side-encryption の値が aws:kms または |

| 名前                                              | 説明                                                                                                                                                                                                                                               | 必須                |
|-------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
|                                                 |                                                                                                                                                                                                                                                  | aws:kms:dsse の場合) |
| x-amz-server-side-encryption-context            | <p>x-amz-server-side-encryption の有効な値が aws:kms または aws:kms:dsse の場合、このヘッダーはオブジェクトの暗号化コンテキストを指定します。このヘッダーの値は、JSON 形式のキーと値のペアを含む UTF-8 文字列を base64 でエンコードした文字列です。</p> <p>タイプ: 文字列</p>                                                              | いいえ               |
| x-amz-server-side-encryption-bucket-key-enabled | <p>x-amz-server-side-encryption の有効な値が aws:kms または aws:kms:dsse の場合、このヘッダーは Amazon S3 が SSE-KMS または DSSE-KMS で S3 バケットキーを使用するかどうかを指定します。このヘッダーを true に設定すると、Amazon S3 は SSE-KMS または DSSE-KMS によるオブジェクト暗号化に S3 バケットキーを使用します。</p> <p>タイプ: ブール値</p> | いいえ               |

**Note**

x-amz-server-side-encryption:aws:kms または x-amz-server-side-encryption:aws:kms:dsse を指定しても x-amz-server-side-encryption-aws-kms-key-id を指定しない場合、Amazon S3 は AWS マネージドキー (aws/S3) を使用してデータを保護します。

- SSE-C を使用する – 独自の暗号化キーを管理する場合は、リクエストで次のフォームフィールドをすべて指定する必要があります。



 Note

SSE-C を使用する場合、Amazon S3 がレスポンスで返す ETag 値はオブジェクトの MD5 ではありません。

| 名前                                              | 説明                                                                                                                                                                                                                                                                                                                                                                              | 必須 |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| x-amz-server-side-encryption-customer-algorithm | <p>オブジェクトを暗号化するとき使用するアルゴリズムを指定します。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> <p>有効な値: AES256</p> <p>制約: 有効な x-amz-server-side-encryption-customer-key および x-amz-server-side-encryption-customer-key-MD5 フィールドが伴う必要があります。</p>                                                                                                                                                         | はい |
| x-amz-server-side-encryption-customer-key       | <p>Amazon S3 がデータの暗号化に使用する、お客様が用意した base64 エンコードされた暗号化キーを指定します。この値はオブジェクトの保存に使用され、その後破棄されます。Amazon では暗号化キーを保存しません。キーは、x-amz-server-side-encryption-customer-algorithm ヘッダーで指定されたアルゴリズムでの使用に適している必要があります。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> <p>制約: 有効な x-amz-server-side-encryption-customer-algorithm および x-amz-server-side-encryption-customer-key-MD5 フィールドが伴う必要があります。</p> | はい |

| 名前                                            | 説明                                                                                                                                                                                                                                                                                                                   | 必須 |
|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| x-amz-server-side-encryption-customer-key-MD5 | <p><a href="#">RFC 1321</a> に従って、暗号化キーの base64 エンコードされた 128 ビット MD5 ダイジェストを指定します。Amazon S3 では、このヘッダーを使用してメッセージの整合性を調べて、送信された暗号化キーにエラーがないことを確認します。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> <p>制約: 有効な x-amz-server-side-encryption-customer-algorithm および x-amz-server-side-encryption-customer-key フィールドが伴う必要があります。</p> | はい |

## レスポンス

### レスポンスヘッダー

このオペレーションの実装では、すべてのレスポンスに共通するレスポンスヘッダーに加え、以下のレスポンスヘッダーを含めることができます。詳細については、「[共通のレスポンスヘッダー](#)」を参照してください。

| 名前                    | 説明                                                                   |
|-----------------------|----------------------------------------------------------------------|
| x-amz-checksum-crc32  | <p>オブジェクトの base64 エンコードされた 32 ビット CRC32 チェックサム。</p> <p>タイプ: 文字列</p>  |
| x-amz-checksum-crc32c | <p>オブジェクトの base64 エンコードされた 32 ビット CRC32C チェックサム。</p> <p>タイプ: 文字列</p> |
| x-amz-checksum-sha1   | <p>オブジェクトの base64 エンコードされた 160 ビット SHA-1 ダイジェスト。</p>                 |

| 名前                                | 説明                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                   | タイプ: 文字列                                                                                                                                                                                                                                                                                                                          |
| x-amz-checksum-sha256             | オブジェクトの base64 エンコードされた 256 ビット SHA-256 ダイジェスト。<br><br>タイプ: 文字列                                                                                                                                                                                                                                                                   |
| x-amz-expiration                  | バケットのライフサイクル設定の一部としてオブジェクトに Expiration アクションが設定されている場合、Amazon S3 はこのヘッダーを返します。ヘッダー値には、expiry-date コンポーネントと URL エンコードされた rule-id コンポーネントが含まれます。バージョンが有効なバケットの場合、このヘッダーは現在のバージョンにのみ適用されます。Amazon S3 には、非現行バージョンが完全に削除可能になる時期を示すヘッダーは用意されていません。詳細については、「 <a href="#">PutBucketLifecycleConfiguration</a> 」を参照してください。<br><br>タイプ: 文字列 |
| success_action_redirect, redirect | アップロードが成功したときにクライアントがリダイレクトされる URL。<br><br>タイプ: 文字列<br><br>祖先: PostResponse                                                                                                                                                                                                                                                       |
| x-amz-server-side-encryption      | Amazon S3 へのこのオブジェクトの保存時に使用されたサーバー側の暗号化アルゴリズム (AES256、aws:kms、aws:kms:dsse など)。<br><br>タイプ: 文字列                                                                                                                                                                                                                                   |

| 名前                                              | 説明                                                                                                                                                                                           |
|-------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x-amz-server-side-encryption-aws-kms-key-id     | <p>x-amz-server-side-encryption ヘッダーの有効な値が aws:kms の場合、このヘッダーはオブジェクトの暗号化に使用された KMS キーの ID を指定します。</p> <p>タイプ: 文字列</p>                                                                        |
| x-amz-server-side-encryption-bucket-key-enabled | <p>x-amz-server-side-encryption の有効な値が aws:kms の場合、このヘッダーは、オブジェクトが S3 バケットキーを使用して SSE-KMS で暗号化されているかどうかを示します。このヘッダーが true に設定されている場合、オブジェクトは SSE-KMS に S3 バケットキーを使用します。</p> <p>タイプ: ブール値</p> |
| x-amz-server-side-encryption-customer-algorithm | <p>SSE-C がリクエストされた場合、レスポンスには、使用された暗号化アルゴリズムを確認するこのヘッダーが含まれます。</p> <p>タイプ: 文字列</p> <p>有効な値 : AES256</p>                                                                                       |
| x-amz-server-side-encryption-customer-key-MD5   | <p>SSE-C がリクエストされた場合、レスポンスには、お客様が用意した暗号化キーのラウンドトリップメッセージの整合性を検証するためのこのヘッダーが含まれます。</p> <p>タイプ: 文字列</p>                                                                                        |
| x-amz-version-id                                | <p>オブジェクトのバージョン。</p> <p>タイプ: 文字列</p>                                                                                                                                                         |

## レスポンス要素

| 名前       | 説明                                                                                                                                                                                           |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Bucket   | オブジェクトが格納されたバケットの名前。<br><br>タイプ: 文字列<br><br>祖先: PostResponse                                                                                                                                 |
| ETag     | エンティティタグ (ETag) は、GET リクエストオペレーションで If-Modified リクエストタグを使用して条件付き GET オペレーションを実行するために使用できるオブジェクトの MD5 ハッシュです。ETag は、オブジェクトの内容にのみ変更を反映し、メタデータには反映しません。<br><br>タイプ: 文字列<br><br>祖先: PostResponse |
| Key      | オブジェクトキー名。<br><br>タイプ: 文字列<br><br>祖先: PostResponse                                                                                                                                           |
| Location | オブジェクトの URI。<br><br>タイプ: 文字列<br><br>祖先: PostResponse                                                                                                                                         |

## 特殊なエラー

このオペレーションの実装では、特殊なエラーは返されません。Amazon S3 のエラーとエラーコードのリストの一般的な情報については、「[エラーレスポンス](#)」を参照してください。

## 例

### リクエスト例

```
POST /Neo HTTP/1.1
Content-Length: 4
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: authorization string
Content-Type: text/plain
Expect: the 100-continue HTTP status code
```

*ObjectContent*

### バージョニングが停止されたレスポンスの例

バケットのバージョニングが停止された場合のレスポンスの例を次に示します。

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdm0dAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiI0Ix1+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: default
Date: Wed, 12 Oct 2009 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: AmazonS3
```

このレスポンスでは、バージョン ID は null です。

### バージョニングが有効になっているレスポンスの例

バケットのバージョニングが有効になっている場合のレスポンスの例を次に示します。

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdm0dAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiI0Ix1+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: 43jfkodU8493jnFJD9fjj3HHNVfdsQUIFDNsidf038jfdsjGFDSIRp
Date: Wed, 01 Mar 2006 12:00:00 GMT
```

```
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"  
Content-Length: 0  
Connection: close  
Server: AmazonS3
```

## 関連リソース

- [CopyObject](#)
- [POST Object](#)
- [GetObject](#)

# POST Object restore

## 説明

このオペレーションは、次のタイプのリクエストを実行します。

- `select` – アーカイブされたオブジェクトに `SELECT` クエリを実行
- `restore an archive` – アーカイブされたオブジェクトを復元

このオペレーションを呼び出すには、`s3:RestoreObject` および `s3:GetObject` アクションを実行するための権限が必要です。バケット所有者はデフォルトでこのアクセス権限を持ち、このアクセス権限を他のユーザーに付与できます。アクセス権限の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[バケットサブリソースオペレーションに関連するアクセス権限](#)」および「[Amazon S3 リソースへのアクセス権限の管理](#)」を参照してください。

## Select リクエストを使用したアーカイブのクエリ

Select タイプのリクエストを使用して、アーカイブされたオブジェクトに SQL クエリを実行できます。Select リクエストによってクエリが実行されているアーカイブされたオブジェクトは、非圧縮のカンマ区切り値 (CSV) ファイルとしてフォーマットされる必要があります。よりホットな Amazon S3 層にデータを復元しなくても、アーカイブされているデータに対してクエリやカスタム分析を実行できます。Select リクエストの概要については、「Amazon Simple Storage Service ユーザーガイド」の「[アーカイブされたオブジェクトのクエリ](#)」を参照してください。

Select リクエストを行う場合は、次の操作を行います。

- `SELECT` クエリを出力するための出力場所を定義します。この場所は、クエリ対象のアーカイブ済みオブジェクトが含まれているバケットと同じ AWS リージョンにある Amazon S3 バケットであることが必要です。ジョブを開始する AWS アカウント には、S3 バケットに書き込むための許可が必要です。バケットに保存された出力オブジェクトに対して、ストレージクラスおよび暗号化を指定できます。出力の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[アーカイブされたオブジェクトのクエリ](#)」を参照してください。

リクエスト本文の S3 構造の詳細については、以下を参照してください。

- [PutObject](#)
- [ACL によるアクセス管理](#)のAmazon Simple Storage Service ユーザーガイド



- 詳細については、Amazon Simple Storage Service ユーザーガイドの「[サーバー側の暗号化を使用したデータの保護](#)」を参照してください。
- リクエスト本文の `SelectParameters` 構造で、クエリに対応する `SELECT` 型の復元について `SQL` 式を定義します。例えば、次のような式を使用できます。
- 次の式では、指定されたオブジェクトからすべてのレコードが返されます。

```
SELECT * FROM Object
```

- オブジェクトに保存されたデータのヘッダーを使用していない場合、位置ヘッダーを使用して列を指定できます。

```
SELECT s._1, s._2 FROM Object s WHERE s._3 > 100
```

- ヘッダーがあり、リクエスト本文の `CSV` 構造の `fileHeaderInfo` を `USE` に設定している場合、クエリでヘッダーを指定できます。( `fileHeaderInfo` フィールドを `IGNORE` に設定した場合は、クエリの最初の行はスキップされます。) 序数の位置とヘッダーの列名を混在させることはできません。

```
SELECT s.Id, s.FirstName, s.SSN FROM S3Object s
```

Amazon Glacier Select の復元で `SQL` を使用方法の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[Amazon S3 Select および Amazon Glacier Select の SQL リファレンス](#)」を参照してください。

Select リクエストを行う場合は、次の操作も実行できます。

- クエリを迅速に処理するには、`Expedited` 階層を指定します。階層の詳細については、このトピックの後半にある「アーカイブの復元」を参照してください。
- クエリ中の入力オブジェクト、および `CSV` でエンコードされたクエリ結果の両方について、データシリアル化形式に関する詳細を指定します。

以下は、Select 機能に関するその他の重要な事実です。

- 出力結果は新しい Amazon S3 オブジェクトです。アーカイブの取得とは異なり、手動またはライフサイクルポリシーによって明示的に削除されるまで保存されます。
- 同じ Amazon S3 オブジェクトに対して複数の Select リクエストを発行できます。Amazon S3 はリクエストの重複を排除しないため、重複するリクエストを発行しないでください。

- Amazon S3 は、オブジェクトが既に復元されている場合でも、Select リクエストを受け入れます。Select リクエストはエラーレスポンス 409 を返しません。

## アーカイブの復元

GLACIER および DEEP\_ARCHIVE ストレージクラスのオブジェクトはアーカイブされています。アーカイブされたオブジェクトにアクセスするには、まず復元リクエストを開始する必要があります。これにより、アーカイブされたオブジェクトの一時コピーが復元されます。復元リクエストでは、復元されたコピーを保存しておく日数を指定します。指定した期間を過ぎると、Amazon S3 は一時コピーを削除しますが、オブジェクトはオブジェクトが復元された GLACIER または DEEP\_ARCHIVE ストレージクラスにアーカイブされたままになります。

特定のオブジェクトバージョンを復元するには、バージョン ID を指定できます。バージョン ID を指定しない場合、Amazon S3 は現在のバージョンを復元します。

復元ジョブが完了するまでにかかる時間は、オブジェクトを復元するストレージクラスと、指定したデータアクセス階層によって異なります。

アーカイブされたオブジェクトを復元する場合 (または Select リクエストを使用する場合)、リクエスト本文の Tier 要素で次のいずれかのデータアクセス階層オプションを指定できます。

- **Expedited** - 迅速取り出しでは、アーカイブのサブセットが迅速に必要なときに GLACIER ストレージクラスに保存されているデータにすばやくアクセスできます。最大規模のアーカイブオブジェクト (250 MB 以上) を除くすべてのアーカイブについては、迅速取り出しでアクセスしたデータは通常 1〜5 分以内で使用可能になります。プロビジョンドキャパシティーは、迅速取り出しの取得容量を必要なときに利用できることを保証します。迅速取り出しとプロビジョンドキャパシティーは、DEEP\_ARCHIVE ストレージクラスでは利用できません。
- **Standard** - 標準取り出しでは、数時間以内にすべてのアーカイブオブジェクトにアクセスできます。標準取り出しは、取り出しオプションを指定しない GLACIER および DEEP\_ARCHIVE 取り出しリクエストのデフォルトのオプションです。標準取り出しは通常、GLACIER ストレージクラスの場合は 3〜5 時間以内に完了し、DEEP\_ARCHIVE ストレージクラスの場合は 12 時間以内に完了します。
- **Bulk** - 大容量取り出しは、Amazon Glacier の最も安価な取り出しオプションで、1 日で大量のデータ (ペタバイトのデータを含む) を安価に取得できます。大容量取り出しは通常、GLACIER ストレージクラスの場合は 5〜12 時間以内に完了し、DEEP\_ARCHIVE ストレージクラスの場合は 48 時間以内に完了します。

Expedited データアクセス用のアーカイブ取り出しオプションおよびプロビジョニングされた容量の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[アーカイブされたオブジェクトの復元](#)」を参照してください。

Amazon S3 の復元速度のアップグレードを使用すると、復元進行中に復元速度をより速く変更できます。進行中の復元の速度をアップグレードするには、同じオブジェクトに対して別の復元リクエストを発行し、新しい Tier リクエスト要素を設定します。復元階層をアップグレードするリクエストを発行するときは、進行中の復元が使用している層よりも速い層を選択する必要があります。Days リクエスト要素などの他のパラメータは変更しないでください。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[進行中の復元速度のアップグレード](#)」を参照してください。

オブジェクトの復元のステータスを取得するには、HEAD リクエストを送信します。オペレーションは x-amz-restore ヘッダーを返します。これは、レスポンスで復元ステータスに関する情報を提供します。Amazon S3 イベント通知を使用して、復元が開始または完了したときに通知できます。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[Amazon S3 イベント通知の設定](#)」を参照してください。

アーカイブされたオブジェクトを復元した後、新しい期間でリクエストを再発行することで復元期間を更新できます。Amazon S3 は、現在の時刻を基準に復元期間を更新し、リクエストに対してのみ課金します。データ転送料金は発生しません。Amazon S3 がオブジェクトの現在の復元リクエストをアクティブに処理している場合、復元期間を更新することはできません。

バケットに有効期限アクションを含むルールを持つライフサイクル設定がある場合、オブジェクトの有効期限は復元リクエストで指定した有効期間を上書きします。例えば、オブジェクトのコピーを 10 日間復元しても、オブジェクトの有効期限が 3 日間に予定されている場合、Amazon S3 はオブジェクトを 3 日間で削除します。ライフサイクル設定の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[PutBucketLifecycleConfiguration](#)」と「[オブジェクトのライフサイクルの管理](#)」を参照してください。

## リクエスト

### 構文

```
POST /ObjectName?restore&versionId=VersionID HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see ##### \(AWS ##### 4\))
Content-MD5: MD5
```

*request body*

 Note

構文には、リクエストヘッダーの一部が表示されます。詳細なリストについては、このトピックの後半にある「リクエストヘッダー」を参照してください。

## リクエストパラメータ

オペレーションの実装では、リクエストパラメータを使用しません。

## リクエストヘッダー

| 名前          | 説明                                                                                                                                                                                      | 必須 |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Content-MD5 | <p>データの base64 エンコードされた 128 ビット MD5 ダイジェスト。このヘッダーをメッセージ整合性チェックとして使用して、リクエスト本文が転送中に破損していないことを確認する必要があります。詳細については、<a href="#">RFC 1864</a>を参照してください。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> | はい |

## リクエストの要素

以下は、アーカイブを復元するためのリクエスト本文の XML の例です。

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Bulk</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

次の表は、リクエスト本文のアーカイブ復元の XML について説明しています。

| 名前                   | 説明                                                                                                                                                                                                                                    | 必須              |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| RestoreRequest       | 復元情報のコンテナ。<br><br>型: コンテナ                                                                                                                                                                                                             | はい              |
| Days                 | 復元された (アクティブ) コピーの有効期間。Amazon Glacier からオブジェクトを復元できる最小日数は 1 です。オブジェクトコピーが指定された有効期間に達すると、Amazon S3 はバケットからオブジェクトコピーを削除します。アーカイブを復元する場合は、この要素が必要です。<br><br>SELECT タイプのリクエストでこの要素を使用しないでください。<br><br>タイプ: 正の整数<br><br>祖先: RestoreRequest | はい、アーカイブを復元する場合 |
| GlacierJobParameters | S3 Glacier ストレージクラスのジョブパラメータのコンテナ。<br><br>SELECT タイプのリクエストでこの要素を使用しないでください。<br><br>型: コンテナ<br><br>祖先: RestoreRequest                                                                                                                  | いいえ             |
| Tier                 | アーカイブを復元するときに使用するデータアクセス階層。Standard がデフォルトです。<br><br>タイプ: Enum<br><br>有効な値: Expedited   Standard   Bulk<br><br>祖先: GlacierJobParameters                                                                                               | いいえ             |

次の XML は、アーカイブされたオブジェクトに対する Select クエリのリクエスト本文です。

```
<RestoreRequest>
  <Type>SELECT</Type>
  <Tier>Expedited</Tier>
  <Description>Job description</Description>
  <SelectParameters>
    <Expression>Select * from Object</Expression>
    <ExpressionType>SQL</ExpressionType>
    <InputSerialization>
      <CSV>
        <FileHeaderInfo>IGNORE</FileHeaderInfo>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <Comments>#</Comments>
      </CSV>
    </InputSerialization>
    <OutputSerialization>
      <CSV>
        <QuoteFields>ASNEEDED</QuoteFields>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
      </CSV>
    </OutputSerialization>
  </SelectParameters>
  <OutputLocation>
    <S3>
      <BucketName>Name of bucket</BucketName>
      <Prefix>Key prefix</Prefix>
      <CannedACL>Canned ACL string</CannedACL>
      <AccessControlList>
        <Grantee>
          <Type>Grantee Type</Type>
        </Grantee>
      <ID>Grantee identifier</ID>
      <URI>Grantee URI</URI>
      <Permission>Granted permission</Permission>
      <DisplayNmae>Display Name</DisplayName>
      <EmailAddress>email</EmailAddress>
    </Grantee>
  </AccessControlList>
</OutputLocation>
</S3>
</BucketName>
</Prefix>
</CannedACL>
</AccessControlList>
</Grantee>
</ID>
</URI>
</Permission>
</DisplayNmae>
</EmailAddress>
</AccessControlList>
</S3>
</OutputLocation>
</SelectParameters>
</Description>
</Tier>
</Type>
</RestoreRequest>
```

```
<Encryption>
  <EncryptionType>Encryption type</EncryptionType>
<KMSKeyId>KMS Key ID</KMSKeyId>
<KMSContext>Base64-encoded JSON<KMSContext>
  </Encryption>
<UserMetadata>
  <MetadataEntry>
    <Name>Key</Name>
    <Value>Value</Value>
  </MetadataEntry>
</UserMetadata>
<Tagging>
  <TagSet>
    <Tag>
      <Key>Tag name</Key>
      <Value>Tag value</Value>
    </Tag>
  </TagSet>
</Tagging>
<StorageClass>Storage class</StorageClass>
</S3>
</OutputLocation>
</RestoreRequest>
```

次の表は、リクエスト本文における SELECT タイプの復元の XML について説明しています。

| 名前             | 説明                                                                                                                                    | 必須  |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------|-----|
| RestoreRequest | 復元情報のコンテナ。<br><br>型: コンテナ                                                                                                             | はい  |
| Tier           | アーカイブを復元するときに使用するデータアクセス階層。<br>Standard がデフォルトです。<br><br>タイプ: Enum<br><br>有効な値: Expedited   Standard   Bulk<br><br>祖先: RestoreRequest | いいえ |

| 名前               | 説明                                                                                       | 必須                      |
|------------------|------------------------------------------------------------------------------------------|-------------------------|
| Description      | 必要に応じて、リクエストの説明。<br><br>タイプ: 文字列<br><br>祖先: RestoreRequest                               | いいえ                     |
| SelectParameters | Select ジョブリクエストのパラメータについて記述します。<br><br>型: コンテナ<br><br>祖先: RestoreRequest                 | はい。リクエストタイプが SELECT の場合 |
| OutputLocation   | Select 復元リクエストの結果を受け取る場所について記述します。<br><br>タイプ: Amazon S3 用コンテナ<br><br>祖先: RestoreRequest | はい。リクエストタイプが SELECT の場合 |

**SelectParameters** コンテナ要素には、以下の要素が含まれます。

| 名前         | 説明                                                                                                                                                                                                                        | 必須 |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Expression | SQL 式。例えば、次のようになります。 <ul style="list-style-type: none"><li>次の SQL 式は、CSV 形式で保存されているオブジェクトからデータの最初の列を取得します。<br/><br/>SELECT s._1 FROM Object s</li><li>次の SQL 式は、オブジェクトからすべてを返します。<br/><br/>SELECT * FROM Object</li></ul> | はい |



| 名前                  | 説明                                                                              | 必須 |
|---------------------|---------------------------------------------------------------------------------|----|
|                     | タイプ: 文字列<br><br>祖先: SelectParameters                                            |    |
| ExpressionType      | 式タイプを識別します。<br><br>型: 文字列<br><br>有効な値: SQL<br><br>祖先: SelectParameters          | はい |
| InputSerialization  | オブジェクトのシリアル化形式について記述します。<br><br>タイプ: CSV 用コンテナ<br><br>祖先: SelectParameters      | はい |
| OutputSerialization | 選択ジョブの結果がシリアル化される方法について記述します。<br><br>タイプ: CSV 用コンテナ<br><br>祖先: SelectParameters | はい |

**InputSerialization** 要素の CSV コンテナ要素には、次の要素が含まれます。

| 名前              | 説明                                                                                        | 必須  |
|-----------------|-------------------------------------------------------------------------------------------|-----|
| RecordDelimiter | 入力で個々のレコードを区切るために使用される 1 文字。デフォルト値の代わりに、任意の区切り文字を指定できます。<br><br>タイプ: 文字列<br><br>デフォルト: \n | いいえ |

| 名前             | 説明                                                                                                                                                                                                                                      | 必須  |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|                | 祖先: CSV                                                                                                                                                                                                                                 |     |
| FieldDelimiter | <p>レコード内で個々のフィールドを区切るために使用される 1 文字。任意の区切り文字を指定できます。</p> <p>タイプ: 文字列</p> <p>デフォルト: ,</p> <p>祖先: CSV</p>                                                                                                                                  | いいえ |
| QuoteCharacter | <p>フィールド区切り文字が値の一部である場合にエスケープに使用される 1 文字。</p> <p>CSV ファイルで次の例を考えてみましょう。</p> <p>"a, b"</p> <p>値を引用符で囲むと、この値は 1 つのフィールドになります。引用符を使用しない場合、カンマはフィールド区切り文字です (2 つの個別のフィールド値、a と b になります)。</p> <p>タイプ: 文字列</p> <p>デフォルト: "</p> <p>祖先: CSV</p> | いいえ |

| 名前                   | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                            | 必須  |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| QuoteEscapeCharacter | <p>既にエスケープされた値内で引用符文字をエスケープするために使用される 1 文字。例えば、値 <code>"" a , b ""</code> は <code>" a , b "</code> として解析されます。</p> <p>タイプ: 文字列</p> <p>デフォルト: <code>"</code></p> <p>祖先: CSV</p>                                                                                                                                                                                                                                                                 | いいえ |
| FileHeaderInfo       | <p>入力データの最初の行について記述します。これは ENUM 値の 1 つです。</p> <ul style="list-style-type: none"><li>NONE: 最初の行はヘッダーではありません。</li><li>IGNORE: 1 行目はヘッダーですが、ヘッダー値を使用して式内の列を示すことはできません。列の位置 (<code>_1</code>、<code>_2</code>、... など) を使用して列 (<code>SELECT s._1 FROM OBJECT s</code>) を指定できます。</li><li>Use: 1 行目はヘッダーです。ヘッダー値を使用して式内の列を識別できます (<code>SELECT "name" FROM OBJECT</code>)。</li></ul> <p>タイプ: Enum</p> <p>有効な値: NONE   USE   IGNORE</p> <p>祖先: CSV</p> | いいえ |

| 名前       | 説明                                                                                                        | 必須  |
|----------|-----------------------------------------------------------------------------------------------------------|-----|
| Comments | <p>この文字が行の冒頭にある場合、その行を無視する必要があることを示すのに使用される 1 文字。コメント行を示す任意の文字を指定できます。</p> <p>タイプ: 文字列</p> <p>祖先: CSV</p> | いいえ |

CSV コンテナ要素 (**OutputSerialization** 要素内) には、次の要素が含まれています。

| 名前              | 説明                                                                                                                                                                                                                                                      | 必須  |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| QuoteFields     | <p>出力フィールドの周囲に引用符を使用するかどうかを示します。</p> <ul style="list-style-type: none"><li>• ALWAYS: 出力フィールドには必ず引用符を使用します。</li><li>• ASNEEDED: 必要に応じて出力フィールドに引用符を使用します。</li></ul> <p>タイプ: Enum</p> <p>有効な値: ALWAYS   ASNEEDED</p> <p>デフォルト: AsNeeded</p> <p>祖先: CSV</p> | いいえ |
| RecordDelimiter | <p>出力で個々のレコードを区切るために使用される 1 文字。デフォルト値の代わりに、任意の区切り文字を指定できます。</p> <p>タイプ: 文字列</p> <p>デフォルト: \n</p>                                                                                                                                                        | いいえ |

| 名前                   | 説明                                                                                                                                                        | 必須  |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|                      | 祖先: CSV                                                                                                                                                   |     |
| FieldDelimiter       | <p>レコード内で個々のフィールドを区切るために使用される 1 文字。任意の区切り文字を指定できます。</p> <p>タイプ: 文字列</p> <p>デフォルト: ,</p> <p>祖先: CSV</p>                                                    | いいえ |
| QuoteCharacter       | <p>フィールド区切り文字が値の一部である場合にエスケープに使用される 1 文字。例えば、値が a, b の場合、Amazon S3 は " a , b " のように、このフィールド値を引用符で囲みます。</p> <p>タイプ: 文字列</p> <p>デフォルト: "</p> <p>祖先: CSV</p> | いいえ |
| QuoteEscapeCharacter | <p>既にエスケープされた値内で引用符文字をエスケープするために使用される 1 文字。例えば、値が " a , b " の場合、Amazon S3 は "" " a , b "" のように、値を引用符で囲みます。</p> <p>タイプ: 文字列</p> <p>祖先: CSV</p>             | いいえ |

S3 コンテナ要素 (**OutputLocation** 要素内) には、次の要素が含まれています。

| 名前                | 説明                                                                                                                                                                                                                     | 必須  |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| AccessControlList | <p>ステージングされた結果へのアクセスを制御する許可のリスト。</p> <p>タイプ: Grant 用コンテナ</p> <p>祖先: S3</p>                                                                                                                                             | いいえ |
| BucketName        | <p>Select 復元結果が保存されている S3 バケットの名前。バケットは、入力アーカイブオブジェクトを含むバケットと同じ AWS リージョンに存在する必要があります。</p> <p>タイプ: 文字列</p> <p>祖先: S3</p>                                                                                               | はい  |
| CannedACL         | <p>Select 復元結果に適用する、既定アクセスコントロールリスト (ACL)。</p> <p>型: 文字列</p> <p>有効な値: private   public-read   public-read-write   aws-exec-read   authenticated-read   bucket-owner-read   bucket-owner-full-control</p> <p>祖先: S3</p> | いいえ |
| Encryption        | <p>保存された結果の暗号化情報が含まれます。</p> <p>タイプ: 暗号化用コンテナ</p> <p>祖先: S3</p>                                                                                                                                                         | いいえ |
| Prefix            |                                                                                                                                                                                                                        | はい  |

| 名前           | 説明                                                                                                                                     | 必須  |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------|-----|
|              | Select 復元結果に付加されるプレフィックス。プレフィックスの最大長は、512 バイトです。<br><br>タイプ: 文字列<br><br>祖先: S3                                                         |     |
| StorageClass | Select 復元結果を保存するために使用されるストレージのクラス。<br><br>タイプ: 文字列<br><br>有効な値: STANDARD   REDUCED_REDUNDANCY   STANDARD_IA   ONEZONE_IA<br><br>祖先: S3 | いいえ |
| Tagging      | タグ情報のコンテナ。<br><br>タイプ: タグ構造<br><br>祖先: S3                                                                                              | いいえ |
| UserMetadata | Select 復元結果とともに保存するメタデータのリストが含まれます。<br><br>タイプ: MetadataEntry 構造<br><br>祖先: S3                                                         | いいえ |

Grantee コンテナ要素 (**AccessControlList** 要素内) には、次の要素が含まれています。

| 名前          | 説明             | 必須  |
|-------------|----------------|-----|
| DisplayName | 被付与者のスクリーンネーム。 | いいえ |

| 名前           | 説明                                                  | 必須  |
|--------------|-----------------------------------------------------|-----|
|              | タイプ: 文字列<br><br>祖先: Grantee                         |     |
| EmailAddress | 被付与者の E メールアドレス。<br><br>タイプ: 文字列<br><br>祖先: Grantee | いいえ |
| ID           | 被付与者の正規ユーザー ID。<br><br>タイプ: 文字列<br><br>祖先: Grantee  | いいえ |
| Type         | 被付与者のタイプ。<br><br>タイプ: 文字列<br><br>祖先: Grantee        | いいえ |
| URI          | 被付与者グループの URI。<br><br>タイプ: 文字列<br><br>祖先: Grantee   | いいえ |
| Permission   | アクセス許可を付与します。<br><br>タイプ: 文字列<br><br>祖先: Grantee    | いいえ |



Encryption コンテナ要素 (S3 内) には、次の要素が含まれています。

| 名前             | 説明                                                                                                                        | 必須  |
|----------------|---------------------------------------------------------------------------------------------------------------------------|-----|
| EncryptionType | <p>ジョブ結果の保存時に使用されるサーバー側の暗号化アルゴリズム。デフォルトでは暗号化しません。</p> <p>タイプ: 文字列</p> <p>有効な値: aws:kms   AES256</p> <p>祖先: Encryption</p> | いいえ |
| KMSContext     | <p>オプション。暗号化タイプが aws:kms である場合、この値を使用して Select 復元結果の暗号化コンテキストを指定できます。</p> <p>タイプ: 文字列</p> <p>祖先: Encryption</p>           | いいえ |
| KMSKeyId       | <p>オブジェクトの暗号化に使用する AWS Key Management Service (AWS KMS) キー ID。</p> <p>タイプ: 文字列</p> <p>祖先: Encryption</p>                  | いいえ |

TagSet コンテナ要素 (Tagging 要素内) には、次の要素が含まれています。

| 名前  | 説明                                                  | 必須  |
|-----|-----------------------------------------------------|-----|
| Tag | <p>タグが含まれています。</p> <p>型: コンテナ</p> <p>祖先: TagSet</p> | いいえ |

タグコンテナ要素 (**TagSet** 要素内) には、次の要素が含まれています。

| 名前    | 説明                                    | 必須  |
|-------|---------------------------------------|-----|
| Key   | タグの名前。<br><br>タイプ: 文字列<br><br>祖先: Tag | いいえ |
| Value | タグの値。<br><br>タイプ: 文字列<br><br>祖先: Tag  | いいえ |

**MetadataEntry** コンテナ要素 (**UserMetadata** 要素内) には、オブジェクトに保存する以下のキーと値のペア要素が含まれています。

| 名前            | 説明                                  | 必須  |
|---------------|-------------------------------------|-----|
| MetadataKey   | メタデータキー。<br><br>タイプ: 文字列<br><br>祖先: | いいえ |
| MetadataEntry | メタデータ値。<br><br>タイプ: 文字列<br><br>祖先:  | いいえ |

## レスポンス

オペレーションが成功すると、200 OK または 202 Accepted ステータスコードが返されます。

- オブジェクトのコピーが以前に復元されていない場合、Amazon S3 はレスポンスで 202 Accepted を返します。
- オブジェクトのコピーが以前に復元されている場合、Amazon S3 はレスポンスで 200 OK を返します。

## レスポンスヘッダー

このオペレーションの実装では、ほとんどのレスポンスに共通のレスポンスヘッダーのみを使用します。詳細については、「[共通のレスポンスヘッダー](#)」を参照してください。

## レスポンス要素

このオペレーションは、レスポンス要素は返しません。

## 特殊なエラー

| エラーコード                                | 説明                                                                                                                                                    | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------------------|
| RestoreAlreadyInProgress              | オブジェクトの復元は既に進行中です。(このエラーは SELECT タイプのリクエストには適用されません。)                                                                                                 | 409 Conflict  | クライアント             |
| GlacierExpeditedRetrievalNotAvailable | S3 Glacier ストレージクラスの迅速取り出しは現在利用できません。後でもう一度お試しください。(Expedited リクエストを処理するのに十分な容量がない場合に返されます。このエラーは Expedited 取得にのみ適用され、Standard または Bulk 取得には適用されません。) | 503           | 該当なし               |

## 例

### 迅速取得オプションを使用してオブジェクトを 2 日間復元する

次の復元リクエストでは、迅速取り出しオプションを使用して、Amazon Glacier から photo1.jpg オブジェクトのコピーを 2 日間復元します。

```
POST /photo1.jpg?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Mon, 22 Oct 2012 01:49:52 GMT
Authorization: authorization string
Content-Length: content length
```

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Expedited</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

examplebucket にオブジェクトの復元されたコピーがない場合、Amazon S3 は次の 202 Accepted レスポンスを返します。

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZ1zYQvPiBlZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

オブジェクトのコピーが既に復元されている場合、Amazon S3 は 200 OK レスポンスを返し、復元されたコピーの有効期限のみを更新します。

### SELECT リクエストを使用してアーカイブをクエリする

Select 復元リクエストの例を次に示します。

```
POST /object-one.csv?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Sat, 20 Oct 2012 23:54:05 GMT
Authorization: authorization string
```

Content-Length: *content length*

```
<RestoreRequest xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Type>SELECT</Type>
  <Tier>Expedited</Tier>
  <Description>this is a description</Description>
  <SelectParameters>
    <InputSerialization>
      <CSV>
        <FileHeaderInfo>IGNORE</FileHeaderInfo>
        <Comments>#</Comments>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
      </CSV>
    </InputSerialization>
    <ExpressionType>SQL</ExpressionType>
    <Expression>select * from object</Expression>
    <OutputSerialization>
      <CSV>
        <QuoteFields>ALWAYS</QuoteFields>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>\t</FieldDelimiter>
        <QuoteCharacter>\'</QuoteCharacter>
      </CSV>
    </OutputSerialization>
  </SelectParameters>
  <OutputLocation>
    <S3>
      <BucketName>example-output-bucket</BucketName>
      <Prefix>test-s3</Prefix>
      <AccessControlList>
        <Grant>
          <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail">
            <EmailAddress>jane-doe@example.com</EmailAddress>
          </Grantee>
          <Permission>FULL_CONTROL</Permission>
        </Grant>
      </AccessControlList>
      <UserMetadata>
        <MetadataEntry>
```

```
<Name>test</Name>
<Value>test-value</Value>
</MetadataEntry>
<MetadataEntry>
  <Name>other</Name>
  <Value>something else</Value>
</MetadataEntry>
</UserMetadata>
<StorageClass>STANDARD</StorageClass>
</S3>
</OutputLocation>
</RestoreRequest>
```

Amazon S3 は以下の 202 Accepted レスポンスを返します。

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZ1zYQvPiB1ZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
x-amz-restore-output-path: js-test-s3/qE8nk5M0XIj-LuZE2HXNw6empQm3znLkH1MWInRYPS-
0rl2W0uj6LyYm-neTvm1-btz3wbBxfMhPykd3jk1-lvZE7w42/
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

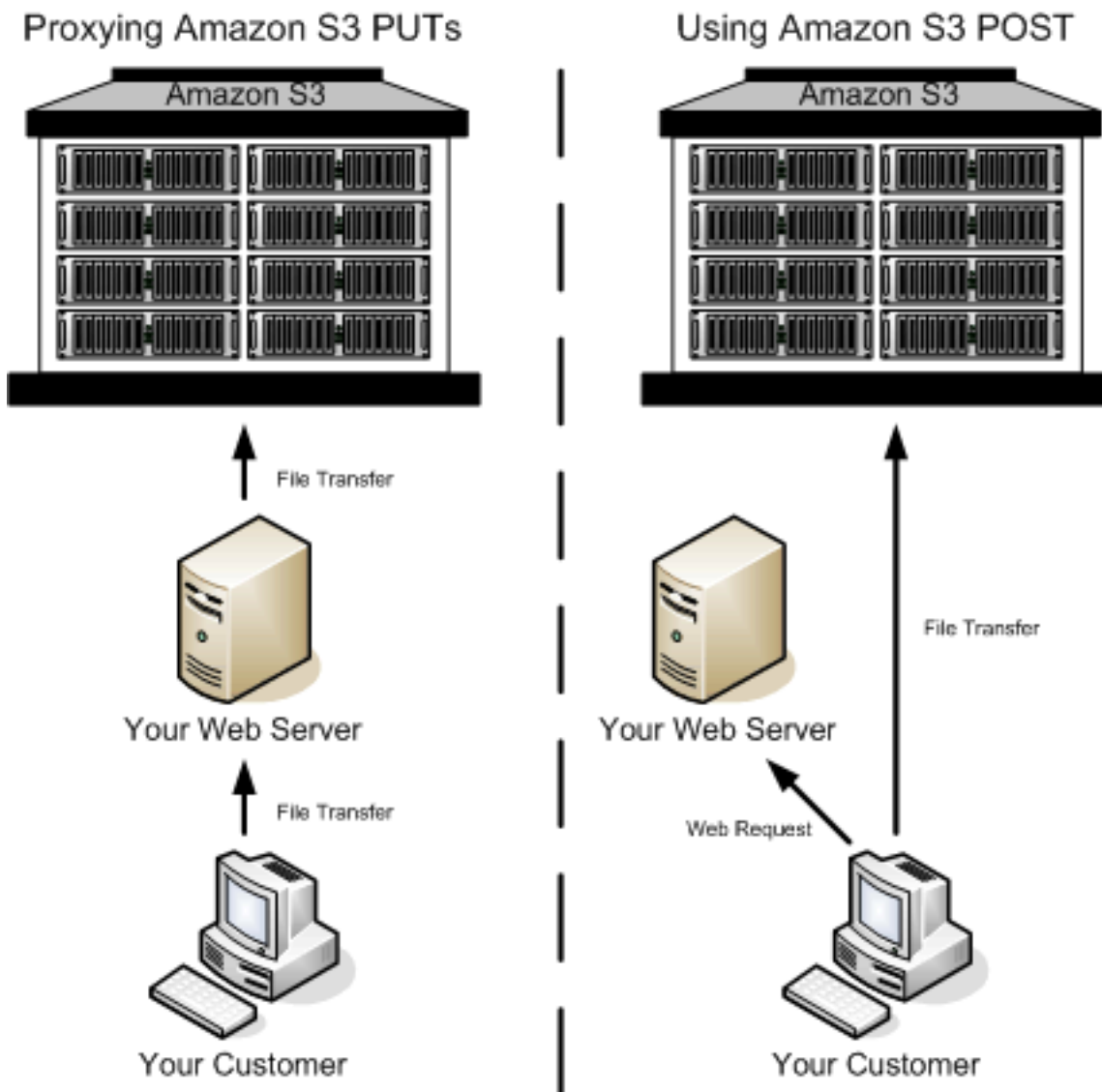
## 詳細

- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)
- [Amazon S3 Select および Amazon Glacier Select の SQL リファレンス](#) (Amazon Simple Storage Service ユーザーガイド)

## HTTP POST を使用したブラウザベースのアップロード

Amazon S3 は HTTP POST リクエストをサポートしているため、ユーザーはコンテンツを Amazon S3 に直接アップロードできます。POST を使用すると、エンドユーザーは認証情報を保護する安全な中間ノードを介してデータを渡すことなく、リクエストを認証できます。したがって、HTTP POST はレイテンシーを低減する可能性があります。

次の図は、Amazon S3 の POST リクエストを使用したアップロードを示しています。



1. ユーザーはウェブブラウザからページにアクセスします。
2. ウェブページには、ユーザーがコンテンツを Amazon S3 にアップロードするために必要なすべての情報が含まれている HTML フォームがあります。
3. ユーザーはウェブブラウザを介して Amazon S3 にコンテンツをアップロードします。

ブラウザベースの POST リクエストを送信するプロセスは次のとおりです。

1. オブジェクトをアップロードできるバケット名や、作成中のオブジェクトに許可するキー名のプレフィックスなど、リクエストで許可する条件を指定するセキュリティポリシーを作成します。
2. ポリシーに基づく署名を作成します。認証されたリクエストの場合、フォームには有効な署名とポリシーが含まれている必要があります。

3. ユーザーが Amazon S3 バケットにオブジェクトをアップロードするためにアクセスできる HTML フォームを作成します。

次のセクションでは、リクエストを認証する署名を作成する方法について説明します。フォームの作成とセキュリティポリシーの詳細については、「[HTML フォームの作成 \(AWS 署名バージョン 4 を使用\)](#)」を参照してください。

## 署名の計算

認証されたリクエストの場合、HTML フォームにはセキュリティポリシーと署名のためのフィールドを含める必要があります。

- セキュリティポリシー (「[POST ポリシー](#)」を参照) は、リクエストで許可される内容を制御します。
- セキュリティポリシーは、署名計算の StringToSign (「[リクエストへの署名の概要](#)」を参照) です。





## 署名を計算するには

1. UTF-8 エンコーディングを使用してポリシーを作成します。
2. UTF-8 エンコードされたポリシーのバイトを base64 に変換します。結果は、StringToSign です。
3. 署名キーを作成します。
4. 署名キーを使用して、HMAC-SHA256 署名アルゴリズムを使用して StringToSign に署名します。

HTML フォーム、セキュリティポリシー、および例の作成の詳細については、以下を参照してください。

- [HTML フォームの作成 \(AWS 署名バージョン 4 を使用\)](#)
- [POST ポリシー](#)
- [例: HTTP POST \(AWS 署名バージョン 4\) を使用したブラウザベースのアップロード](#)

## HTML フォームの作成 (AWS 署名バージョン 4 を使用)

### トピック

- [HTML フォーム宣言](#)
- [HTML フォームフィールド](#)

ユーザーがブラウザ (HTTP POST リクエスト) を使用してコンテンツを Amazon S3 にアップロードできるようにするには、HTML フォームを使用します。HTML フォームは、フォーム宣言とフォームフィールドで構成されます。フォーム宣言には、リクエストの概要情報が含まれています。フォームフィールドには、詳細なリクエスト情報が含まれています。

このセクションでは、HTML フォームの作成方法について説明します。HTTP POST を使用したブラウザベースのアップロードの実例と、リクエスト認証に関連する署名計算については、「[例: HTTP POST \(AWS 署名バージョン 4\) を使用したブラウザベースのアップロード](#)」を参照してください。

フォームとポリシーは UTF-8 エンコーディングされている必要があります。content 属性に charset=UTF-8 を指定することで、フォームに UTF-8 エンコーディングを適用できます。HTML 見出しの UTF-8 エンコードの例を次に示します。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

リクエストヘッダーの UTF-8 エンコードの例を次に示します。

```
Content-Type: text/html; charset=UTF-8
```

#### Note

フォームデータと境界 (ファイルのコンテンツは除く) は 20 KB を超えることはできません。

## HTML フォーム宣言

HTML フォーム宣言には、次の 3 つの属性があります。

- **action** – リクエストを処理する URL を指定します。これはバケットの URL に設定されていなければなりません。例えば、バケットの名前が `examplebucket` の場合、URL は `http://examplebucket.s3.amazonaws.com/` です。

#### Note

キー名はフォームフィールドで指定されます。

- **method** – メソッドは POST である必要があります。
- **enctype** – エンクロージャタイプ (enctype) は、ファイルアップロードとテキストエリアアップロードの両方に対して、`multipart/form-data` が設定されていなければなりません。enctype の詳細については、[RFC 1867](https://tools.ietf.org/html/rfc1867) を参照してください。

これは、バケット `examplebucket` のフォーム宣言です。

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post">
```

```
enctype="multipart/form-data">
```

## HTML フォームフィールド

以下の表に、フォーム内で使用できるフィールドのリストを示します。特に、リクエストの認証に使用できる署名フィールドがあります。署名計算アルゴリズム (x-amz-algorithm)、署名キーの生成に使用した認証情報スコープ (x-amz-credential)、および署名の計算に使用した日付 (x-amz-date) を指定するフィールドがあります。Amazon S3 はこの情報を使用して署名を再作成します。署名が一致すると、Amazon S3 はリクエストを処理します。

### Note

変数 `${filename}` は、ユーザーによって指定されたファイル名に自動的に置き換えられ、すべてのフォームフィールドで認識されます。ブラウザまたはクライアントがファイルへの完全または部分的なパスを提供した場合、最後のスラッシュ (/) またはバックスラッシュ (\) 以降のテキストのみが使用されます (例えば、`C:\Program Files\directory1\file.txt` は `file.txt` として解釈されます)。ファイルまたはファイル名が指定されていない場合、変数は空の文字列に置き換えられます。

policy 要素など、認証されたリクエストに必要な要素を指定しない場合、リクエストは匿名と見なされ、パブリック読み取りおよび書き込み用にバケットを設定している場合にのみ成功します。

| 要素名 | 説明                                                                                                                                                                                                                                                 | 必須  |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| acl | <p>Amazon S3 のアクセスコントロールリスト (ACL)。無効な ACL が指定されている場合、Amazon S3 はリクエストを拒否します。ACL の詳細については、「<a href="#">Amazon S3 ACL の使用</a>」を参照してください。</p> <p>タイプ: 文字列</p> <p>デフォルト: プライベート</p> <p>有効な値: private   public-read   public-read-write   aws-exec-</p> | いいえ |

| 要素名                                                                                 | 説明                                                                                                                                                                                                                                                                                                                                                      | 必須  |
|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|                                                                                     | read   authenticated-read   bucket-owner-read   bucket-owner-full-control                                                                                                                                                                                                                                                                               |     |
| Cache-Control<br>Content-Type<br>Content-Disposition<br>Content-Encoding<br>Expires | REST 固有ヘッダー。詳細については、「 <a href="#">PutObject</a> 」を参照してください。                                                                                                                                                                                                                                                                                             | いいえ |
| key                                                                                 | <p>アップロードされたオブジェクトのキーの名前。</p> <p>ユーザーによって指定されたファイル名を使用するには、<code>\${filename}</code> 変数を使用します。例えば、ファイル <code>photo1.jpg</code> をアップロードし、キー名として <code>/user/user1/\${filename}</code> を指定した場合、ファイルは <code>/user/user1/photo1.jpg</code> として保存されます。</p> <p>詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">オブジェクトメタデータの使用</a>」を参照してください。</p> | はい  |

| 要素名                     | 説明                                                                                                                                                                                                                                                   | 必須            |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| policy                  | <p>リクエストで許可されているものについて説明する base64 でエンコードされたセキュリティポリシー。認証されたリクエストには、ポリシーが必要です。</p> <p>セキュリティポリシーがないリクエストは匿名と見なされ、誰でも書き込むことができるバケットでのみ動作します。</p>                                                                                                       | 認証されたリクエストに必須 |
| success_action_redirect | <p>アップロードが成功したときにクライアントがリダイレクトされる URL。</p> <p>success_action_redirect が指定されていない場合、または Amazon S3 が URL を解釈できない場合、Amazon S3 は success_action_status フィールドで指定されている空のドキュメントタイプを返します。</p> <p>アップロードが失敗した場合、Amazon S3 はエラーを返し、ユーザーを別の URL にリダイレクトしません。</p> | いいえ           |

| 要素名                   | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 必須  |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| success_action_status | <p>アップロードが正常に行われたとき、success_action_redirect が指定されていない場合にクライアントに返されるステータスコード。</p> <p>有効な値は、200、201、または 204 (デフォルト) です。</p> <p>値が 200 または 204 に設定されている場合、Amazon S3 は空のドキュメントとステータスコードを返します。</p> <p>値が 201 に設定されている場合、Amazon S3 は XML ドキュメントとステータスコード 201 を返します。XML ドキュメントの内容については、「<a href="#">POST Object</a>」を参照してください。</p> <p>値が設定されていないか無効である場合は、Amazon S3 は空のドキュメントとステータスコード 204 を返します。</p> <div><p> <b>Note</b></p><p>Adobe Flash Player のバージョンによっては、本文が空の HTTP レスポンスが適切に処理されないことがあります。Adobe Flash でアップロードをサポートするには、success_action_status を 201 に設定することをお勧めします。</p></div> | いいえ |

| 要素名              | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 必須            |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| x-amz-algorithm  | <p>リクエストの認証に使用される署名アルゴリズム。AWS 署名バージョン 4 の場合、値は AWS4-HMAC-SHA256 です。</p> <p>このフィールドは、ポリシードキュメントがリクエストに含まれる場合に必要です。</p>                                                                                                                                                                                                                                                                                                                                                                                         | 認証されたリクエストに必須 |
| x-amz-credential | <p>このフィールドは、アクセスキー ID の他に、署名が有効なリージョンおよびサービスを特定するスコープも指定します。これは、署名計算の署名キーの計算に使用したスコープと同じである必要があります。</p> <p>これは、次のような形式の文字列です。</p> <pre>&lt;your-access-key-id&gt; /&lt;date&gt;/&lt;aws-region&gt; /&lt;aws-service&gt; /aws4_request</pre> <p>例えば、次のようになります。</p> <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> <p>Amazon S3 の場合、aws-service 文字列は s3 です。Amazon S3 aws-region 文字列のリストについては、「AWS 全般のリファレンス」の「<a href="#">リージョンとエンドポイント</a>」を参照してください。これは、ポリシードキュメントがリクエストに含まれる場合に必要です。</p> | 認証されたリクエストに必須 |

| 要素名                  | 説明                                                                                                                                                                                                                                                                                                                                                                                                      | 必須            |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| x-amz-date           | <p>ISO8601 形式の日付値です。例えば、20130728T000000Z です。</p> <p>署名キーの作成に使用したのと同じ日付です (例: 20130728)。これは、署名したポリシー (x-amz-date ) で指定した値と同じである必要があります。</p> <p>これは、ポリシードキュメントがリクエストに含まれる場合に必要です。</p>                                                                                                                                                                                                                     | 認証されたリクエストに必須 |
| x-amz-security-token | <p>Amazon DevPay およびセッション認証情報で使われるセキュリティトークン</p> <p>リクエストで Amazon DevPay を使用する場合は、製品トークン用とユーザートークン用の 2 つの x-amz-security-token フォームフィールドが必要です。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">DevPay の使用</a>」を参照してください。</p> <p>リクエストでセッション認証情報が使用される場合は、単一の x-amz-security-token フォームが必要です。詳細については、「IAM ユーザーガイド」の「<a href="#">Requesting Temporary Security Credentials</a>」を参照してください。</p> | いいえ           |
| x-amz-signature      | <p>(AWS 署名バージョン 4) セキュリティポリシーの HMAC-SHA256 ハッシュ。</p> <p>このフィールドは、ポリシードキュメントがリクエストに含まれる場合に必要です。</p>                                                                                                                                                                                                                                                                                                      | 認証されたリクエストに必須 |



| 要素名          | 説明                                                                                                                                                 | 必須  |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| x-amz-meta-* | このプレフィックスで始まるフィールド名はユーザー定義メタデータです。それぞれが保存され、キーと値のペアのセットとして返されます。Amazon S3 はユーザー定義メタデータを検証または解釈しません。詳細については、「 <a href="#">PutObject</a> 」を参照してください。 | いいえ |
| x-amz-*      | 他の x-amz-* ヘッダーについては、「POST Object」( <a href="#">POST Object</a> ) を参照してください。                                                                       | いいえ |
| file         | ファイルまたはテキストコンテンツ。<br><br>ファイルまたはコンテンツはフォームの最後のフィールドでなければなりません。<br><br>複数のファイルを一度にアップロードすることはできません。                                                 | はい  |

条件付き項目は認証されたリクエストに必要であり、匿名リクエストではオプションです。

フォームの作成方法がわかったので、次に署名できるセキュリティポリシーを作成できます。詳細については、「[POST ポリシー](#)」を参照してください。

## POST ポリシー

### トピック

- [有効期限](#)
- [条件一致](#)
- [条件](#)
- [文字のエスケープ](#)

HTTP POST を使用して認証されたリクエストを行うために必要なポリシーは、リクエストが満たす必要がある条件を指定する JavaScript Object Notation (JSON) で記述された UTF-8 および base64 でエンコードされたドキュメントです。ポリシードキュメントは、どのように設計するかに応じて、アップロードごと、ユーザーごと、すべてのアップロードに対して、またはニーズに応じたその他の設計にしたがってアクセスの粒度を制御できます。

このセクションでは、POST ポリシーについて説明します。POST ポリシーを使用した署名計算の例については、「[例: HTTP POST \(AWS 署名バージョン 4\) を使用したブラウザベースのアップロード](#)」を参照してください。

#### Note

ポリシードキュメントはオプションですが、リクエストで許可される内容を制御するために、ポリシードキュメントを使用することを強くお勧めします。バケットをパブリックに書き込み可能にした場合、どのユーザーがバケットに書き込めるかをまったく制御できません。

POST ポリシードキュメントの例を次に示します。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

POST ポリシーには、常に expiration および conditions 要素が含まれます。このポリシー例では、2 つの条件一致タイプ (完全一致と starts-with 一致) を使用しています。以下のセクションでは、これらの要素について説明します。

## 有効期限

expiration 要素は、POST ポリシーの有効期限日を ISO 8601 GMT 日付形式で指定します。例えば、2013-08-01T12:00:00.000Z は、2013 年 8 月 1 日午前 0 時 GMT より後は POST ポリシーが無効になることを指定します。

## 条件一致

以下は、POST ポリシー条件を指定するために使用できる条件一致タイプを説明する表です (次のセクションで説明)。フォーム内で指定したフォームフィールドごとに少なくとも 1 つの条件を指定する必要がありますが、1 つのフォームフィールドに複数の条件を指定し、より複雑な一致条件を作成することもできます。

| 条件一致タイプ                | 説明                                                                                                                                                                                                                                       |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 完全一致                   | <p>フォームフィールドの値は、指定された値と一致する必要があります。この例は、ACL が public-read に設定される必要があることを示しています:</p> <pre>{"acl": "public-read" }</pre> <p>この例は、ACL が public-read に設定される必要があることを別の方法で示しています:</p> <pre>[ "eq", "\$acl", "public-read" ]</pre>              |
| 先頭が一致                  | <p>値は、指定された値で始まる必要があります。次の例は、オブジェクトキーが user/user1 で開始する必要があることを示しています。</p> <pre>["starts-with", "\$key", "user/user1/"]</pre>                                                                                                            |
| カンマ区切りリスト内のコンテンツタイプの一致 | <p>カンマを含む starts-with 条件の Content-Types 値はリストとして解釈されます。リスト内の各値は、条件全体が合格するための条件を満たしている必要があります。例えば、次の設定があるとします。</p> <pre>["starts-with", "\$Content-Type", "image/"]</pre> <p>次の値は条件を満たします。</p> <pre>"image/jpg,image/png,image/gif"</pre> |

| 条件一致タイプ     | 説明                                                                                                                                                                                                                    |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <p>次の値は条件を満たしません。</p> <pre>["image/jpg,text/plain"]</pre> <div> <b>Note</b><br/>Content-Type 以外のデータ要素は、カンマの有無に関係なく文字列として扱われます。</div> |
| 任意のコンテンツに一致 | <p>フィールド内の任意のコンテンツを許可するように POST ポリシーを設定するには、starts-with に空の値 ("" ) を使用します。この例では、success_action_redirect に任意の値を許可します。</p> <pre>["starts-with", "\$success_action_redirect", ""]</pre>                                  |
| 範囲を指定       | <p>範囲指定できるフォームフィールドについては、範囲の上限値と下限値をコンマで区切ります。次の例は、1～10 MiB のサイズのファイルを許可します。</p> <pre>["content-length-range", 1048576, 10485760]</pre>                                                                               |

POST ポリシーでサポートされている特定の条件については、「[条件](#)」を参照してください。

## 条件

POST ポリシーの conditions はオブジェクトの配列であり、それぞれがリクエストの検証に使用されます。これらの条件を使用して、リクエストで許可される内容を制限できます。例えば、前述のポリシー条件には以下が必要です。

- リクエストでは、johnsmith バケット名を指定する必要があります。
- オブジェクトキー名には user/eric プレフィックスが必要です。
- オブジェクト ACL は public-read に設定する必要があります。

フォームで指定する各フォームフィールド (x-amz-signature、file、policy、および x-ignore- プレフィックスを持つフィールド名を除く) は、条件のリストに表示される必要があります。

 Note

フォーム内のすべての変数は、POST ポリシーの検証前に展開されます。したがって、条件との照合は必ず、展開されたフォームフィールドに対して行われます。オブジェクトキー名を特定のプレフィックス (user/user1) に制限するとします。この場合、キーフォームフィールドを user/user1/\${filename} に設定します。POST ポリシーは [ "starts-with", "\$key", "user/user1/" ] である必要があります ( [ "starts-with", "\$key", "user/user1/\${filename}" ] は入力しないでください)。詳細については、「[条件一致](#)」を参照してください。

以下の表にポリシードキュメント条件を示します。

| 要素名                  | 説明                                                                                                |
|----------------------|---------------------------------------------------------------------------------------------------|
| acl                  | <p>フォームの送信で使用する必要がある ACL 値を指定します。</p> <p>この条件は、次のセクションで説明する完全一致と starts-with 条件一致タイプをサポートします。</p> |
| bucket               | <p>許容されるバケット名を指定します。</p> <p>この条件は、完全に一致する条件一致タイプをサポートします。</p>                                     |
| content-length-range | <p>アップロードされたコンテンツの最小サイズと最大サイズ。</p> <p>この条件は、content-length-range 条件一致タイプをサポートします。</p>             |
| Cache-Control        |                                                                                                   |

| 要素名                                 | 説明                                                                                                            |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------|
| Content-Type                        | REST 固有ヘッダー。詳細については、「 <a href="#">POST Object</a> 」を参照してください。<br><br>この条件は、完全一致と starts-with 条件一致タイプをサポートします。 |
| Content-Disposition                 |                                                                                                               |
| Content-Encoding                    |                                                                                                               |
| Expires                             |                                                                                                               |
| key                                 | アップロードされたオブジェクトの許容可能なキー名またはプレフィックス。<br><br>この条件は、完全一致と starts-with 条件一致タイプをサポートします。                           |
| success_action_redirect<br>redirect | アップロードが成功したときにクライアントがリダイレクトされる URL。<br><br>この条件は、完全一致と starts-with 条件一致タイプをサポートします。                           |
| success_action_status               | アップロードが正常に行われたとき、success_action_redirect が指定されていない場合にクライアントに返されるステータスコード。<br><br>この条件は完全一致をサポートします。           |
| x-amz-algorithm                     | 署名の計算の際に使用する必要がある署名アルゴリズム。AWS 署名バージョン 4 の場合、値は AWS4-HMAC-SHA256 です。<br><br>この条件は完全一致をサポートします。                 |

| 要素名              | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x-amz-credential | <p>署名の計算に使用した認証情報。署名が有効なリージョンおよびサービスを特定するアクセスキー ID とスコープを指定します。これは、署名計算の署名キーの計算に使用したスコープと同じである必要があります。</p> <p>これは、次のような形式の文字列です。</p> <pre>&lt;your-access-key-id&gt; /&lt;date&gt;/&lt;aws-region&gt; /&lt;aws-service&gt; /aws4_request</pre> <p>例えば、次のようになります。</p> <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> <p>Amazon S3 の場合、aws-service 文字列は s3 です。Amazon S3 aws-region 文字列のリストについては、「AWS 全般のリファレンス」の「<a href="#">リージョンとエンドポイント</a>」を参照してください。このフィールドは、POST ポリシードキュメントがリクエストに含まれる場合に必要です。</p> <p>この条件は完全一致をサポートします。</p> |
| x-amz-date       | <p>ISO8601 形式の文字列で指定された日付値。例えば、20130728T000000Z です。日付は、署名計算の署名キーの作成に使用した日付と同じである必要があります。</p> <p>このフィールドは、POST ポリシードキュメントがリクエストに含まれる場合に必要です。</p> <p>この条件は完全一致をサポートします。</p>                                                                                                                                                                                                                                                                                                                                                                            |

| 要素名                  | 説明                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x-amz-security-token | <p>Amazon DevPay のセキュリティトークン。</p> <p>Amazon DevPay を使用する各リクエストでは、製品トークン用とユーザートークン用の 2 つの x-amz-security-token フォームフィールドが必要です。結果的に、値はコンマで区切られていなければなりません。例えば、ユーザートークンが eW91dHViZQ== で、製品トークンが b0hnNVNKWVJIQTA= の場合、POST ポリシーエントリは { "x-amz-security-token": "eW91dHViZQ==,b0hnNVNKWVJIQTA=" } に設定します。</p> <p>Amazon DevPay の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">DevPay の使用</a>」を参照してください。</p> |
| x-amz-meta-*         | <p>ユーザーが指定したメタデータ。</p> <p>この条件は、完全一致と starts-with 条件一致タイプをサポートします。</p>                                                                                                                                                                                                                                                                                                                                                |
| x-amz-*              | <p>他の x-amz-* ヘッダーについては、「POST Object」(<a href="#">POST Object</a>) を参照してください。</p> <p>この条件は完全一致をサポートします。</p>                                                                                                                                                                                                                                                                                                           |

**Note**

ツールキットによりフォームフィールドが追加されている場合は (例えば、Flash では filename が追加される)、そのフィールドを POST ポリシードキュメントに追加する必要があります。この機能を管理できる場合は、そのフィールドの機能が Amazon S3 に無視されるようにプレフィックスの x-ignore- をフィールドに追加します。これは、そのフィールドの機能の将来のバージョンには影響しません。



## 文字のエスケープ

以下の表に、POST ポリシードキュメント内でエスケープする必要のある文字を示します。

| エスケープ<br>シーケンス | 説明              |
|----------------|-----------------|
| \\             | バックスラッシュ        |
| \\\$           | ドル記号            |
| \\b            | Backspace       |
| \\f            | フォームフィード        |
| \\n            | 改行              |
| \\r            | キャリッジリターン       |
| \\t            | 水平タブ            |
| \\v            | 垂直タブ            |
| \\uxxxx        | すべての Unicode 文字 |

フォームとポリシーを理解し、署名の仕組みを理解したら、POST アップロードの例を試すことができます。署名を計算するには、コードを記述する必要があります。この例では、サンプルフォームと、署名計算をテストするために使用できる POST ポリシーを提供します。詳細については、「[例: HTTP POST \(AWS 署名バージョン 4\) を使用したブラウザベースのアップロード](#)」を参照してください。

# 例: HTTP POST (AWS 署名バージョン 4) を使用したブラウザベースのアップロード

このセクションでは、HTTP POST リクエストを使用してコンテンツを Amazon S3 に直接アップロードする例を示します。

署名バージョン 4 の詳細については、「[署名バージョン 4 の署名プロセス](#)」を参照してください。

## HTTP POST を使用した Amazon S3 へのファイルのアップロード

この例では、ファイルのアップロードに使用できるサンプルの POST ポリシーとフォームを示します。このトピックでは、ポリシーと架空の認証情報の例は、ワークフローおよびその結果として生成される署名とポリシーハッシュを示しています。このデータをテストスイートとして使用して、署名計算コードを検証できます。

この例では、次のサンプルの認証情報を使用して署名計算を行います。これらの認証情報を使用して、署名計算コードを検証できます。ただし、AWS にリクエストを送信するときは、これらを独自の認証情報に置き換える必要があります。

| パラメータ              | 値                                        |
|--------------------|------------------------------------------|
| AWSSecretAccessKey | AKIAIOSFODNN7EXAMPLE                     |
| AWSAccessKeyId     | wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY |

## サンプルポリシーとフォーム

次の POST ポリシーは、特定の条件での Amazon S3 へのアップロードをサポートしています。

```
{ "expiration": "2015-12-30T12:00:00.000Z",
  "conditions": [
    {"bucket": "sigv4examplebucket"},
    ["starts-with", "$key", "user/user1/"],
    {"acl": "public-read"},
    {"success_action_redirect": "http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"}
  ]
}
```

```
{
  "x-amz-server-side-encryption": "AES256",
  ["starts-with", "$x-amz-meta-tag", ""],

  {"x-amz-credential": "AKIAIOSFODNN7EXAMPLE/20151229/us-east-1/s3/aws4_request"},
  {"x-amz-algorithm": "AWS4-HMAC-SHA256"},
  {"x-amz-date": "20151229T000000Z" }
]
}
```

この POST ポリシーは、リクエストに次の条件を設定します。

- アップロードは、2015 年 12 月 30 日の UTC 正午までに行う必要があります。
- コンテンツは sigv4examplebucket にのみアップロードできます。指定した署名はこのスコープ内でのみ有効であるため、バケットは認証情報スコープ (x-amz-credential フォームパラメータ) で指定したリージョンにある必要があります。
- user/user1 で始まる任意のキー名を指定できます。例えば、user/user1/MyPhoto.jpg。
- ACL は public-read に設定する必要があります。
- アップロードが成功すると、ユーザーのブラウザは `http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html` にリダイレクトされます。
- オブジェクトはイメージファイルである必要があります。
- x-amz-meta-uuid タグは 14365123651274 に設定する必要があります。
- x-amz-meta-tag には任意の値を含めることができます。

この POST ポリシーの Base64 エンコードされたバージョンを次に示します。この値は、署名計算で StringToSign として使用します。

```
eyJhZiZxhwaXJhdGlvbiI6IClYMDl1TEYlTMwVDEyOjAwOjAwLjAwMFoiLA0KICAiY29uZGl0aW9ucyI6IFsNCiAgICB7ImJ
```

上記のポリシーをコピー/貼り付ける際は、計算されたハッシュがこの値と一致するように、キャリッジリターンと改行 (つまり、CRLF 行末記号付きの ASCII テキスト) が含まれている必要があります。

サンプルの認証情報を使用して署名を作成する場合、署名の値は次のようになります (署名の計算では、日付はポリシーの x-amz-date (20151229) と同じです)。

```
8afdbf4008c03f22c2cd3cdb72e4afbb1f6a588f3255ac628749a66d7f09699e
```

次のフォームの例では、前述の POST ポリシーを指定し、sigv4examplebucket への POST リクエストをサポートしています。テキストエディタでコンテンツをコピーして貼り付け、exampleform.html として保存します。その後、exampleform.html を使用して、イメージファイルを特定のバケットにアップロードできます。提供された署名が Amazon S3 が計算した署名と一致する場合、リクエストは成功します。

 Note

S3 に正常にアップロードするには、バケット名、日付、認証情報、ポリシー、および署名を有効な値に更新する必要があります。

```
<html>
  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

  </head>
  <body>

    <form action="http://sigv4examplebucket.s3.amazonaws.com/" method="post"
    enctype="multipart/form-data">
      Key to upload:
      <input type="input" name="key" value="user/user1/${filename}" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="http://
sigv4examplebucket.s3.amazonaws.com/successful_upload.html" />
      Content-Type:
      <input type="input" name="Content-Type" value="image/jpeg" /><br />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      <input type="hidden" name="x-amz-server-side-encryption" value="AES256" />
      <input type="text" name="X-Amz-Credential" value="AKIAIOSFODNN7EXAMPLE/20151229/
us-east-1/s3/aws4_request" />
      <input type="text" name="X-Amz-Algorithm" value="AWS4-HMAC-SHA256" />
      <input type="text" name="X-Amz-Date" value="20151229T000000Z" />

      Tags for File:
      <input type="input" name="x-amz-meta-tag" value="" /><br />
      <input type="hidden" name="Policy" value='<Base64-encoded policy string>' />
      <input type="hidden" name="X-Amz-Signature" value="<signature-value>" />
      File:
      <input type="file" name="file" /> <br />
```

```
<!-- The elements after this will be ignored -->
<input type="submit" name="submit" value="Upload to Amazon S3" />
</form>

</html>
```

POST パラメータでは、大文字と小文字は区別されません。たとえば、`x-amz-signature` や `X-Amz-Signature` と指定することができます。

## AWS Amplify ライブラリを使用した Amazon S3 へのブラウザベースのアップロード

このセクションでは、AWS Amplify JavaScript ライブラリを使用して Amazon S3 にファイルをアップロードする方法について説明します。

AWS Amplify ライブラリの設定については、「[AWS Amplify のインストールと設定](#)」を参照してください。

### AWS Amplify JavaScript ライブラリを使用した Amazon S3 へのファイルのアップロード

AWS Amplify ライブラリ Storage モジュールは、パブリックまたはプライベート Amazon S3 ストレージでユーザーコンテンツを管理するためのシンプルなブラウザベースのアップロードメカニズムを提供します。

#### Example: AWS Amplify 手動セットアップ

次の例は、AWS Amplify ストレージモジュールを使用するための手動セットアップを示しています。Storage モジュールのデフォルトの実装では、Amazon S3 を使用します。

```
import Amplify from 'aws-amplify';
Amplify.configure({
  Auth: {
    identityPoolId: 'XX-XXXX-X:XXXXXXXXX-XXXX-1234-abcd-1234567890ab', //REQUIRED -
    Amazon Cognito Identity Pool ID
    region: 'XX-XXXX-X', // REQUIRED - Amazon Cognito Region
    userPoolId: 'XX-XXXX-X_abcd1234', //OPTIONAL - Amazon Cognito User Pool ID
    userPoolWebClientId: 'XX-XXXX-X_abcd1234', //OPTIONAL - Amazon Cognito Web
    Client ID
  },
```

```
Storage: {  
  bucket: '', //REQUIRED - Amazon S3 bucket  
  region: 'XX-XXXX-X', //OPTIONAL - Amazon service region  
}  
);
```

### Example: Amazon S3 にデータを配置する

次の例に、Amazon S3 にパブリックデータを put する方法を示します。

```
Storage.put('test.txt', 'Hello')  
  .then (result => console.log(result))  
  .catch(err => console.log(err));
```

次の例に、Amazon S3 にパブリックデータを put する方法を示します。

```
Storage.put('test.txt', 'Private Content', {  
  level: 'private',  
  contentType: 'text/plain'  
})  
  .then (result => console.log(result))  
  .catch(err => console.log(err));
```

AWS Amplify Storage モジュールの使用の詳細については、「[AWS Amplify Storage](#)」を参照してください。

## 詳細

### [AWS Amplify クイックスタート](#)

# 一般的なリクエストヘッダー

次の表では、さまざまなタイプの Amazon S3 REST リクエストで使用できるヘッダーについて説明します。

| ヘッダー名                         | 説明                                                                                                                                                                                                                                                                                    |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Authorization                 | リクエスト認証に必要な情報。詳細については、「Amazon Simple Storage Service 開発者ガイド」の「 <a href="#">認証ヘッダー</a> 」を参照してください。匿名リクエストの場合、このヘッダーは必要ありません。                                                                                                                                                           |
| Access-Control-Request-Method | 事前 CORS リクエストとして送信される HTTP メソッドのリスト。事前 CORS 評価が成功すると、次の CORS リクエストで指定されたメソッドを使用できます。                                                                                                                                                                                                  |
| Content-Length                | RFC 2616 に基づくメッセージの長さ (ヘッダーなし)。このヘッダーは、ログ記録や ACL など、XML をロードする PUT およびオペレーションに必要です。                                                                                                                                                                                                   |
| Content-Type                  | リクエストの本文にコンテンツがある場合のリソースのコンテンツタイプ。例:text/plain                                                                                                                                                                                                                                        |
| Content-MD5                   | RFC 1864 に従ってメッセージの base64 エンコードされた 128 ビット MD5 ダイジェスト (ヘッダーなし)。このヘッダーをメッセージ整合性チェックとして使用し、データが最初送信されたデータと同じであることを確認します。オプションですが、エンドツーエンドの整合性チェックとして Content-MD5 メカニズムを使用することをお勧めします。REST リクエスト認証の詳細については、「Amazon Simple Storage Service 開発者ガイド」の「 <a href="#">REST 認証</a> 」を参照してください。 |
| Date                          | Authorization ヘッダーに含める署名を作成するときに使用できる日付。Date ヘッダーを署名に使用する場合は、ISO 8601 基本形式で指定する必要があります。その場合、x-amz-date ヘッダーは必要ありません。x-amz-date が存                                                                                                                                                     |

| ヘッダー名  | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | <p>在する場合は、常に Date ヘッダーの値よりも優先されることに注意してください。</p> <p>Date ヘッダーを署名に使用しない場合は、<a href="#">RFC 2616</a> のセクション 3.3 に記載されている完全な日付の形式のいずれかを使用できます。例えば、Wed, 01 Mar 2006 12:00:00 GMT という日付/時間は、Amazon S3 で使用できる有効な日付/時間ヘッダーです。</p> <p>Date ヘッダーを署名に使用する場合は、ISO 8601 基本形式の YYYYMMDD'T'HHMMSS'Z' で指定する必要があります。</p> <p>Date が指定されていても、ISO 8601 基本形式でない場合は、x-amz-date ヘッダーも含める必要があります。Date を ISO 8601 基本形式で指定した場合は、署名リクエストに十分に対応できるため、x-amz-date ヘッダーは必要ありません。詳細については、「<a href="#">署名バージョン 4 の日付の処理</a>」(アマゾン ウェブ サービス用語集) を参照してください。</p> |
| Expect | <p>100-continue を使用しているアプリケーションは、確認を受け取るまでリクエスト本文を送信しません。ヘッダーに基づいてメッセージが拒否された場合、メッセージの本文は送信されません。このヘッダーは、本文を送信する場合にのみ使用できます。</p> <p>有効な値: 100-continue</p>                                                                                                                                                                                                                                                                                                                                                                    |
| Host   | <p>パス形式のリクエストの場合、値は s3.amazonaws.com です。仮想形式のリクエストの場合、値は BucketName.s3.amazonaws.com です。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">仮想ホスティング</a>」を参照してください。</p> <p>このヘッダーは HTTP 1.1 に必要です (ほとんどのツールキットではこのヘッダーが自動的に追加されます)。HTTP/1.0 リクエストではオプションです。</p>                                                                                                                                                                                                                                                      |



| ヘッダー名                | 説明                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Origin               | 初期リクエストのサーバー名を指定するエンドポイント。                                                                                                                                                                                                                                                                                                                                                                                      |
| x-amz-content-sha256 | 署名バージョン 4 を使用してリクエストを認証する場合、このヘッダーはリクエストペイロードのハッシュを提供します。詳細については、「 <a href="#">Authorization ヘッダーの署名計算: 単一チャンクでのペイロードの転送 (AWS 署名バージョン 4)</a> 」を参照してください。オブジェクトをチャンクでアップロードする場合、署名がヘッダーのみを対象とし、ペイロードがないことを示す値を STREAMING-AWS4-HMAC-SHA256-PAYLOAD に設定します。詳細については、「 <a href="#">Authorization ヘッダーの署名計算: 複数チャンクでのペイロードの転送 (チャンクアップロード) (AWS 署名バージョン 4)</a> 」を参照してください。                                            |
| x-amz-date           | <p>Authorization ヘッダー内の署名を作成するときに使用される日付。ISO 8601 基本形式の YYYYMMDD'T'HHMMSS'Z' にする必要があります。例えば、20170210T120000Z という日付/時間は、Amazon S3 で使用できる有効な x-amz-date です。</p> <p>x-amz-date はすべてのリクエストに対してオプションです。署名リクエストで使用する日付よりも優先される日付として使用できます。Date ヘッダーを ISO 8601 基本形式で指定した場合、x-amz-date は必要ありません。x-amz-date が存在する場合は、常に Date ヘッダーの値よりも優先されます。詳細については、「<a href="#">署名バージョン 4 の日付の処理</a>」(アマゾン ウェブ サービス用語集) を参照してください。</p> |

| ヘッダー名                | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x-amz-security-token | <p data-bbox="613 226 1333 260">このヘッダーは、以下のシナリオで使用できます。</p> <ul data-bbox="613 310 1507 1037" style="list-style-type: none"><li data-bbox="613 310 1507 709">• Amazon DevPay オペレーションのセキュリティトークンを提供するには、Amazon DevPay を使用する各リクエストに 2 つの x-amz-security-token ヘッダーが必要です。1 つは製品トークン用、もう 1 つはユーザートークン用です。Amazon S3 は、認証されたリクエストを受信すると、計算した署名を提供された署名と比較します。署名の計算に使用される複数値ヘッダーの形式が正しくないと、認証の問題が発生する可能性があります。</li><li data-bbox="613 730 1507 1037">• 一時的なセキュリティ認証情報を使用する際にセキュリティトークンを提供するには – IAM から取得した一時的なセキュリティ認証情報を使用してリクエストを行う場合は、このヘッダーを使用してセキュリティトークンを提供する必要があります。一時的なセキュリティ認証情報の詳細については、「<a href="#">リクエストの作成</a>」を参照してください。</li></ul> <p data-bbox="613 1108 1507 1239">このヘッダーは、Amazon DevPay を使用するリクエストと、一時的なセキュリティ認証情報を使用して署名されたリクエストに必要です。</p> |

## 共通のレスポンスヘッダー

以下の表は、ほとんどの Amazon S3 レスポンスに共通のレスポンスヘッダーを説明したものです。

| 名前                               | 説明                                                                                                                                                                                                                                  |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Access-Control-Allow-Credentials | <p>サーバーが CORS リクエストに認証情報を含めることを許可するかどうかを決定するブール値。Access-Control-Allow-Origin リクエストヘッダーが '*' に設定されている場合、Access-Control-Allow-Credentials レスポンスヘッダーは省略され、それ以外の場合は CORS 評価が成功したときに true に設定されます。</p> <p>タイプ: ブール値</p> <p>デフォルト: なし</p> |
| Access-Control-Allow-Headers     | <p>CORS リクエストで許可される HTTP ヘッダーのリスト。Access-Control-Allow-Headers レスポンスヘッダーは、成功した CORS 評価に対して返され、許可されたすべての Access-Control-Request-Headers を明示的に指定します。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p>                                             |
| Access-Control-Allow-Methods     | <p>許可される HTTP メソッドを指定するリスト。Amazon S3 は、CORS 評価が成功した場合にのみ、許可された CORS メソッドからの CORS リクエストを許可します。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p>                                                                                                  |
| Access-Control-Allow-Origin      | <p>許可されたオリジンの場所。Amazon S3 は、CORS 評価が成功した場合にのみ Access-Control-Allow-Origin レスポンスヘッダーを送信します。リクエストオリジンが CORS 設定で許可されているオリジン '*' と一致する場合、'*' は元のオリジンではなく、このレスポンスヘッダーで返されます。</p>                                                         |

| 名前                            | 説明                                                                                                                                                                     |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                               | タイプ: 文字列<br><br>デフォルト: なし                                                                                                                                              |
| Access-Control-Expose-Headers | CORS 評価が成功したときにアプリケーションへのアクセスを公開するレスポンスヘッダーをサーバーが識別できるようにするリスト。<br><br>タイプ: 文字列<br><br>デフォルト: なし                                                                       |
| Access-Control-Max-Age        | リソース、HTTP メソッド、およびオリジンによって識別される CORS プリフライトリクエストのレスポンスを、ブラウザがキャッシュできる時間 (秒数)。Access-Control-Max-Age レスポンスヘッダーは CORS 評価が成功した場合にのみ返されます。<br><br>タイプ: 整数<br><br>デフォルト: なし |
| Vary                          | CORS 評価結果が変化するリクエストヘッダーを示すリスト。Vary レスポンスヘッダーは CORS 評価が成功した場合にのみ返されます。<br><br>タイプ: 文字列<br><br>デフォルト: なし                                                                 |
| Content-Length                | レスポンスの本文のバイト長。<br><br>タイプ: 文字列<br><br>デフォルト: なし                                                                                                                        |
| Content-Type                  | コンテンツの MIME タイプ。例えば、Content-Type: text/html; charset=utf-8 です。<br><br>タイプ: 文字列<br><br>デフォルト: なし                                                                        |

| 名前         | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Connection | <p>サーバーへの接続が開いているか閉じているかを指定する値。</p> <p>タイプ: Enum</p> <p>有効な値: open   close</p> <p>デフォルト: なし</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Date       | <p>Amazon S3 が応答した日時 (Wed, 01 Mar 2006 12:00:00 GMT など)。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| ETag       | <p>オブジェクトのエンティティタグ (ETag) は、そのオブジェクトの特定のバージョンを表します。ETag は、オブジェクトのコンテンツに加えた変更のみを反映し、メタデータに加えた変更は反映しません。ETag は、オブジェクトデータの MD5 ダイジェストである場合と、そうでない場合があります。そのどちらになるかは、オブジェクトの作成方法や暗号化方法によって以下のように決まります。</p> <ul style="list-style-type: none"><li>• AWS マネジメントコンソールまたは PUT Object、POST Object、または Copy オペレーションによって作成されたオブジェクト:</li><li>• Amazon S3 マネージドキー (SSE-S3) によるサーバー側の暗号化によってプレーンテキストまたは暗号化されたオブジェクトには、データの MD5 ダイジェストである ETag があります。</li><li>• お客様が用意したキー (SSE-C) または AWS Key Management Service (AWS KMS) キー (SSE-KMS) によるサーバー側の暗号化によって暗号化されたオブジェクトには、オブジェクトデータの MD5 ダイジェストではない ETag があります。</li><li>• マルチパートアップロードまたはパートコピーのアップロードオペレーションのいずれかによって作成されたオブジェクトには、暗号化の方法に関係なく、MD5 ダイジェストではない ETag があります。</li></ul> <p>タイプ: 文字列</p> |

| 名前                           | 説明                                                                                                                                                                                                                          |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Server                       | <p>レスポンスを作成したサーバーの名前。</p> <p>タイプ: 文字列</p> <p>デフォルト: AmazonS3</p>                                                                                                                                                            |
| x-amz-delete-marker          | <p>返されたオブジェクトが削除マーカーであったか (true) そうでなかったか (false) を指定する値。</p> <p>タイプ: ブール値</p> <p>有効な値: true   false</p> <p>デフォルト: false</p>                                                                                                |
| x-amz-id-2                   | <p>AWS が問題のトラブルシューティングを行う際に、x-amz-request-id ヘッダーと組み合わせて使用される特別なトークン。これらのリクエスト ID を使用したサポートの詳細については、「<a href="#">Amazon S3 のトラブルシューティング</a>」を参照してください。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p>                                 |
| x-amz-request-id             | <p>Amazon S3 によって作成される値であり、リクエストを一意に識別します。この値は AWS が問題のトラブルシューティングを行う際に、x-amz-id-2 ヘッダーと組み合わせて使用されます。これらのリクエスト ID を使用したサポートの詳細については、「<a href="#">Amazon S3 のトラブルシューティング</a>」を参照してください。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> |
| x-amz-server-side-encryption | <p>Amazon S3 へのこのオブジェクトの保存時に使用されるサーバー側の暗号化アルゴリズム (AES256、aws:kms など)。</p> <p>有効な値: AES256   aws:kms</p>                                                                                                                     |

| 名前               | 説明                                                                                                                                                                                                                                                       |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x-amz-version-id | <p>オブジェクトのバージョン。バージョニングを有効にすると、Amazon S3 はバケットに追加されたオブジェクトのランダムな番号を生成します。値は UTF-8 でエンコードされており、URL に対応しています。バージョニングが中断されているバケット内のオブジェクトを PUT する場合、バージョン ID は常に null です。</p> <p>タイプ: 文字列</p> <p>有効な値: null   URL 対応の UTF-8 エンコード文字列</p> <p>デフォルト: null</p> |

# エラーレスポンス

このセクションでは、Amazon S3 エラーに関するリファレンス情報を提供します。

## Note

- 一般に、S3 バケット所有者は、HTTP 200 OK 成功レスポンスと HTTP 4XX クライアントエラーレスポンスを含むリクエストに対して課金されます。バケット所有者は、HTTP 503 Slow Down エラーなどの HTTP 5XX サーバーエラーレスポンスに対しては課金されません。請求されない HTTP 3XX および 4XX ステータスコードの S3 エラーコードの詳細については、「Amazon S3 ユーザーガイド」の「[Amazon S3 のエラーレスポンスに対する請求](#)」を参照してください。バケットがリクエスト支払いバケットとして設定されている場合の請求の詳細については、「Amazon S3 ユーザーガイド」の「[リクエスト支払いの課金の仕組み](#)」を参照してください。
- SOAP のサポートは HTTP 経由では廃止されましたが、HTTPS 経由では引き続き利用可能です。Amazon S3 の新機能は SOAP でサポートされていません。SOAP の代わりに、REST API か AWS SDK を使用することをお勧めします。

## トピック

- [REST エラーレスポンス](#)
- [エラーコードのリスト](#)
- [SELECT Object Content エラーコードのリスト](#)
- [レプリケーション関連のエラーコードのリスト](#)
- [タグ付け関連のエラーコードのリスト](#)
- [Amazon S3 注釈関連のエラーコードのリスト](#)
- [Amazon S3 on Outposts エラーコードのリスト](#)
- [Amazon S3 ストレージレンズエラーコードのリスト](#)
- [Amazon S3 Object Lambda エラーコードのリスト](#)
- [Amazon S3 の非同期エラーコードのリスト](#)
- [Amazon S3 Access Grants エラーコードのリスト](#)
- [Amazon FSx 関連のエラーコードのリスト](#)



- [Amazon S3 Tables のエラーコードのリスト](#)
- [Amazon S3 のエラーに関するベストプラクティス](#)

## REST エラーレスポンス

エラーが発生した場合、ヘッダー情報には、以下の項目が含まれます。

- コンテンツタイプ: application/xml
- 該当する HTTP ステータスコード (3xx、4xx、または 5xx)

さらにレスポンスの本文には、エラーに関する情報も含まれます。次のサンプルエラーは、すべての REST エラーレスポンスに共通する、レスポンス要素の構文を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

次の表に、REST エラーレスポンスの要素を示します。

| 名前    | 説明                                                                                                                                                              |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Code  | <p>エラーコードは、エラー状態を個別に識別する文字列です。エラーを検出してタイプ別に処理するプログラムによって読み取りおよび解釈されるためのものです。詳細については、「<a href="#">エラーコードのリスト</a>」を参照してください。</p> <p>タイプ: 文字列</p> <p>祖先: Error</p> |
| Error | <p>すべてのエラー要素のコンテナです。</p> <p>型: コンテナ</p> <p>祖先: なし</p>                                                                                                           |

| 名前        | 説明                                                                                                                                                                                                                             |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Message   | <p>エラーメッセージには、エラー状態の一般的な説明が英語で含まれます。これは人が理解できるようにするためのものです。シンプルなプログラムの場合、処理できない、または処理されないエラー状態が発生すると、メッセージがエンドユーザーに直接表示されます。より徹底したエラー処理を備え、適切に国際化されている洗練されたプログラムでは、エラーメッセージが無視される傾向にあります。</p> <p>タイプ: 文字列</p> <p>祖先: Error</p> |
| RequestId | <p>エラーに関連付けられたリクエストの ID です。</p> <p>タイプ: 文字列</p> <p>祖先: Error</p>                                                                                                                                                               |
| Resource  | <p>エラーに関連するバケットまたはオブジェクトです。</p> <p>タイプ: 文字列</p> <p>祖先: Error</p>                                                                                                                                                               |

多くのエラーレスポンスには、プログラミングエラーを診断する開発者が読んで理解できるように追加の構造化データが含まれています。例えば、REST PUT リクエストと共に Content-MD5 ヘッダーを送信し、これがサーバーで計算されたダイジェストと一致しない場合、BadDigest エラーが発生します。エラーレスポンスには、サーバーが計算したダイジェスト、およびサーバーに指定したダイジェストなどの詳細要素も含まれます。開発中にこの情報を使用してエラーを診断できます。実稼働では、正常に動作するプログラムによりこの情報がエラーログに記録されることがあります。

一般的なレスポンス要素については、「[エラーレスポンス](#)」を参照してください。

## エラーコードのリスト

次の表に、Amazon S3 エラーコードを示します。

| エラーコード                        | 説明                                                                                                   | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|-------------------------------|------------------------------------------------------------------------------------------------------|-----------------|--------------------|
| AccessControlListNotSupported | バケットでは ACL が許可されません。                                                                                 | 400 Bad Request | クライアント             |
| AccessDenied                  | アクセスが拒否されました                                                                                         | 403 Forbidden   | クライアント             |
| AccessPointAlreadyOwnedByYou  | アカウントには、同じ名前のアクセスポイントが既に存在します。                                                                       | 409 Conflict    | クライアント             |
| AccountProblem                | AWS アカウント アカウントに問題があり、それによりオペレーションを正常に完了することができません。詳細については、「 <a href="#">AWS に問い合わせる</a> 」を参照してください。 | 403 Forbidden   | クライアント             |
| AllAccessDisabled             | この Amazon S3 リソースへのすべてのアクセスは無効になっています。詳細については、「 <a href="#">AWS に問い合わせる</a> 」を参照してください。              | 403 Forbidden   | クライアント             |
| AmbiguousGrantByEmailAddress  | 指定した E メールアドレスが複数のアカウントに関連付けられています。                                                                  | 400 Bad Request | クライアント             |

| エラーコード                            | 説明                                                                              | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|-----------------------------------|---------------------------------------------------------------------------------|-----------------|--------------------|
| AuthorizationHeaderMalformed      | 指定した認可ヘッダーが無効です。                                                                | 400 Bad Request | 該当なし               |
| AuthorizationQueryParametersError | 指定した認可クエリパラメータが無効です。                                                            | 400 Bad Request | 該当なし               |
| BadDigest                         | 指定した Content-MD5 またはチェックサム値が、サーバーで受信した値と一致しませんでした。                              | 400 Bad Request | クライアント             |
| BucketAlreadyExists               | リクエストされたバケット名は使用できません。バケット名前空間は、システムのすべてのユーザーによって共有されています。別の名前を使用してもう一度試してください。 | 409 Conflict    | クライアント             |

| エラーコード                        | 説明                                                                                                                                                                                                                                                                                                                | HTTP ステータスコード                         | SOAP 障害コードのプレフィックス |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|--------------------|
| BucketAlreadyOwnedByYou       | <p>作成しようとしたバケットは既に存在しており、そのバケットの所有者はあなたです。Amazon S3 は、米国東部 (バージニア北部) リージョン (us-east-1) を除くすべての AWS リージョンでこのエラーを返します。レガシー互換性のために、us-east-1 で既に所有している既存のバケットを再作成すると、Amazon S3 は 200 OK を返し、バケットアクセスコントロールリスト (ACL) をリセットします。</p> <p>Amazon S3 on Outposts の場合、作成しようとしたバケットは Outpost に既に存在しており、そのバケットの所有者はあなたです。</p> | 409 Conflict (us-east-1 を除くすべてのリージョン) | クライアント             |
| BucketHasAccessPointsAttached | 削除しようとしたバケットには、アクセスポイントがアタッチされています。バケットを削除する前にアクセスポイントを削除します。                                                                                                                                                                                                                                                     | 400 Bad Request                       | クライアント             |
| BucketNotEmpty                | 削除しようとしたバケットは空ではありません。                                                                                                                                                                                                                                                                                            | 409 Conflict                          | クライアント             |

| エラーコード                         | 説明                                                                                                                               | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------|
| ClientTokenConflict            | マルチリージョンアクセスポイントのべき等性トークンが別のリクエストに既に使用されています。                                                                                    | 409 Conflict    | クライアント             |
| ConnectionClosedByRequester    | WriteGetObjectResponse 本文の読み取り中にエラーが発生した場合に、呼び出し元に返されます。                                                                         | 400 Bad Request | クライアント             |
| ConditionalRequestConflict     | 競合するオペレーションが発生しました。PutObject を使用すると、リクエストを再試行できます。マルチパートアップロードを使用する場合は、別の CreateMultipartUpload リクエストを開始し、各パートを再アップロードする必要があります。 | 409 Conflict    | クライアント             |
| CredentialsNotSupported        | このリクエストは認証情報をサポートしていません。                                                                                                         | 400 Bad Request | クライアント             |
| CrossLocationLoggingProhibited | クロスリージョンのログ記録は許可されていません。1 つの AWS リージョンのバケットは、別のリージョンのバケットに情報をログ記録できません。                                                          | 403 Forbidden   | クライアント             |

| エラーコード               | 説明                                                                                                                                | HTTP ステータスコード          | SOAP 障害コードのプレフィックス |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------|------------------------|--------------------|
| DeviceNotActiveError | デバイスは現在アクティブではありません。                                                                                                              | 400 Bad Request        | クライアント             |
| EndpointNotFound     | リクエストを正しいエンドポイントに直接送信してください。                                                                                                      | 400 Bad Request        | クライアント             |
| EntityTooSmall       | 提案されたアップロードが最小許容オブジェクトサイズより小さくなっています。                                                                                             | 400 Bad Request        | クライアント             |
| EntityTooLarge       | 提案されたアップロードが最大許容オブジェクトサイズを超えています。詳細については、「AWS 全般のリファレンス」の「 <a href="#">Amazon Simple Storage Service エンドポイントとクォータ</a> 」を参照してください。 | 400 Bad Request        | クライアント             |
|                      | 提案されたダウンロードが最大許容サイズを超えています。                                                                                                       | 405 Method Not Allowed | クライアント             |
| ExpiredToken         | 提供されたトークンの有効期限が切れています。                                                                                                            | 400 Bad Request        | クライアント             |

| エラーコード                                  | 説明                                                                                                                                                                                                                | HTTP ステータスコード      | SOAP 障害コードのプレフィックス |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|--------------------|
| IllegalLocationConstraintException      | <p>このエラーは、次の原因によって発生する可能性があります。</p> <ul style="list-style-type: none"><li>• バケットが存在するリージョンとは異なるリージョンからバケットにアクセスしようとしています。</li><li>• リクエストが送信されたリージョンのエンドポイントとは異なるリージョンに対応するロケーション制約があるバケットを作成しようとしています。</li></ul> | 400<br>Bad Request | クライアント             |
| IllegalVersioningConfigurationException | リクエストで指定したバージョンニング設定が無効です。                                                                                                                                                                                        | 400<br>Bad Request | クライアント             |
| IncompleteBody                          | Content-Length HTTP ヘッダーで指定されたバイト数が提供されていません。                                                                                                                                                                     | 400<br>Bad Request | クライアント             |
| IncorrectEndpoint                       | 指定されたバケットが別のリージョンに存在します。リクエストを正しいエンドポイントに直接送信してください。                                                                                                                                                              | 400<br>Bad Request | クライアント             |



| エラーコード                              | 説明                                      | HTTP ステータスコード             | SOAP 障害コードのプレフィックス |
|-------------------------------------|-----------------------------------------|---------------------------|--------------------|
| IncorrectNumberOfFilesInPostRequest | POST では、リクエストごとに 1 つのファイルアップロードのみが必要です。 | 400 Bad Request           | クライアント             |
| InlineDataTooLarge                  | インラインデータが最大許容サイズを超えています。                | 400 Bad Request           | クライアント             |
| InternalError                       | 内部エラーが発生しました。もう一度試してください。               | 500 Internal Server Error | の                  |
| InvalidAccessKeyId                  | 指定された AWS アクセスキー ID が見つかりません。           | 403 Forbidden             | クライアント             |
| InvalidAccessPoint                  | 指定されたアクセスポイント名またはアカウントが無効です。            | 400 Bad Request           | クライアント             |
| InvalidAccessPointAliasError        | 指定されたアクセスポイントエイリアス名が無効です。               | 400 Bad Request           | クライアント             |
| InvalidAddressingHeader             | 匿名ロールを指定する必要があります。                      | 該当なし                      | クライアント             |

| エラーコード                              | 説明                                                                                                                                                                                                                                                                                                                 | HTTP ステータスコード         | SOAP 障害コードのプレフィックス |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|--------------------|
| InvalidArgument                     | <p>このエラーは、次の原因によって発生する可能性があります。</p> <ul style="list-style-type: none"><li>• bucket-region パラメータで指定したリージョンとは異なるリージョンエンドポイントに ListBuckets リクエストが行なわれた場合。</li><li>• 指定された引数は有効ではありませんでした。</li><li>• リクエストに必要なヘッダーがありませんでした。</li><li>• 指定された引数が不完全であるか、形式が正しくありませんでした。</li><li>• 指定された引数は、3 以上の長さである必要があります。</li></ul> | 400<br>Bad<br>Request | クライアント             |
| InvalidBucketAclWithObjectOwnership | バケットでは、ObjectOwnership の BucketOwnerEnforced 設定で ACL を設定することはできません。                                                                                                                                                                                                                                                | 400<br>Bad<br>Request | クライアント             |

| エラーコード                          | 説明                                            | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|---------------------------------|-----------------------------------------------|-----------------|--------------------|
| InvalidBucketName               | 指定されたバケットが無効です。                               | 400 Bad Request | クライアント             |
| InvalidBucketOwnerAWSAccountID  | 期待されるバケット所有者パラメータの値は AWS アカウント ID である必要があります。 | 400 Bad Request | クライアント             |
| InvalidBucketState              | リクエストはバケットの現在の状態で有効ではありません。                   | 409 Conflict    | クライアント             |
| InvalidDigest                   | 指定した Content-MD5 またはチェックサム値が無効です。             | 400 Bad Request | クライアント             |
| InvalidEncryptionAlgorithmError | 指定した暗号化リクエストが無効です。有効値は AES256 です。             | 400 Bad Request | クライアント             |
| InvalidHostHeader               | リクエストで指定されたホストヘッダーに適正でない形式のアドレス指定が使用されていました。  | 400 Bad Request | クライアント             |
| InvalidHttpMethod               | リクエストが、予期しない HTTP メソッドを使用して実行されました。           | 400 Bad Request | クライアント             |

| エラーコード                    | 説明                                                                                      | HTTP ステータスコード      | SOAP 障害コードのプレフィックス |
|---------------------------|-----------------------------------------------------------------------------------------|--------------------|--------------------|
| InvalidLocationConstraint | 指定された場所 (リージョン) の制約が無効です。バケットのリージョンの選択の詳細については、「 <a href="#">バケットの概要</a> 」を参照してください。    | 400<br>Bad Request | クライアント             |
| InvalidObjectState        | 現在のオブジェクトの状態では、オペレーションは無効です。                                                            | 403<br>Forbidden   | クライアント             |
| InvalidPart               | 指定されたパートが 1 つ以上見つかりませんでした。パートがアップロードされていないか、指定されたエンティティタグがパートのエンティティタグと一致していない可能性があります。 | 400<br>Bad Request | クライアント             |
| InvalidPartOrder          | パートのリストが昇順ではありません。パートのリストはパートの番号順に指定する必要があります。                                          | 400<br>Bad Request | クライアント             |
| InvalidPayer              | このオブジェクトへのすべてのアクセスが無効になっています。詳細については、「 <a href="#">AWS に問い合わせる</a> 」を参照してください。          | 403<br>Forbidden   | クライアント             |

| エラーコード                | 説明                                            | HTTP ステータスコード                       | SOAP 障害コードのプレフィックス |
|-----------------------|-----------------------------------------------|-------------------------------------|--------------------|
| InvalidPolicyDocument | フォームの内容がポリシードキュメントで指定された条件を満たしていません。          | 400 Bad Request                     | クライアント             |
| InvalidRange          | リクエストされた範囲を満たすことができません。                       | 416 Requested Range Not Satisfiable | クライアント             |
| InvalidRegion         | オプトインしていないリージョンにマルチリージョンアクセスポイントを作成しようとしています。 | 403 Forbidden                       | クライアント             |

| エラーコード         | 説明                                                                                                                                                                                                                                                                                                                                                                                                | HTTP ステータスコード      | SOAP 障害コードのプレフィックス |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|--------------------|
| InvalidRequest | <p>このエラーは、次の原因によって発生する可能性があります。</p> <ul style="list-style-type: none"><li>• ページ分割されていない<br/>ListBuckets リクエストは、承認された汎用バケットクォータが 10,000 を超えるアカウントから行われます。ページ分割されたリクエストを実行して、10,000 を超えるバケットを持つアカウントのバケットを一覧表示する必要があります。</li><li>• リクエストで間違った署名バージョンを使用しています。AWS4-HMAC-SHA256（署名バージョン 4）を使用してください。</li><li>• アクセスポイントは、既存のバケットに対してのみ作成できます。</li><li>• アクセスポイントが削除可能な状態ではありません。</li><li>• </li></ul> | 400<br>Bad Request | クライアント             |

| エラーコード | 説明                                                                                                                                                                                                                                                                                                                                                                                                                       | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------------------|
|        | <p>アクセスポイントは、既存のバケットに対してのみ作成できます。</p> <ul style="list-style-type: none"><li>• 次のトークンが無効です。</li><li>• ライフサイクルルールで少なくとも 1 つのアクションを指定する必要があります。</li><li>• 少なくとも 1 つのライフサイクルルールを指定する必要があります。</li><li>• ライフサイクルルールの数は、許可されている 1000 ルールの制限を超えてはなりません。</li><li>• MaxResults パラメータの範囲が無効です。</li><li>• SOAP リクエストは HTTPS 接続を介して行う必要があります。</li><li>• Amazon S3 Transfer Acceleration は、DNS に準拠していない名前のバケットではサポートされていません。</li></ul> |               |                    |

| エラーコード | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------------------|
|        | <ul style="list-style-type: none"><li>Amazon S3 Transfer Acceleration は、ピリオド (.) が名前に含まれているバケットではサポートされていません。</li><li>Amazon S3 Transfer Acceleration エンドポイントは、仮想スタイルのリクエストのみをサポートします。</li><li>Amazon S3 Transfer Acceleration は、このバケットでは設定されていません。</li><li>Amazon S3 Transfer Acceleration は、このバケットでは無効になっています。</li><li>Amazon S3 Transfer Acceleration は、このバケットではサポートされていません。サポートが必要な場合は、<a href="#">サポート</a> にお問い合わせください。</li><li>Amazon S3 Transfer Acceleration はこのバケットでは有効にできません。サポートが必要な場合</li></ul> |               |                    |



| エラーコード                  | 説明                                                                                                                                                                                                                                          | HTTP ステータスコード         | SOAP 障害コードのプレフィックス |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|--------------------|
|                         | <p>は、<a href="#">サポート</a> にお問い合わせください。</p> <ul style="list-style-type: none"><li>• HTTP ヘッダーとクエリパラメータで指定された値が競合しています。</li><li>• HTTP ヘッダーと POST フォームフィールドで指定された値が競合しています。</li><li>• サイズが 5GB を超えるオブジェクトに対して CopyObject 要求が行われました。</li></ul> |                       |                    |
| InvalidSessionException | タイムアウトしたか有効期限が切れたために、セッションがこれ以上存在しなくなった場合に返されます。                                                                                                                                                                                            | 400<br>Bad<br>Request | クライアント             |
| InvalidSignature        | 計算したリクエスト署名が、指定された署名と一致しません。AWS シークレットアクセスキーと署名方法を確認してください。詳細については、「 <a href="#">REST リクエストの署名と認証</a> 」を参照してください。                                                                                                                           | 400<br>Bad<br>Request | クライアント             |

| エラーコード                        | 説明                                                             | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|-------------------------------|----------------------------------------------------------------|-----------------|--------------------|
| InvalidSecurity               | 提供されたセキュリティ認証情報が無効です。                                          | 403 Forbidden   | クライアント             |
| InvalidSOAPRequest            | SOAP リクエスト本文が無効です。                                             | 400 Bad Request | クライアント             |
| InvalidStorageClass           | 指定されたストレージクラスが無効です。                                            | 400 Bad Request | クライアント             |
| InvalidTargetBucketForLogging | ログ記録用のターゲットバケットが存在しないか、ユーザーが所有者ではないか、ログ配信グループに適切なアクセス許可がありません。 | 400 Bad Request | クライアント             |
| InvalidToken                  | 指定されたトークンは形式に誤りがあるか無効です。                                       | 400 Bad Request | クライアント             |
| InvalidURI                    | 指定された URI を解析できませんでした。                                         | 400 Bad Request | クライアント             |
| KeyTooLongError               | キーが長すぎます。                                                      | 400 Bad Request | クライアント             |

| エラーコード                | 説明                                      | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|-----------------------|-----------------------------------------|-----------------|--------------------|
| KMS.DisabledException | 指定された KMS キーが有効になっていないため、リクエストが拒否されました。 | 400 Bad Request | クライアント             |

| エラーコード                       | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | HTTP ステータスコード         | SOAP 障害コードのプレフィックス |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|--------------------|
| KMS.InvalidKeyUsageException | <p>次のいずれかの理由でリクエストが拒否されました。</p> <ul style="list-style-type: none"><li>• KMS キーの KeyUsage 値が API オペレーションと互換性がありません。</li><li>• オペレーションで指定された暗号化アルゴリズムまたは署名アルゴリズムが、KMS キーのキーマテリアルタイプ (KeySpec) と互換性がありません。</li></ul> <p>データキーの暗号化、復号、再暗号化、生成を行うには、KeyUsage が ENCRYPT_DECRYPT である必要があります。メッセージに署名して検証するには、KeyUsage が SIGN_VERIFY である必要があります。メッセージ認証コード (MAC) を生成して検証するには、KeyUsage が GENERATE_VERIFY_MAC である必要があります。キーアグリーメントシークレットを取得するには、KeyUsage が KEY_AGREEMENT である必要があります。KMS キーの KeyUsage を検索するには、D</p> | 400<br>Bad<br>Request | クライアント             |

| エラーコード | 説明                                                                                                                 | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|--------|--------------------------------------------------------------------------------------------------------------------|---------------|--------------------|
|        | <p>escribeKey オペレーションを使用します。</p> <p>特定の KMS キーでサポートされている暗号化アルゴリズムまたは署名アルゴリズムを検索するには、DescribeKey オペレーションを使用します。</p> |               |                    |

| エラーコード                       | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | HTTP ステータスコード      | SOAP 障害コードのプレフィックス |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|--------------------|
| KMS.KMSInvalidStateException | <p>指定されたリソースの状態がこのリクエストに対して有効ではないため、リクエストが拒否されました。この例外は、次のいずれかを意味します。</p> <ul style="list-style-type: none"><li>• KMS キーのキーの状態は、このオペレーションと互換性がありません。</li></ul> <p>キーの状態を検索するには、DescribeKey オペレーションを使用します。各 KMS オペレーションと互換性があるキーの状態の詳細については、「AWS Key Management Service デベロッパーガイド」の「<a href="#">AWS KMS keys のキーの状態</a>」を参照してください。</p> <ul style="list-style-type: none"><li>• カスタムキーストアの KMS キーへの暗号化オペレーションの場合、この例外は、多くの原因が想定できる、一般的な障害を意味します。原因を特定するには、例外に付随するエラーメッセージを使用します。</li></ul> | 400<br>Bad Request | クライアント             |

| エラーコード                            | 説明                                                | HTTP ステータスコード      | SOAP 障害コードのプレフィックス |
|-----------------------------------|---------------------------------------------------|--------------------|--------------------|
| KMS.NotFoundException             | 指定されたエンティティまたはリソースが見つからないため、リクエストが拒否されました。        | 400<br>Bad Request | クライアント             |
| MalformedACLError                 | 指定された ACL の書式に誤りがあるか、AWS の公開済みスキーマに一致しませんでした。     | 400<br>Bad Request | クライアント             |
| MalformedPOSTRequest              | POST リクエストの本文が正しい形式の multipart/form-data ではありません。 | 400<br>Bad Request | クライアント             |
| MalformedXML                      | 指定された XML の書式に誤りがあるか、Amazon の公開済みスキーマに一致しませんでした。  | 400<br>Bad Request | クライアント             |
| MaxMessageLengthExceeded          | リクエストが大きすぎます。                                     | 400<br>Bad Request | クライアント             |
| MaxPostPreDataLengthExceededError | アップロードファイルの前の POST リクエストフィールドが大きすぎます。             | 400<br>Bad Request | クライアント             |
| MetadataTooLarge                  | メタデータヘッダーが許容されるメタデータの最大サイズを超えています。                | 400<br>Bad Request | クライアント             |

| エラーコード                     | 説明                                      | HTTP ステータスコード          | SOAP 障害コードのプレフィックス |
|----------------------------|-----------------------------------------|------------------------|--------------------|
| MethodNotAllowed           | 指定されたメソッドは、このリソースに対して許可されていません。         | 405 Method Not Allowed | クライアント             |
| MissingAttachment          | SOAP アタッチメントが想定されていたにもかかわらず、見つかりませんでした。 | 400 Bad Request        | クライアント             |
| MissingAuthenticationToken | リクエストに署名がありませんでした。                      | 403 Forbidden          | クライアント             |
| MissingContentLength       | Content-Length HTTP ヘッダーを指定する必要があります。   | 411 Length Required    | クライアント             |
| MissingRequestBodyError    | 空の XML ドキュメントをリクエストとして送信しました。           | 400 Bad Request        | クライアント             |
| MissingSecurityElement     | SOAP 1.1 リクエストにセキュリティ要素が含まれていません。       | 400 Bad Request        | クライアント             |
| MissingSecurityHeader      | リクエストに必要なヘッダーがありません。                    | 400 Bad Request        | クライアント             |



| エラーコード                       | 説明                               | HTTP ステータスコード      | SOAP 障害コードのプレフィックス |
|------------------------------|----------------------------------|--------------------|--------------------|
| NoLoggingStatusForKey        | キーのログ記録ステータスサブリソースのようなものではありません。 | 400<br>Bad Request | クライアント             |
| NoSuchAsyncRequest           | 指定されたリソースは見つかりませんでした。            | 404<br>Not Found   | クライアント             |
| NoSuchBucket                 | 指定されたバケットが存在しません。                | 404<br>Not Found   | クライアント             |
| NoSuchBucketPolicy           | 指定されたバケットにバケットポリシーが設定されていません。    | 404<br>Not Found   | クライアント             |
| NoSuchCORSConfiguration      | 指定されたバケットには CORS 設定がありません。       | 404<br>Not Found   | クライアント             |
| NoSuchKey                    | 指定されたキーが存在しません。                  | 404<br>Not Found   | クライアント             |
| NoSuchLifecycleConfiguration | 指定されたライフサイクル設定が存在しません。           | 404<br>Not Found   | クライアント             |

| エラーコード                        | 説明                                                                         | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|-------------------------------|----------------------------------------------------------------------------|-----------------|--------------------|
| NoSuchMultiRegionAccessPoint  | 指定されたマルチリージョンアクセスポイントが存在しません。                                              | 404 Not Found   | クライアント             |
| NoSuchObjectLockConfiguration | 指定されたオブジェクトには ObjectLock 設定がありません。                                         | 404 Not Found   | クライアント             |
| NoSuchWebsiteConfiguration    | 指定されたバケットにはウェブサイト設定がありません。                                                 | 404 Not Found   | クライアント             |
| NoSuchTagSet                  | 指定されたタグが存在しません。                                                            | 404 Not Found   | クライアント             |
| NoSuchUpload                  | 指定されたマルチパートアップロードは存在しません。アップロード ID が無効であるか、マルチパートアップロードが中止または完了した可能性があります。 | 404 Not Found   | クライアント             |
| NoSuchVersion                 | リクエストで指定されたバージョン ID が既存のバージョンと一致しません。                                      | 404 Not Found   | クライアント             |
| NotDeviceOwnerError           | トークンを生成したデバイスを、認証済みのユーザーが所有していません。                                         | 400 Bad Request | クライアント             |

| エラーコード                               | 説明                                                                                                                                                         | HTTP ステータスコード       | SOAP 障害コードのプレフィックス |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|--------------------|
| NotImplemented                       | 指定されたヘッダーは、実装されていない機能を意味します。                                                                                                                               | 501 Not Implemented | の                  |
| NotModified                          | リソースは変更されませんでした。                                                                                                                                           | 304 Not Modified    | の                  |
| NoTransformationDefined              | この Object Lambda アクセスポイントの変換が見つかりませんでした。                                                                                                                   | 404 Not Found       | クライアント             |
| NotSignedUp                          | アカウントが Amazon S3 に対してサインアップされていません。Amazon S3 を使用する前に、サインアップする必要があります。サインアップは <a href="https://aws.amazon.com/s3">https://aws.amazon.com/s3</a> で行うことができます。 | 403 Forbidden       | クライアント             |
| ObjectLockConfigurationNotFoundError | このバケットにはオブジェクトロック設定が存在しません。                                                                                                                                | 404 Not Found       | クライアント             |
| OwnershipControlsNotFound            | バケットの所有権コントロールが見つかりませんでした。                                                                                                                                 | 404 Not Found       | クライアント             |

| エラーコード                        | 説明                                                                                 | HTTP ステータスコード           | SOAP 障害コードのプレフィックス |
|-------------------------------|------------------------------------------------------------------------------------|-------------------------|--------------------|
| OperationAborted              | 競合する条件付きオペレーションが現在このリソースに対して進行中です。もう一度試してください。                                     | 409 Conflict            | クライアント             |
| PermanentRedirect             | アクセスしようとしているバケットは、指定されたエンドポイントを使用してアドレス指定する必要があります。今後のリクエストはこのエンドポイントに送信してください。    | 301 Moved Permanently   | クライアント             |
| PermanentRedirectControlError | アクセスしようとしている API 操作は、指定されたエンドポイントを使用してアドレス指定する必要があります。今後のリクエストはこのエンドポイントに送信してください。 | 301 Moved Permanently   | クライアント             |
| PreconditionFailed            | 指定した前提条件の少なくとも 1 つが保持されませんでした。                                                     | 412 Precondition Failed | クライアント             |
| Redirect                      | 一時的なリダイレクト。ドメインネームシステム (DNS) サーバーが更新されている間、このバケットにリダイレクトされています。                    | 307 Temporary Redirect  | クライアント             |

| エラーコード                       | 説明                                                                                    | HTTP ステータスコード           | SOAP 障害コードのプレフィックス |
|------------------------------|---------------------------------------------------------------------------------------|-------------------------|--------------------|
| RequestHeaderSectionTooLarge | リクエストの実行に使用されたリクエストヘッダーとクエリパラメータが最大許容サイズを超えています。                                      | 400 Bad Request         | クライアント             |
| RequestIsNotMultipartContent | バケット POST リクエストは、 <code>enctype</code> が <code>multipart/form-data</code> である必要があります。 | 412 Precondition Failed | クライアント             |
| RequestTimeout               | サーバーへのソケット接続がタイムアウト期間内に読み書きされませんでした。                                                  | 400 Bad Request         | クライアント             |
| RequestTimeTooSkewed         | リクエスト時間とサーバー時間の差が大きすぎます。                                                              | 403 Forbidden           | クライアント             |
| RequestTorrentOfBucketError  | バケットの Torrent ファイルのリクエストは許可されていません。                                                   | 400 Bad Request         | クライアント             |
| ResponseInterrupted          | <code>WriteGetObjectResponse</code> 本文の読み取り中にエラーが発生した場合に、呼び出し元に返されます。                 | 400 Bad Request         | クライアント             |
| RestoreAlreadyInProgress     | オブジェクトの復元は既に進行中です。                                                                    | 409 Conflict            | クライアント             |

| エラーコード                                    | 説明                                                                                                                                                      | HTTP ステータスコード           | SOAP 障害コードのプレフィックス |
|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|--------------------|
| ServerSideEncryptionConfigurationNotFound | サーバー側の暗号化設定が見つかりませんでした。                                                                                                                                 | 400 Bad Request         | クライアント             |
| ServiceUnavailable                        | サービスでリクエストを処理できません。                                                                                                                                     | 503 Service Unavailable | の                  |
| SignatureDoesNotMatch                     | 計算したリクエスト署名が、指定された署名と一致しません。AWS シークレットアクセスキーと署名方法を確認してください。詳細については、「 <a href="#">REST リクエストの署名と認証</a> 」および「 <a href="#">SOAP リクエストの認証方法</a> 」を参照してください。 | 403 Forbidden           | クライアント             |
| SlowDown                                  | リクエストレートを下げてください。                                                                                                                                       | 503 Slow Down           | の                  |
| 503 SlowDown                              | 減速してください。                                                                                                                                               | 503 Slow Down           | の                  |

| エラーコード                | 説明                                                                                                                                    | HTTP ステータスコード          | SOAP 障害コードのプレフィックス |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------|------------------------|--------------------|
| TagPolicyException    | タグポリシーでは、次のタグキーに指定された値は許可されていません。                                                                                                     | 400 Bad Request        | クライアント             |
| TemporaryRedirect     | ドメインネームシステム (DNS) サーバーが更新されている間、このバケットにリダイレクトされています。                                                                                  | 307 Temporary Redirect | クライアント             |
| TokenCodeInvalidError | 指定したシリアル番号および/またはトークンコードが無効です。                                                                                                        | 400 Bad Request        | クライアント             |
| TokenRefreshRequired  | 提供されたトークンを更新する必要があります。                                                                                                                | 400 Bad Request        | クライアント             |
| TooManyAccessPoints   | アカウントで許可されているよりも多くのアクセスポイントを作成しようとした。詳細については、「AWS 全般のリファレンス」の「 <a href="#">Amazon Simple Storage Service エンドポイントとクォータ</a> 」を参照してください。 | 400 Bad Request        | クライアント             |

| エラーコード                                        | 説明                                                                                                                                                    | HTTP ステータスコード      | SOAP 障害コードのプレフィックス |
|-----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|--------------------|
| TooManyBuckets                                | アカウントで許可されているよりも多くのバケットを作成しようとした。詳細については、「AWS 全般のリファレンス」の「 <a href="#">Amazon Simple Storage Service エンドポイントとクォータ</a> 」を参照してください。                     | 400<br>Bad Request | クライアント             |
| TooManyMultiRegion<br>AccessPointregionsError | アカウントで許可されているよりも多くのリージョンを持つマルチリージョンアクセスポイントを作成しようとした。詳細については、「AWS 全般のリファレンス」の「 <a href="#">Amazon Simple Storage Service エンドポイントとクォータ</a> 」を参照してください。 | 400<br>Bad Request | クライアント             |
| TooManyMultiRegion<br>AccessPoints            | アカウントで許可されているよりも多くのマルチリージョンアクセスポイントを作成しようとした。詳細については、「AWS 全般のリファレンス」の「 <a href="#">Amazon Simple Storage Service エンドポイントとクォータ</a> 」を参照してください。         | 400<br>Bad Request | クライアント             |



| エラーコード                          | 説明                                                                                                           | HTTP ステータスコード      | SOAP 障害コードのプレフィックス |
|---------------------------------|--------------------------------------------------------------------------------------------------------------|--------------------|--------------------|
| UnauthorizedAccessError         | 中国リージョンにのみ適用されます。ICP ライセンスを持たないバケットに対してリクエストが行われたときに返されます。詳細については、「 <a href="#">ICP Recordal</a> 」を参照してください。 | 403<br>Forbidden   | クライアント             |
| UnexpectedContent               | This request contains unsupported content.                                                                   | 400<br>Bad Request | クライアント             |
| UnexpectedIPError               | 中国リージョンにのみ適用されます。予期されない IP であったため、このリクエストは拒否されました。                                                           | 403<br>Forbidden   | クライアント             |
| UnsupportedArgument             | リクエストにサポートされていない引数が含まれていました。                                                                                 | 400<br>Bad Request | クライアント             |
| UnsupportedSignature            | 提供されたリクエストは、サポートされていない STS トークンのバージョンで署名されているか、署名のバージョンがサポートされていません。                                         | 400<br>Bad Request | クライアント             |
| UnresolvableGrantByEmailAddress | 指定した E メールアドレスが、登録されているアカウントと一致しません。                                                                         | 400<br>Bad Request | クライアント             |

| エラーコード                 | 説明                                                                          | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|------------------------|-----------------------------------------------------------------------------|-----------------|--------------------|
| UserKeyMustBeSpecified | バケット POST リクエストは、指定されたフィールド名を含んでいなければなりません。指定されている場合は、フィールドの順序を確認してください。    | 400 Bad Request | クライアント             |
| NoSuchAccessPoint      | 指定されたアクセスポイントが存在しません。                                                       | 404 Not Found   | クライアント             |
| InvalidTag             | リクエストに無効なタグ入力が含まれています。例えば、リクエストに重複するキー、長すぎるキーまたは値、またはシステムタグが含まれている可能性があります。 | 400 Bad Request | クライアント             |
| MalformedPolicy        | ポリシーに無効なプリンシパルが含まれています。                                                     | 400 Bad Request | クライアント             |

## SELECT Object Content エラーコードのリスト

### Important

新規のお客様への Amazon S3 Select の提供は終了しました。Amazon S3 Select をご利用の既存のお客様は、今後も通常どおり使用できます。[詳細情報](#)

次の表に、SELECT Object Content が返す可能性のある特殊なエラーを示します。Amazon S3 のエラーとエラーコードのリストの一般的な情報については、「[エラーレスポンス](#)」を参照してください。

| エラーコード                     | 説明                                                                                                  | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|----------------------------|-----------------------------------------------------------------------------------------------------|---------------|--------------------|
| AmbiguousFieldName         | フィールド名がファイル内の複数のフィールドと一致しています。SQL 式とファイルを確認して、もう一度試してください。                                          | 400           | クライアント             |
| Busy                       | サービスは利用できません。後でもう一度お試しください。                                                                         | 503           | クライアント             |
| CastFailed                 | SQL 式で CAST を使用して、あるデータ型から別のデータ型に変換しようとしたが、失敗しました。                                                  | 400           | クライアント             |
| ColumnTooLong              | 結果の列の長さが 1 MB の maxCharsPerColumn を超えています。                                                          | 400           | クライアント             |
| CSVEscapingRecordDelimiter | 引用符で囲まれたレコード区切り文字がファイルで見つかりました。引用符で囲まれたレコード区切り文字を許可するには、AllowQuotedRecordDelimiter を 'TRUE' に設定します。 | 400           | クライアント             |
| CSVParsingError            | CSV ファイルの解析中にエラーが発生しました。ファイルを確認して、もう一度試してください。                                                      | 400           | クライアント             |

| エラーコード                                                 | 説明                                                                                                       | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|--------------------------------------------------------|----------------------------------------------------------------------------------------------------------|---------------|--------------------|
| CSVUnescapedQuote                                      | CSV ファイルの解析中にエスケープされていない引用符が見つかりました。引用符で囲まれたレコード区切り文字を許可するには、AllowQuotedRecordDelimiter を 'TRUE' に設定します。 | 400           | クライアント             |
| EmptyRequestBody                                       | リクエスト本文を空にすることはできません。                                                                                    | 400           | クライアント             |
| EvaluatorBindingDoesNotExist                           | 列名または指定されたパスが SQL 式に存在しません。                                                                              | 400           | クライアント             |
| EvaluatorInvalidArguments                              | SQL 式の関数呼び出しの引数の数が正しくありません。                                                                              | 400           | クライアント             |
| EvaluatorInvalidTimestampFormatPattern                 | SQL 式のタイムスタンプ形式の文字列が無効です。                                                                                | 400           | クライアント             |
| EvaluatorInvalidTimestampFormatPatternSymbol           | SQL 式で指定したタイムスタンプの形式パターンに、無効な記号が含まれています。                                                                 | 400           | クライアント             |
| EvaluatorInvalidTimestampFormatPatternSymbolForParsing | SQL 式で指定したタイムスタンプの形式パターンに、タイムスタンプ解析に適用できない有効な書式記号が含まれています。                                               | 400           | クライアント             |
| EvaluatorInvalidTimestampFormatPatternToken            | SQL 式で指定したタイムスタンプの形式パターンに、無効なトークンが含まれています。                                                               | 400           | クライアント             |

| エラーコード                                               | 説明                                                                                                      | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|------------------------------------------------------|---------------------------------------------------------------------------------------------------------|---------------|--------------------|
| EvaluatorLikePatternInvalidEscapeSequence            | LIKE 式で指定した引数が無効でした。                                                                                    | 400           | クライアント             |
| EvaluatorNegativeLimit                               | LIMIT を負の値にすることはできません。                                                                                  | 400           | クライアント             |
| EvaluatorTimestampFormatPatternDuplicateFields       | SQL 式で指定したタイムスタンプの形式パターンに、タイムスタンプフィールドを表す複数の書式指定子が含まれています。                                              | 400           | クライアント             |
| EvaluatorTimestampFormatPatternHourClockAmPmMismatch | SQL 式で指定したタイムスタンプの形式パターンに、12 時間形式の記号が含まれているのに AM/PM フィールドが含まれていないか、24 時間形式の指定があるのに AM/PM フィールドが含まれています。 | 400           | クライアント             |
| EvaluatorUnterminatedTimestampFormatPatternToken     | SQL 式で指定したタイムスタンプの形式パターンに、終了されていないトークンが含まれています。                                                         | 400           | クライアント             |
| ExpressionTooLong                                    | SQL 式が長すぎます。SQL 式の最大バイト長は 256 KB です。                                                                    | 400           | クライアント             |
| ExternalEvalException                                | クエリは評価できません。ファイルを確認して、もう一度試してください。                                                                      | 400           | クライアント             |
| IllegalSqlFunctionArgument                           | SQL 関数で不正な引数を使用されました。                                                                                   | 400           | クライアント             |

| エラーコード                           | 説明                                                    | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|----------------------------------|-------------------------------------------------------|---------------|--------------------|
| IncorrectSqlFunctionArgumentType | SQL 式の関数呼び出しで誤った引数タイプが指定されました。                        | 400           | クライアント             |
| IntegerOverflow                  | SQL 式で整数オーバーフローまたはアンダーフローが発生しました。                     | 400           | クライアント             |
| InternalError                    | 内部エラーが発生しました。                                         | 500           | クライアント             |
| InvalidCast                      | SQL 式で CAST を使用して、あるデータ型から別のデータ型に変換しようとしたましたが、失敗しました。 | 400           | クライアント             |
| InvalidColumnIndex               | SQL 式の列インデックスが無効です。                                   | 400           | クライアント             |
| InvalidCompressionFormat         | ファイルがサポートされている圧縮形式ではありません。GZIP と BZIP2 のみがサポートされています。 | 400           | クライアント             |
| InvalidDataSource                | データソースタイプが無効です。CSV、JSON、Parquet のみがサポートされています。        | 400           | クライアント             |
| InvalidDataType                  | SQL 式に無効なデータ型が含まれています。                                | 400           | クライアント             |
| InvalidExpressionType            | ExpressionType 値は無効です。SQL 式のみがサポートされています。             | 400           | クライアント             |

| エラーコード                  | 説明                                                               | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|-------------------------|------------------------------------------------------------------|---------------|--------------------|
| InvalidFileHeaderInfo   | FileHeaderInfo 値は無効です。NONE、USE、IGNORE のみがサポートされています。             | 400           | クライアント             |
| InvalidJsonType         | JsonType 値は無効です。DOCUMENT と LINES のみがサポートされています。                  | 400           | クライアント             |
| InvalidKeyPath          | SQL 式のキーパスが無効です。                                                 | 400           | クライアント             |
| InvalidQuoteFields      | QuoteFields 値は無効です。ALWAYS と ASNEEDED のみがサポートされています。              | 400           | クライアント             |
| InvalidRequestParameter | SelectRequest 要素のパラメータの値が無効です。サービスの API ドキュメントを確認して、もう一度試してください。 | 400           | クライアント             |
| InvalidScanRange        | 指定されたスキャン範囲が無効です。                                                | 400           | クライアント             |
| InvalidTableAlias       | SQL 式に無効なテーブルエイリアスが含まれています。                                      | 400           | クライアント             |
| InvalidTextEncoding     | エンコードタイプが無効です。UTF-8 エンコーディングのみがサポートされています。                       | 400           | クライアント             |
| JSONParsingError        | JSON ファイルの解析中にエラーが発生しました。ファイルを確認して、もう一度試してください。                  | 400           | クライアント             |

| エラーコード                 | 説明                                                                         | HTTP ステータスコード          | SOAP 障害コードのプレフィックス |
|------------------------|----------------------------------------------------------------------------|------------------------|--------------------|
| LexerInvalidChar       | SQL 式に無効な文字が含まれています。                                                       | 400                    | クライアント             |
| LexerInvalidIONLiteral | SQL 式に無効な演算子が含まれています。                                                      | 400                    | クライアント             |
| LexerInvalidLiteral    | SQL 式に無効な演算子が含まれています。                                                      | 400                    | クライアント             |
| LexerInvalidOperator   | SQL 式に無効なリテラルが含まれています。                                                     | 400                    | クライアント             |
| LikeInvalidInputs      | SQL 式の LIKE 句に指定された引数が無効です。                                                | 400                    | クライアント             |
| MalformedXML           | 指定された XML の書式に誤りがあるか、AWS の公開済みスキーマに一致しませんでした。サービスのドキュメントを確認して、もう一度試してください。 | 400                    | クライアント             |
| MaxOperatorsExceeded   | SQL 式を解析できませんでした。複雑さを軽減してみてください。例えば、使用する演算子の数を減らします。                       | 400                    | クライアント             |
| MethodNotAllowed       | 指定されたメソッドは、このリソースに対して許可されていません。                                            | 405 Method Not Allowed | クライアント             |



| エラーコード                         | 説明                                                                                                                                                                                  | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------------------|
| MissingRequiredParameter       | SelectRequest エンティティにリクエストされたパラメータが含まれていません。サービスのドキュメントを確認して、もう一度試してください。                                                                                                           | 400           | クライアント             |
| MultipleDataSourcesUnsupported | 複数のデータソースはサポートされていません。                                                                                                                                                              | 400           | クライアント             |
| NumberFormatError              | 数値の解析中にエラーが発生しました。このエラーは、整数のアンダーフローまたはオーバーフローが原因である可能性があります。                                                                                                                        | 400           | クライアント             |
| ObjectSerializationConflict    | InputSerialization で複数の形式 (CSV、JSON、または Parquet) が指定されているか、OutputSerialization で複数の形式 (CSV または JSON) が指定されています。InputSerialization および OutputSerialization では、それぞれに 1 つの形式のみを指定できます。 | 400           | クライアント             |
| OverMaxColumn                  | 結果の列数が最大許容列数を超過しています。                                                                                                                                                               | 400           | クライアント             |
| OverMaxParquetBlockSize        | Parquet ファイルが最大行グループサイズを超過しています。                                                                                                                                                    | 400           | クライアント             |

| エラーコード                                   | 説明                                                   | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|------------------------------------------|------------------------------------------------------|---------------|--------------------|
| OverMaxRecordSize                        | 入力または結果のレコードの長さが 1 MB の maxCharsPerRecord 制限を超えています。 | 400           | クライアント             |
| ParquetParsingError                      | Parquet ファイルの解析中にエラーが発生しました。ファイルを確認して、もう一度試してください。   | 400           | クライアント             |
| ParquetUnsupportedCompressionCodec       | 指定された Parquet 圧縮コーデックはサポートされていません。                   | 400           | クライアント             |
| ParseAsteriskIsNotAloneInSelectList      | * が SQL 式のドット表記なしで使用されている場合、他の式は SELECT リストで許可されません。 | 400           | クライアント             |
| ParseCannotMixSqbandWildcardInSelectList | SQL 式の SELECT リストでは、同じ式内に [] と * を混在させることはできません。     | 400           | クライアント             |
| ParseCastArity                           | SQL 式 CAST のアリティが正しくありません。                           | 400           | クライアント             |
| ParseEmptySelect                         | SQL 式に空の SELECT 句が含まれています。                           | 400           | クライアント             |
| ParseExpected2TokenTypes                 | SQL 式で期待されるトークンが見つかりませんでした。                          | 400           | クライアント             |
| ParseExpectedArgumentDelimiter           | SQL 式で期待される引数区切り文字が見つかりませんでした。                       | 400           | クライアント             |

| エラーコード                                    | 説明                                   | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|-------------------------------------------|--------------------------------------|---------------|--------------------|
| ParseExpectedDatePart                     | SQL 式で期待される日付部分が見つかりませんでした。          | 400           | クライアント             |
| ParseExpectedExpression                   | 期待される SQL 式が見つかりませんでした。              | 400           | クライアント             |
| ParseExpectedIdentForAlias                | SQL 式のエイリアスに対して期待される識別子が見つかりませんでした。  | 400           | クライアント             |
| ParseExpectedIdentForAt                   | SQL 式の AT 名前に対して期待される識別子が見つかりませんでした。 | 400           | クライアント             |
| ParseExpectedIdentForGroupName            | GROUP は SQL 式ではサポートされていません。          | 400           | クライアント             |
| ParseExpectedKeyword                      | SQL 式で期待されるキーワードが見つかりませんでした。         | 400           | クライアント             |
| ParseExpectedLeftParenAfterCast           | SQL 式で CAST の後に期待される左括弧が見つかりませんでした。  | 400           | クライアント             |
| ParseExpectedLeftParenBuiltinFunctionCall | SQL 式で期待される左括弧が見つかりませんでした。           | 400           | クライアント             |
| ParseExpectedLeftParenValueConstructor    | SQL 式で期待される左括弧が見つかりませんでした。           | 400           | クライアント             |
| ParseExpectedMember                       | SQL 式でサポートされていない MEMBER が使用されています。   | 400           | クライアント             |

| エラーコード                                     | 説明                                               | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|--------------------------------------------|--------------------------------------------------|---------------|--------------------|
| ParseExpectedNumber                        | SQL 式で期待される数値が見つかりませんでした。                        | 400           | クライアント             |
| ParseExpectedRightParenBuiltinFunctionCall | SQL 式で期待される右括弧が見つかりませんでした。                       | 400           | クライアント             |
| ParseExpectedTokenType                     | SQL 式で期待されるトークンが見つかりませんでした。                      | 400           | クライアント             |
| ParseExpectedTypeName                      | SQL 式で期待されるタイプ名が見つかりませんでした。                      | 400           | クライアント             |
| ParseExpectedWhenClause                    | SQL 式で期待される WHEN 句が見つかりませんでした。CASE はサポートされていません。 | 400           | クライアント             |
| ParseInvalidContextForWildcardInSelectList | SQL 式の SELECT リストでの * の使用は無効です。                  | 400           | クライアント             |
| ParseInvalidPathComponent                  | SQL 式に無効なパスコンポーネントが含まれています。                      | 400           | クライアント             |
| ParseInvalidTypeParam                      | SQL 式に無効なパラメータ値が含まれています。                         | 400           | クライアント             |
| ParseMalformedJoin                         | JOIN は SQL 式ではサポートされていません。                       | 400           | クライアント             |
| ParseMissingIdentifierAfterAt              | SQL 式の @ 記号の後に期待される識別子が見つかりませんでした。               | 400           | クライアント             |

| エラーコード                             | 説明                                         | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|------------------------------------|--------------------------------------------|---------------|--------------------|
| ParseNonUnaryAggregateFunctionCall | SQL 式の集計関数では、1 つの引数のみがサポートされています。          | 400           | クライアント             |
| ParseSelectMissingFrom             | SQL 式の SELECT リストの後に FROM が欠落しています。        | 400           | クライアント             |
| ParseUnexpectedKeyword             | SQL 式に予期しないキーワードが含まれています。                  | 400           | クライアント             |
| ParseUnexpectedOperator            | SQL 式に予期しない演算子が含まれています。                    | 400           | クライアント             |
| ParseUnexpectedTerm                | SQL 式に予期しない用語が含まれています。                     | 400           | クライアント             |
| ParseUnexpectedToken               | SQL 式に予期しないトークンが含まれています。                   | 400           | クライアント             |
| ParseUnknownOperator               | SQL 式に無効な演算子が含まれています。                      | 400           | クライアント             |
| ParseUnsupportedAlias              | SQL 式でサポートされていない ALIAS が使用されています。          | 400           | クライアント             |
| ParseUnsupportedCallWithStar       | SQL 式では、(*) をパラメータとする COUNT のみがサポートされています。 | 400           | クライアント             |
| ParseUnsupportedCase               | SQL 式でサポートされていない CASE が使用されています。           | 400           | クライアント             |
| ParseUnsupportedCaseClause         | SQL 式でサポートされていない CASE が使用されています。           | 400           | クライアント             |

| エラーコード                          | 説明                                                               | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|---------------------------------|------------------------------------------------------------------|---------------|--------------------|
| ParseUnsupportedLiteralsGroupBy | SQL 式でサポートされていない GROUP BY が使用されています。                             | 400           | クライアント             |
| ParseUnsupportedSelect          | SQL 式でサポートされていない SELECT が使用されています。                               | 400           | クライアント             |
| ParseUnsupportedSyntax          | SQL 式にサポートされていない構文が含まれています。                                      | 400           | クライアント             |
| ParseUnsupportedToken           | SQL 式にサポートされていないトークンが含まれています。                                    | 400           | クライアント             |
| TruncatedInput                  | オブジェクトの解凍に失敗しました。オブジェクトがリクエストで指定された形式を使用して適切に圧縮されていることを確認してください。 | 400           | クライアント             |
| UnauthorizedAccess              | この操作を実行する権限がありません。                                               | 401           | クライアント             |
| UnrecognizedFormatException     | 無効なレコードタイプが検出されました。                                              | 400           | クライアント             |
| UnsupportedFunction             | サポートされていない SQL 関数が検出されました。                                       | 400           | クライアント             |
| UnsupportedParquetType          | 指定された Parquet 型はサポートされていません。                                     | 400           | クライアント             |
| UnsupportedRangeHeader          | 範囲ヘッダーは、このオペレーションではサポートされていません。                                  | 400           | クライアント             |

| エラーコード                     | 説明                                                                                      | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|----------------------------|-----------------------------------------------------------------------------------------|---------------|--------------------|
| UnsupportedScanRangeInput  | スキャン範囲クエリは、このタイプのオブジェクトではサポートされていません。                                                   | 400           | クライアント             |
| UnsupportedSqlOperation    | サポートされていない SQL オペレーションが検出されました。                                                         | 400           | クライアント             |
| UnsupportedSqlStructure    | サポートされていない SQL 構造が検出されました。SQL リファレンスを確認してください。                                          | 400           | クライアント             |
| UnsupportedStorageClass    | サポートされていないストレージクラスが検出されました。サポートされているストレージクラスは、STANDARD、STANDARD_IA、および ONEZONE_IA のみです。 | 400           | クライアント             |
| UnsupportedSyntax          | 無効な構文が検出されました。                                                                          | 400           | クライアント             |
| UnsupportedTypeForQuerying | クエリに、比較に使用できない型が含まれています (例: Parquet INT96 列型が 0 より大きいことを確認する)。                          | 400           | クライアント             |
| ValueParseFailure          | SQL 式でタイムスタンプ解析エラーが発生しました。                                                              | 400           | クライアント             |

# レプリケーション関連のエラーコードのリスト

次の表に、Replication オペレーションが返す可能性のある特殊なエラーを示します。Amazon S3 のエラーとエラーコードのリストの一般的な情報については、「[エラーレスポンス](#)」を参照してください。

| エラーコード          | 説明                                                                                                                                                                                                                                                                                                                                                                                                                 | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------------------|
| InvalidArgument | <p>このエラーは、次の原因によって発生する可能性があります。</p> <ul style="list-style-type: none"><li>• &lt;Account&gt; 要素が空になっている場合。有効なアカウント ID が含まれている必要があります。</li><li>• &lt;Account&gt; 要素で指定された AWS アカウントが送信先バケット所有者と一致する必要があります。</li><li>• ReplicationTime-Status には値を含める必要があります。</li><li>• ReplicationTime-ReplicationTimeValue には値を含める必要があります。</li><li>• Replication-ReplicationTimeValue-Minutes 値は 15 である必要があります。</li><li>•</li></ul> | 400           | クライアント             |



| エラーコード | 説明                                                                                                                                                                                                                                                                              | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------------------|
|        | <p>ReplicationMetrics には Status が含まれている必要があります。</p> <ul style="list-style-type: none"><li>• ReplicationMetrics には EventThreshold が含まれている必要があります。</li><li>• EventThreshold-ReplicationTimeValue-Minutes 値は 15 である必要があります。</li><li>• Rule ID に ASCII 以外の文字を含めることはできません。</li></ul> |               |                    |

| エラーコード         | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|--------------------|
| InvalidRequest | <p>このエラーは、次の原因によって発生する可能性があります。</p> <ul style="list-style-type: none"><li>• <code>&lt;AccessControlTranslation&gt;</code> の <code>&lt;Owner&gt;</code> には値があるため、<code>&lt;Account&gt;</code> 要素を指定する必要があります。</li><li>• <code>&lt;Account&gt;</code> 要素が空になっている場合。有効なアカウント ID が含まれている必要があります。</li><li>• レプリケーション <code>destination</code> には、<code>ReplicationTime</code> と <code>Metrics</code> の両方が含まれているか、どちらも含まれていない必要があります。</li><li>• <code>ReplicationTime</code> と <code>ReplicationMetrics</code> は同じステータスである必要があります。</li><li>• S3 Replication Time Control (S3 RTC) は、この AWS リージョンではサポートされていません。</li></ul> | 400           | クライアント             |

| エラーコード                                | 説明                        | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|---------------------------------------|---------------------------|---------------|--------------------|
| ReplicationConfigurationNotFoundError | このバケットにはレプリケーション設定はありません。 | 404 Not Found | クライアント             |

## タグ付け関連のエラーコードのリスト

次の表に、ストレージレンズグループに対して

TagResource、UntagResource、ListTagsForResource オペレーションが返す可能性のある特殊なエラーを示します。一般的な Amazon S3 のエラーとエラーコードのリストについては、「[エラーレスポンス](#)」を参照してください。

| エラーコード         | 説明                                                                                                                                  | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------|
| InvalidRequest | リソース ARN の AWS リージョンが、このリクエストで指定されたリージョンと一致しません。リソース ARN の AWS アカウントが、このリクエストで指定されたアカウント ID と一致しません。resourceArn の AWS パーティションは無効です。 | 400 Bad Request | サポートされていません        |
| InvalidTag     | このリクエストに無効なタグキーまたは値が含まれています。有効な文字には、[a-zA-Z+-=._:/] が含まれます。タグキーの長さは最大 128 文字です。タグ値の長さは最大 256 文字で                                    | 400 Bad Request | サポートされていません        |

| エラーコード             | 説明                                                   | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|--------------------|------------------------------------------------------|-----------------|--------------------|
|                    | す。リクエストに重複するタグキーがあります。ユーザー定義タグキーは aws: で始めることはできません。 |                 |                    |
| NoSuchResource     | 指定されたリソースは存在しません。                                    | 404 Not Found   | サポートされていません        |
| TagPolicyException | タグポリシーでは、次のタグキーに指定された値は許可されていません。                    | 400 Bad Request | サポートされていません        |
| TooManyTags        | タグの数が 50 タグの制限を超えています。                               | 400 Bad Request | サポートされていません        |

## Amazon S3 注釈関連のエラーコードのリスト

次の表に、注釈オペレーション

(PutObjectAnnotation、GetObjectAnnotation、ListObjectAnnotations、DeleteObjectAnnotation) と注釈関連の CopyObject 動作が返す可能性のある特殊なエラーを示します。Amazon S3 のエラーとエラーコードのリストの一般的な情報については、「[エラーレスポンス](#)」を参照してください。

| エラーコード       | 説明                                         | HTTP ステータスコード | SOAP 障害コードのプレフィックス |
|--------------|--------------------------------------------|---------------|--------------------|
| AccessDenied | アクセスが拒否されました。呼び出し元にリクエストされた注釈オペレーションを実行するア | 403 Forbidden | サポートされていません        |

| エラーコード                    | 説明                                                                                                                                  | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------|
|                           | アクセス許可がないか、オブジェクトが Object Lock によって COMPLIANCE モードで保護されています。                                                                        |                 |                    |
| AnnotationLimitExceeded   | オブジェクト注釈数が制限 (1000) を超えています。                                                                                                        | 400 Bad Request | サポートされていません        |
| AnnotationNameTooLong     | 注釈名が長すぎます。注釈名の長さは最大 512 バイトです。                                                                                                      | 400 Bad Request | サポートされていません        |
| AnnotationRequestConflict | 競合するオペレーションが現在この注釈に対して進行中です。これは、クロスリージョンコピー中に、ソース注釈が削除された場合に発生する可能性があります。                                                           | 409 Conflict    | サポートされていません        |
| EntityTooSmall            | 提案されたアップロードが最小許容サイズより小さくなっています。注釈ペイロードは 1 バイト以上である必要があります。                                                                          | 400 Bad Request | サポートされていません        |
| InvalidAnnotationName     | 注釈名に無効な文字が含まれているか、名前が予約プレフィックス (aws または s3、大文字と小文字を区別しない) で始まっています。注釈名には、Unicode 文字、数字、アンダースコア (_)、ピリオド (.)、ハイフン (-) のみを含めることができます。 | 400 Bad Request | サポートされていません        |

| エラーコード          | 説明                                                                                                                                                                                                                          | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------|
| InvalidArgument | 注釈ディレクティブが無効です。有効な値は、COPY および EXCLUDE です。ListObjectAnnotations の場合、max-annotation-results は 1～1000 の間で指定する必要があります。                                                                                                          | 400 Bad Request | サポートされていません        |
| InvalidPrefix   | 指定した注釈プレフィックスが無効です。                                                                                                                                                                                                         | 400 Bad Request | サポートされていません        |
| InvalidRequest  | このエラーは、オブジェクトが注釈をサポートしていない SSE-C で暗号化されているために発生する可能性があります。ソースオブジェクトには、送信先にコピーできない注釈が含まれてます。送信先がサポートされていないリージョンにあるためです。注釈テーブル設定リクエストが無効です (暗号化の指定中にテーブルを無効にする、既に有効になっているテーブルを有効にする、CREATING ステータスのテーブルを更新する、空のリクエストを送信するなど)。 | 400 Bad Request | サポートされていません        |

| エラーコード           | 説明                                                                                                            | HTTP ステータスコード          | SOAP 障害コードのプレフィックス |
|------------------|---------------------------------------------------------------------------------------------------------------|------------------------|--------------------|
| MethodNotAllowed | 指定されたメソッドは、このリソースに対して許可されていません。例えば、このエラーは、リクエストが削除マーカをターゲットにしている場合に発生します。                                     | 405 Method Not Allowed | サポートされていません        |
| NoSuchAnnotation | 指定された注釈が存在しません。                                                                                               | 404 Not Found          | サポートされていません        |
| NoSuchKey        | 指定されたキーが存在しません。                                                                                               | 404 Not Found          | サポートされていません        |
| NoSuchVersion    | リクエストで指定されたバージョン ID が既存のバージョンと一致しません。                                                                         | 404 Not Found          | サポートされていません        |
| NotImplemented   | Amazon S3 ディレクトリバケットでは注釈はサポートされていません。                                                                         | 501 Not Implemented    | サポートされていません        |
| OperationAborted | 競合する条件付きオペレーションが現在このリソースに対して進行中です。これは、同時変更中、またはクロスリージョンコピー中にソース注釈数または ETag が変更された場合に発生する可能性があります。もう一度試してください。 | 409 Conflict           | サポートされていません        |

| エラーコード                            | 説明                                                                                | HTTP ステータスコード              | SOAP 障害コードのプレフィックス |
|-----------------------------------|-----------------------------------------------------------------------------------|----------------------------|--------------------|
| PreconditionFailed                | 指定した前提条件の少なくとも 1 つが保持されませんでした。x-amz-object-if-match ETag が現在のオブジェクト ETag と一致しません。 | 412 Precondition Failed    | サポートされていません        |
| UnsupportedMediaType              | リクエストペイロードには、サポートされていないメディアタイプが含まれています。注釈ペイロードは、有効な UTF-8 エンコードテキストである必要があります。    | 415 Unsupported Media Type | サポートされていません        |
| UserAnnotationNameMustBeSpecified | ユーザー注釈名を指定する必要があります。                                                              | 400 Bad Request            | サポートされていません        |

## Amazon S3 on Outposts エラーコードのリスト

次の表に、Amazon S3 on Outposts のオペレーションが返す可能性のある特殊なエラーを示します。Amazon S3 のエラーとエラーコードのリストの一般的な情報については、「[エラーレスポンス](#)」を参照してください。

| エラーコード     | 説明                                          | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|------------|---------------------------------------------|-----------------|--------------------|
| BadRequest | 前回の削除の試行により、バケットは移行状態になっています。後でもう一度お試しください。 | 400 Bad Request | サポートされていません        |



| エラーコード               | 説明                                                                                                                                                     | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------|
| InvalidRequest       | このエラーは、次の原因によって発生する可能性があります。 <ul style="list-style-type: none"><li>Amazon VPC 設定が必要です。</li><li>S3 on Outposts アクセスポイントでのパブリックアクセスは許可されていません。</li></ul> | 400 Bad Request | クライアント             |
| InvalidOutpostState  | リクエストは Outpost の現在の状態で有効ではありません。                                                                                                                       | 409 Conflict    | サポートされていません        |
| InvalidRequest       | アクセスポイントが削除可能な状態ではありません。                                                                                                                               | 400 Bad Request | サポートされていません        |
| NoSuchOutpost        | 指定された Outpost は存在しません。                                                                                                                                 | 404 Not Found   | サポートされていません        |
| UnsupportedOperation | 指定されたアクションはサポートされていません。                                                                                                                                | 404 Not Found   | サポートされていません        |
| InsufficientCapacity | 容量不足です。                                                                                                                                                | 507 ストレージ不足     | サポートされていません        |

# Amazon S3 ストレージレンズエラーコードのリスト

次の表に、Amazon S3 ストレージレンズのオペレーションが返す可能性のある特殊なエラーを示します。一般的な Amazon S3 のエラーとエラーコードのリストについては、「[エラーレスポンス](#)」を参照してください。

| エラーコード                              | 説明                                                                          | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|-------------------------------------|-----------------------------------------------------------------------------|-----------------|--------------------|
| AccessDenied                        | このリージョンは、S3 ストレージレンズのホームリージョンとしてサポートされていません。                                | 403 Forbidden   | サポートされていません        |
| AccountNotAuthorized                | このアカウントには、AWS Organizations を使用する権限がありません。管理アカウントまたは委任された管理者アカウントを使用してください。 | 403 Forbidden   | サポートされていません        |
| ActivityMetricsMustEnabled          | アクティビティメトリクスを有効にする必要があります。                                                  | 400 Bad Request | サポートされていません        |
| AWSOrganizationsNotInUseException   | このアカウントは、組織に属していません。                                                        | 403 Forbidden   | サポートされていません        |
| DefaultConfigurationDeleteForbidden | デフォルト設定は削除できません。                                                            | 403 Forbidden   | サポートされていません        |
| DuplicateStorageLensGroupARN        | この設定には、同じストレージレンズグループ ARN のエントリが 2 つ以上含まれています。                              | 400 Bad Request | サポートされていません        |

| エラーコード                | 説明                                                                                                                                                                                               | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------|
| EmptyExcludeContainer | <p>このエラーは、以下の原因で発生することがあります。</p> <ul style="list-style-type: none"><li>除外コンテナを空にすることはできません。</li><li>除外コンテナにはバケットを 1 つ以上含める必要があります。</li><li>除外コンテナにはリージョンを 1 つ以上含める必要があります。</li></ul>             | 400 Bad Request | サポートされていません        |
| EmptyExcludeElement   | Exclude 要素を使用してストレージレンズグループを指定する必要があります。                                                                                                                                                         | 400 Bad Request | サポートされていません        |
| EmptyIncludeContainer | <p>このエラーは、以下の原因で発生することがあります。</p> <ul style="list-style-type: none"><li>インクルードコンテナを空にすることはできません。</li><li>インクルードコンテナにはバケットを 1 つ以上含める必要があります。</li><li>インクルードコンテナにはリージョンを 1 つ以上含める必要があります。</li></ul> | 400 Bad Request | サポートされていません        |

| エラーコード                               | 説明                                                       | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|--------------------------------------|----------------------------------------------------------|-----------------|--------------------|
| InvalidAWSOrgArn                     | 設定に不正な形式の AWS Organizations ARN があります。                   | 400 Bad Request | サポートされていません        |
| EmptyIncludeElement                  | Include 要素を使用してストレージレンズグループを指定する必要があります。                 | 400 Bad Request | サポートされていません        |
| InvalidBucketFilter                  | 組織レベルの設定では、バケットフィルターはサポートされていません。                        | 400 Bad Request | サポートされていません        |
| InvalidConfigId                      | 設定 ID が無効です。                                             | 400 Bad Request | サポートされていません        |
| InvalidDestination                   | S3 バケット ARN の形式が正しくありません。                                | 400 Bad Request | サポートされていません        |
| InvalidEncryptionMethod              | 指定できる暗号化方法は 1 つのみです。                                     | 400 Bad Request | サポートされていません        |
| InvalidFilterForDefaultConfiguration | デフォルト設定にはフィルターを含めないでください。                                | 400 Bad Request | サポートされていません        |
| InvalidIncludeExcludeContainers      | 設定では、インクルードコンテナまたは除外コンテナのいずれかを指定できます。設定で両方を指定することはできません。 | 400 Bad Request | サポートされていません        |

| エラーコード                                     | 説明                                                                                 | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|--------------------------------------------|------------------------------------------------------------------------------------|-----------------|--------------------|
| InvalidIncludeExcludeElements              | Include または Exclude 要素は 1 つのみ使用できます。少なくとも 1 つの Include または Exclude 要素が存在する必要があります。 | 400 Bad Request | サポートされていません        |
| InvalidKMSEncryptionKeyId                  | KMS キー ID ARN が無効です。                                                               | 400 Bad Request | サポートされていません        |
| InvalidMaximumPrefixDepth                  | MaxDepth は [1,10] の範囲内である必要があります。                                                  | 400 Bad Request | サポートされていません        |
| InvalidMinimumStorageBytesPercentage       | MinStorageBytesPercentage は [1.00,100.00] の範囲内である必要があります。                          | 400 Bad Request | サポートされていません        |
| InvalidOrganizationARN                     | 設定の AWS Organizations ARN が無効です。                                                   | 400 Bad Request | サポートされていません        |
| InvalidOrganizationForDefaultConfiguration | デフォルト設定では、組織レベルのメトリクスはサポートされていません。                                                 | 400 Bad Request | サポートされていません        |
| InvalidRegionForDefaultConfiguration       | 指定されたリージョンは、デフォルト設定ではサポートされていません。                                                  | 400 Bad Request | サポートされていません        |
| InvalidRegionName                          | リージョン名が無効です。                                                                       | 400 Bad Request | サポートされていません        |

| エラーコード                             | 説明                                                                                                                                                                                                                  | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------|
| InvalidStorageLensArn              | S3 ストレージレンズ ARN は入力に必須ではありません。                                                                                                                                                                                      | 400 Bad Request | サポートされていません        |
| InvalidStorageLensGroupARN         | このストレージレンズグループ ARN が無効であるか、アカウント内のストレージレンズグループのみが許可されています。さらに、ストレージレンズグループ ARN の構造 <code>arn::s3::storage-lens-group/</code> に従い、64 文字の制限に従う必要があります。ストレージレンズグループ名には、a~z、A~Z、0~9、ハイフン (-)、アンダースコア (_) のみを含めることができます。 | 400 Bad Request | サポートされていません        |
| MissingAccountLevelActivityMetrics | アクティビティメトリクスがバケットレベルで有効になっている場合、アクティビティメトリクスはアカウントレベルで有効にする必要があります。                                                                                                                                                 | 400 Bad Request | サポートされていません        |
| MissingBucketLevelActivityMetrics  | アクティビティメトリクスがアカウントレベルで有効になっている場合、アクティビティメトリクスはバケットレベルで有効にする必要があります。                                                                                                                                                 | 400 Bad Request | サポートされていません        |

| エラーコード                           | 説明                                                    | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|----------------------------------|-------------------------------------------------------|-----------------|--------------------|
| MissingEncryptionMethod          | 暗号化メソッドを空白にすることはできません。または SSE-KMS のいずれかを指定します。。SSE-S3 | 400 Bad Request | サポートされていません        |
| MissingPrefixLevelStorageMetrics | プレフィックスレベルのストレージメトリクスは、プレフィックスレベルが有効になっている場合に必須です。    | 400 Bad Request | サポートされていません        |
| OrganizationAccessDenied         | このアカウントには、AWS Organizations を追加する権限がありません。            | 403 Forbidden   | サポートされていません        |
| OrgConfigurationNotSupported     | 指定されたリージョンは、設定で AWS Organizations をサポートしていません。        | 403 Forbidden   | サポートされていません        |
| ServiceNotEnabledForOrg          | S3 ストレージレンズサービスにリンクされたロールが組織に対して有効になっていません。           | 403 Forbidden   | サポートされていません        |
| StorageMetricsMustEnabled        | プレフィックスレベルのストレージメトリクスを有効にする必要があります。                   | 400 Bad Request | サポートされていません        |
| TooManyBuckets                   | バケットコンテナには 50 個を超えるバケットを含めることはできません。                  | 400 Bad Request | サポートされていません        |
| TooManyRegions                   | リージョンコンテナには 50 個を超えるリージョンを含めることはできません。                | 400 Bad Request | サポートされていません        |

| エラーコード                   | 説明                                                   | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|--------------------------|------------------------------------------------------|-----------------|--------------------|
| TooManyStorageLensGroups | 50 個を超えるストレージレンズグループをストレージレンズダッシュボードにアタッチすることはできません。 | 400 Bad Request | サポートされていません        |

次の表に、S3 ストレージレンズグループのオペレーションが返す可能性のある特殊なエラーを示します。一般的な Amazon S3 のエラーとエラーコードのリストについては、「[エラーレスポンス](#)」を参照してください。

| エラーコード                     | 説明                                                                                    | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|----------------------------|---------------------------------------------------------------------------------------|-----------------|--------------------|
| AccessDenied               | ストレージレンズグループのアクションを実行するためのアクセス許可がありません。このリジョンは、S3 ストレージレンズグループのホームリジョンとしてサポートされていません。 | 403 Forbidden   | サポートされていません        |
| ConfigurationAlreadyExists | 指定された設定は既に存在しています。                                                                    | 409 Conflict    | サポートされていません        |
| DuplicateElement           | タグは一意である必要があります。And 論理演算子に重複するタグキーが含まれています。Or 論理演算子に重複するタグが含まれています。論理演算子に重複する         | 400 Bad Request | サポートされていません        |



| エラーコード                 | 説明                                                                                                           | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|------------------------|--------------------------------------------------------------------------------------------------------------|-----------------|--------------------|
|                        | プレフィックスまたはサフィックスが含まれています。                                                                                    |                 |                    |
| InvalidAge             | DaysLessThan および DaysGreaterThan は正の数である必要があります。                                                             | 400 Bad Request | サポートされていません        |
| InvalidFilter          | フィルターには、And、Or、MatchAnyTag、MatchAnyPrefix、MatchAnySuffix、MatchObjectAge、MatchObjectSize のいずれかの要素を含める必要があります。 | 400 Bad Request | サポートされていません        |
| InvalidLogicalOperator | 論理演算子 And または Or には、少なくとも 2 つのサブ要素が存在する必要があります。                                                              | 400 Bad Request | サポートされていません        |
| InvalidMatchAnyPrefix  | パラメータ MatchAnyPrefix を空にすることはできません。                                                                          | 400 Bad Request | サポートされていません        |
| InvalidMatchAnySuffix  | パラメータ MatchAnySuffix を空にすることはできません。                                                                          | 400 Bad Request | サポートされていません        |
| InvalidMatchAnyTag     | パラメータ MatchAnyTag を空にすることはできません。                                                                             | 400 Bad Request | サポートされていません        |

| エラーコード                    | 説明                                                                                                                  | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|---------------------------|---------------------------------------------------------------------------------------------------------------------|-----------------|--------------------|
| InvalidMatchObjectAge     | パラメータ MatchObjectAge を空にすることはできません。                                                                                 | 400 Bad Request | サポートされていません        |
| InvalidMatchObjectSize    | パラメータ MatchObjectSize を空にすることはできません。                                                                                | 400 Bad Request | サポートされていません        |
| InvalidName               | ストレージレンズグループ名パラメータは 1~64 文字にする必要があります。ストレージレンズグループ Name パラメータは、^[a-zA-Z0-9\-_]+\$ パターンを使用する必要があります。                 | 400 Bad Request | サポートされていません        |
| InvalidNumericCombination | このオブジェクトの経過時間またはオブジェクトサイズの組み合わせは無効です。                                                                               | 400 Bad Request | サポートされていません        |
| InvalidPrefix             | プレフィックスの最大長は 1,024 文字です。プレフィックス文字列を空にすることはできません。                                                                    | 400 Bad Request | サポートされていません        |
| InvalidSize               | BytesLessThan および BytesGreaterThan は正の数である必要があります。最大オブジェクトサイズは 50 TB を超えることはできません。最小オブジェクトサイズを 50 TB 以上にすることはできません。 | 400 Bad Request | サポートされていません        |

| エラーコード                | 説明                                                                                                                                                                                                  | HTTP ステータスコード   | SOAP 障害コードのプレフィックス |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------------|
| InvalidSuffix         | サフィックスの最大長は 1,024 文字です。サフィックス文字列を空にすることはできません。                                                                                                                                                      | 400 Bad Request | サポートされていません        |
| InvalidTag            | オブジェクトタグキーの長さは 128 文字以下にする必要があります。オブジェクトタグのキー文字列を null または空にすることはできません。タグ値の最大長は 256 文字です。オブジェクトタグキーに無効な文字が含まれています。オブジェクトタグキーには、a~z、A~Z、0~9、スペース、および <code>^(_.:\/+=\-\@]*)*\$</code> のみを含める必要があります。 | 400 Bad Request | サポートされていません        |
| MismatchedName        | リクエストで指定された名前がストレージレンズグループ名と一致しません。                                                                                                                                                                 | 400 Bad Request | サポートされていません        |
| TooManyConfigurations | 許可されている上限 50 を超えるストレージレンズグループ設定を作成しようとしてしました。                                                                                                                                                       | 400 Bad Request | サポートされていません        |
| TooManyElements       | Element が、論理演算子内で許可される要素の最大数を超過しています。使用できるプレフィックス、サフィックス、またはタグは 10 個のみです。                                                                                                                           | 400 Bad Request | サポートされていません        |

# Amazon S3 Object Lambda エラーコードのリスト

次の表に、S3 Object Lambda が返す可能性のある特殊なエラーを示します。一般的な Amazon S3 のエラーとエラーコードのリストについては、「[エラーレスポンス](#)」を参照してください。

GetObject 以外のリクエスト中にサポートするアクセスポイントから受信したエラーレスポンスは、変更されずに発信者に送信されます。

| エラーコード                 | 説明                                                                                                                                                                                                                                | HTTP ステータスコード   |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| LambdaInvalidResponse  | <p>AWS Lambda に対して WriteGetObjectResponse が ValidationError で応答したため、元の発信者に返されました。</p> <p>詳細については、ValidationErrorMessage を参照してください。ValidationError のすべてのケースで LambdaInvalidResponse エラーが発生するわけではありません。</p>                          | 400 Bad Request |
| LambdaInvocationFailed | <p>Lambda 関数の呼び出しに失敗しました。</p> <p>S3 Object Lambda が設定された Lambda 関数の呼び出しを正常に完了できない場合、呼び出し元は次のエラーを受け取る場合があります。</p> <p>このエラーメッセージには、関数を呼び出す際に AWS Lambda サービスが返す結果のエラーの詳細 (ステータスコード、エラーコード、エラーメッセージ、リクエスト ID など) が含まれている場合があります。</p> | 400 Bad Request |

| エラーコード         | 説明                                                                                                                                                                                                                                                                         | HTTP ステータスコード |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| LambdaNotFound | <p>AWS Lambda 関数が見つかりませんでした。</p> <p>設定済みの Lambda 関数、バージョン、またはエイリアスが、呼び出し試行時に見つかりませんでした。S3 Object Lambda アクセスポイントの設定が適切な Lambda 関数 ARN を指していることを確認します。</p> <p>このエラーメッセージには、関数を呼び出す際に AWS Lambda サービスが返す結果のエラーの詳細 (ステータスコード、エラーコード、エラーメッセージ、リクエスト ID など) が含まれている場合があります。</p> | 404 Not Found |

| エラーコード                    | 説明                                                                                                                                                                                                                                                                                            | HTTP ステータスコード                       |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| LambdaPermissionError     | <p>呼び出し元は Lambda 関数を呼び出す権限がありません。</p> <p>呼び出し元には Lambda 関数を呼び出すアクセス許可が必要です。呼び出し元にアタッチされているポリシーを調べて、設定された関数に <code>lambda:Invoke</code> を使用することが許可されていることを確認します。</p> <p>このエラーメッセージには、関数を呼び出す際に AWS Lambda サービスが返す結果のエラーの詳細 (ステータスコード、エラーコード、エラーメッセージ、リクエスト ID など) が含まれている場合があります。</p>          | 403<br>Forbidden                    |
| LambdaResponseNotReceived | <p>Lambda 関数は <code>WriteGetObjectResponse</code> を正常に呼び出すことなく終了しました。</p> <p><code>GetObject</code> レスポンスデータは、Lambda 関数が <code>WriteGetObjectResponse</code> API オペレーションを呼び出すことで提供されます。関数の Amazon CloudWatch logs は、関数が正常に終了しても、この API オペレーションを正常に呼び出せなかった理由について、より多くのインサイトを提供する可能性があります。</p> | 500<br>Internal<br>Service<br>Error |

| エラーコード             | 説明                                                                                                       | HTTP ステータスコード                       |  |
|--------------------|----------------------------------------------------------------------------------------------------------|-------------------------------------|--|
| LambdaRuntimeError | <p>Lambda 関数が実行中に失敗しました。</p> <p>Lambda 関数から明示的なエラーを受信しました。失敗の詳細については、AWS CloudFormation ログを確認してください。</p> | 500<br>Internal<br>Service<br>Error |  |
| LambdaTimeout      | <p>Lambda 関数が許可された時間内に応答しませんでした。</p> <p>Lambda 関数が WriteGetObjectResponse への呼び出しを 60 秒以内に完了できませんでした。</p> | 500<br>Internal<br>Service<br>Error |  |

| エラーコード          | 説明                                                                                                                                                                                                                                                                                                                          | HTTP ステータスコード   |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| SlowDown        | <p>AWS Lambda を含むオペレーションのリクエストレートを下げてください。</p> <p>関数の呼び出しが AWS Lambda によってスロットリングされました。これはおそらく、設定された同時実行数の制限に達したためと考えられます。詳細については、「AWS Lambda デベロッパーガイド」の「<a href="#">Lambda 関数の同時実行の予約</a>」を参照してください。</p> <p>このエラーメッセージには、関数を呼び出す際に AWS Lambda サービスが返す結果のエラーの詳細 (ステータスコード、エラーコード、エラーメッセージ、リクエスト ID など) が含まれている場合があります。</p> | 503 Slow Down   |
| ValidationError | <p>検証エラーは、WriteGetObjectResponse API オペレーションから返され、発生した理由はさまざまである可能性があります。詳細については、エラーメッセージを参照してください。</p>                                                                                                                                                                                                                     | 400 Bad Request |

## Amazon S3 の非同期エラーコードのリスト

次の表に、非同期リクエストが返す可能性のある特殊なエラーを示します。Amazon S3 のエラーとエラーコードのリストの一般的な情報については、「[エラーレスポンス](#)」を参照してください。



これらのエラーは、DescribeMultiRegionAccessPointOperation を使用するなど、非同期リクエストの状態についてクエリを実行すると返されます。これらのリクエストは非同期であるため、これらのすべてのエラーのステータスコードは 200 OK です。

| エラーコード                                         | 説明                                                                       | HTTP ステータスコード |
|------------------------------------------------|--------------------------------------------------------------------------|---------------|
| AccessDenied                                   | アクセスが拒否されました。                                                            | 200 OK        |
| InternalErrors                                 | 内部サーバーエラーが発生しました。                                                        | 200 OK        |
| MalformedPolicy                                | 指定されたポリシー構文が無効です。                                                        | 200 OK        |
| MultiRegionAccessPointAlreadyOwnedByYou        | 同じ名前のマルチリージョンアクセスポイントが既に存在します。                                           | 200 OK        |
| MultiRegionAccessPointModifiedByAnotherRequest | 別のリクエストが指定されたリソースを変更しているため、アクションが失敗しました。前のリクエストが完了したら、リクエストを再送信してみてください。 | 200 OK        |
| MultiRegionAccessPointNotReady                 | 指定されたマルチリージョンアクセスポイントを更新する準備ができていません。                                    | 200 OK        |
| MultiRegionAccessPointSameBucketRegion         | マルチリージョンアクセスポイントの作成に使用されるバケットを同じリージョンにすることはできません。                        | 200 OK        |
| MultiRegionAccessPointUnsupportedRegion        | マルチリージョンアクセスポイントの作成に指定されたバケットの 1 つがサポートされていないリージョンにあります。                 | 200 OK        |

| エラーコード                       | 説明                            | HTTP ステータスコード |
|------------------------------|-------------------------------|---------------|
| NoSuchBucket                 | 指定されたバケットが存在しません。             | 200 OK        |
| NoSuchMultiRegionAccessPoint | 指定されたマルチリージョンアクセスポイントが存在しません。 | 200 OK        |

## Amazon S3 Access Grants エラーコードのリスト

次の表に、S3 Access Grants リクエストが返す可能性のある特殊なエラーを示します。Amazon S3 のエラーとエラーコードのリストの一般的な情報については、「[エラーレスポンス](#)」を参照してください。

| エラーコード                                         | 説明                                               | HTTP ステータスコード |
|------------------------------------------------|--------------------------------------------------|---------------|
| AccessGrantAlreadyExists                       | 指定されたアクセス許可は既に存在しています。                           | 409           |
| AccessGrantsInstanceAlreadyExists              | Access Grants インスタンスは既に存在しています。                  | 409           |
| AccessGrantsInstanceNotEmptyError              | Access Grants インスタンスを削除する前に、ロケーションをクリーンアップしてください | 400           |
| AccessGrantsInstanceDoesNotExistError          | Access Grants インスタンスが存在しません。                     | 404           |
| AccessGrantsInstanceResourcePolicyDoesNotExist | Access Grants インスタンスのリソースポリシーが存在しません。            | 404           |
| AccessGrantsLocationAlreadyExistsError         | 指定された Access Grants ロケーションは既に存在しています。            | 409           |

| エラーコード                                  | 説明                                                                                                           | HTTP ステータスコード |  |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------|---------------|--|
| AccessGrantsLocationNotEmptyError       | Access Grants ロケーションを削除する前に、Access Grants をクリーンアップしてください                                                     | 400           |  |
| AccessGrantsLocationsQuotaExceededError | Access Grants ロケーションのクォータを超えました。Access Grants ロケーションのクォータ: <i>&lt;value&gt;</i> 。引き上げが必要な場合は、S3 にお問い合わせください。 | 409           |  |
| AccessGrantsQuotaExceededError          | Access Grants のクォータを超えました。Access Grants のクォータ: <i>&lt;value&gt;</i> 。引き上げが必要な場合は、S3 にお問い合わせください。             | 409           |  |
| InvalidTag                              | リクエストに重複するタグキーがあります。重複するタグキーを削除して、もう一度試してください。                                                               | 400           |  |
| InvalidAccessGrant                      | 指定された Access Grants が無効です。                                                                                   | 400           |  |
| InvalidAccessGrantLocation              | 指定された Access Grants ロケーションが無効です。                                                                             | 400           |  |
| InvalidIamRole                          | 指定された IAM ロールが無効です、                                                                                          | 400           |  |
| InvalidIdentityCenterInstance           | 指定されたアイデンティティセンターインスタンスが無効です。                                                                                | 400           |  |
| InvalidResourcePolicy                   | 指定されたリソースポリシーが無効です。                                                                                          | 400           |  |
| InvalidResourcePolicy                   | 指定されたリソースポリシーが無効です。                                                                                          | 400           |  |

| エラーコード                          | 説明                                                                                                                                                                        | HTTP ステータスコード    |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| InvalidTag                      | このリクエストに無効なタグキーまたは値が含まれています。有効な文字には、[a-zA-Z+-._:/] が含まれます。タグキーの長さは最大 128 文字です。タグ値の長さは最大 256 文字です。                                                                         | 400              |
| NoSuchAccessGrantError          | 指定された Access Grants が存在しません。                                                                                                                                              | 404              |
| NoSuchAccessGrantsLocationError | 指定された Access Grants ロケーションが存在しません。                                                                                                                                        | 404              |
| AccessDenied                    | リクエストされた S3 プレフィックス <i>&lt;requested target&gt;</i> に対する <i>&lt;requested permission&gt;</i> アクセス許可がありません。                                                                | 403<br>Forbidden |
| StsNotAuthorizedError           | GetDataAccess オペレーションを呼び出す際にエラー (StsNotAuthorizedError) が発生しました。ユーザー access-grants.s3.amazonaws.com は、リソース <i>&lt;IAM Role ARN&gt;</i> に対して sts:AssumeRole を実行する権限がありません。 | 403              |
| StsPackedPolicyTooLargeError    | GetDataAccess オペレーションを呼び出す際にエラー (StsPackedPolicyTooLargeError) が発生しました。セッションに対してシリアル化されたトークンが大きすぎます。                                                                      | 400              |

| エラーコード             | 説明                                            | HTTP ステータスコード |
|--------------------|-----------------------------------------------|---------------|
| StsValidationError | エラーメッセージは、検証エラーによって異なります。                     | 400           |
| InvalidTags        | タグキーは、システムタグの AWS 予約済みプレフィックスで始めることはできません。    | 400           |
| TooManyTags        | タグの数が 50 タグの制限を超えています。一部のタグを削除して、もう一度試してください。 | 400           |

## Amazon FSx 関連のエラーコードのリスト

次の表に、Amazon S3 Access Points を介して Amazon FSx に対して行われたリクエストが返す可能性のある特殊なエラーを示します。Amazon S3 のエラーとエラーコードのリストの一般的な情報については、「[エラーレスポンス](#)」を参照してください。

| エラーコード               | 説明                                                                                                                                                                            | HTTP ステータスコード    |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| AccessDenied         | <p>このエラーは、次の原因によって発生する可能性があります。</p> <ul style="list-style-type: none"><li>発信者は、指定された S3 オペレーションを実行する権限がありません。</li><li>発信者は、指定されたキーの Amazon FSx リソースにアクセスする権限がありません。</li></ul> | 403<br>Forbidden |
| InsufficientCapacity | ファイルシステムの最大ストレージ容量に達しました。                                                                                                                                                     | 507<br>Insuffici |

| エラーコード     | 説明                | HTTP ステータスコード   |
|------------|-------------------|-----------------|
|            |                   | ent Capacity    |
| InvalidKey | 指定されたキーが無効です。     | 400 Bad Request |
| SlowDown   | リクエストレートを下げてください。 | 503 Slow Down   |

## Amazon S3 Tables のエラーコードのリスト

次の表に、Amazon S3 Tables のオペレーションが返す可能性のある特殊なエラーを示します。Amazon S3 のエラーとエラーコードのリストの一般的な情報については、「[エラーレスポンス](#)」を参照してください。

| エラーコード                       | 説明                                           | HTTP ステータスコード             |
|------------------------------|----------------------------------------------|---------------------------|
| AccessDeniedException        | 必要なアクセス許可がないため、アクションを実行できません。                | 403 Forbidden             |
| BadRequestException          | リクエストが無効であるか、形式が正しくありません。                    | 400 Bad Request           |
| ConflictException            | 以前の書き込みと競合しているため、リクエストが失敗しました。リクエストは再試行できます。 | 409 Conflict              |
| ForbiddenException           | 発信者にはリクエストを行う権限がありません。                       | 403 Forbidden             |
| InternalServerErrorException | 内部サーバーエラーのため、リクエストが失敗しました。                   | 500 Internal Server Error |

| エラーコード                   | 説明                                | HTTP ステータスコード         |
|--------------------------|-----------------------------------|-----------------------|
| NotFoundException        | 指定されたリソースが見つからないため、リクエストが拒否されました。 | 404 Not Found         |
| TooManyRequestsException | 1 秒あたりのリクエスト数の上限を超えました。           | 429 Too Many Requests |

## Amazon S3 のエラーに関するベストプラクティス

多くのエラーレスポンスには、プログラミングエラーを診断する開発者が読んで理解できるように追加の構造化データが含まれています。たとえば、REST PUT リクエストと共に Content-MD5 ヘッダーを送信し、これがサーバーで計算されたダイジェストと一致しない場合、BadDigest エラーが発生します。エラーレスポンスには、計算されたダイジェスト、指定されたダイジェストなど、詳細要素も含まれます。開発中にこの情報を使用してエラーを診断できます。実稼働では、正常に動作するプログラムによりこの情報がエラーログに記録されることがあります。

Amazon S3 と連携するアプリケーションを設計するときは、Amazon S3 のエラーを適切に処理することが重要です。このセクションでは、アプリケーションの設計時に考慮すべき問題について説明します。

### InternalError の場合は再試行する

内部エラーとは、Amazon S3 内で発生するエラーです。

InternalError レスポンスを受け取った場合、リクエストが処理されていない可能性があります。たとえば、PUT リクエストが InternalError を返す場合、それ以降の GET は古い値を取得することもある場合、更新された値を取得することもあります。

Amazon S3 が InternalError レスポンスを返す場合は、リクエストを再試行してください。

### SlowDown エラーを繰り返すアプリケーションを調整する

S3 は分散システムの常として、意図的であるかどうかに関係なくリソースの過剰な消費を検出して対応する保護メカニズムを備えています。SlowDown エラーは、リクエストレートが高いためにこのようなメカニズムのいずれかが引き起こされた場合に発生することがあります。リクエストレート

を低くすると、この種のエラーは減るかなくなります。一般に、ほとんどのユーザーはこのようなエラーを日常的に経験することはありません。ただし、詳細を知りたい場合や、SlowDown エラーが頻繁にまたは予期せず発生する場合は、[Amazon S3 デベロッパーフォーラム](https://aws.amazon.com/premiumsupport/)に投稿するか、または サポート (<https://aws.amazon.com/premiumsupport/>) にサインアップしてください。

## エラーを分離する

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

Amazon S3 には、SOAP と REST API の両方で使用されるエラーコードが用意されています。SOAP API は標準の Amazon S3 エラーコードを返します。REST API は標準の HTTP サーバーのように見え、既存の HTTP クライアント (例: ブラウザ、HTTP クライアントライブラリ、プロキシ、キャッシュなど) を操作するように設計されています。HTTP クライアントがエラーを適切に処理できるように、各 Amazon S3 エラーは HTTP ステータスコードにマッピングされています。

HTTP ステータスコードは Amazon S3 エラーコードよりも内容がおおまかであるため、エラーに関する情報は少なくなります。たとえば、NoSuchKey と NoSuchBucket という Amazon S3 エラーはどちらも HTTP 404 Not Found ステータスコードにマッピングされます。

HTTP ステータスコードに含まれるエラー情報は少なくなりますが、クライアントが HTTP を認識できて Amazon S3 API を認識できない場合も、通常はエラーが適切に処理されます。

このため、エラーを処理するときや Amazon S3 エラーをエンドユーザーに報告するときは、HTTP ステータスコードではなく、エラーに関する情報が多く含まれる Amazon S3 エラーコードを使用します。さらに、アプリケーションをデバッグするときは、人が読み取り可能な XML エラーレスポンスの <Details> 要素も参照してください。



# AWS 用語集

最新の AWS の用語については、「AWS の用語集 リファレンス」の「[AWS 用語集](#)」を参照してください。

# Amazon S3 のリソース

次の表は、このサービスを利用する際に役立つ関連リソースのリストです。

| リソース                                                  | 説明                                                                                      |
|-------------------------------------------------------|-----------------------------------------------------------------------------------------|
| <a href="#">Amazon Simple Storage Service ユーザーガイド</a> | 入門ガイドは、簡単なユースケースを基にサービスのクイックチュートリアルを提供します。                                              |
| <a href="#">Amazon Simple Storage Service ユーザーガイド</a> | 開発者ガイドでは、Amazon S3 オペレーションを使用してタスクを実行する方法について説明します。                                     |
| <a href="#">Amazon S3 に関する技術上のよくある質問</a>              | 本製品に関して開発者からよく寄せられる上位 20 の質問です。                                                         |
| <a href="#">Amazon S3 リリースノート</a>                     | リリースノートには、最新リリースの概要が記載されています。新機能、解決された問題、既知の問題が具体的に記載されています。                            |
| <a href="#">Amazon Web Services のツール</a>              | 関連ドキュメント、コードサンプル、リリースノートなどの情報に一か所でアクセスできるため、AWS SDK およびツールを使って革新的なアプリケーションを構築するのに役立ちます。 |
| <a href="#">AWS マネジメントコンソール</a>                       | コンソールを使用して、Amazon S3 のほとんどの機能をプログラミングを行うことなく実行できます。                                     |
| <a href="#">ディスカッションフォーラム</a>                         | 開発者が Amazon Web Services に関連する技術的な質問について話し合うためのコミュニティベースのフォーラムです。                       |
| <a href="#">サポート センター</a>                             | 当社の開発者フォーラム、技術上のよくある質問、サービスステータスページ、Premium Support へのアクセスを含む、AWS テクニカルサポートのホームページです。   |
| <a href="#">サポート</a>                                  | サポートに関する情報のメインウェブページです。AWS インフラストラクチャサービスでのアプリケーションの                                    |

| リソース                           | 説明                                                                   |
|--------------------------------|----------------------------------------------------------------------|
|                                | 構築および実行を支援するために 1 対 1 での迅速な対応を行うサポートチャネルとして機能します。                    |
| <a href="#">Amazon S3 製品情報</a> | Amazon S3 に関する情報のメインのウェブページです。                                       |
| <a href="#">お問い合わせ</a>         | AWS の支払い、アカウント設定その他に関する連絡先です。                                        |
| <a href="#">利用規約</a>           | Amazon.com、Amazon.co.jp およびその関連会社における著作権、商標、その他に関する規約について詳細に説明しています。 |

# 付録

## トピック

- [付録: SelectObjectContent レスポンス](#)
- [付録: OPTIONS オブジェクト](#)
- [付録: SOAP API](#)
- [付録: リクエストの認証 \(AWS 署名バージョン 2\)](#)
- [付録: ライフサイクル設定 API \(廃止\)](#)

# 付録: SelectObjectContent レスポンス

## 説明

Amazon S3 Select オペレーションは、シンプルな構造化クエリ言語 (SQL) ステートメントに基づいて Amazon S3 オブジェクトのコンテンツをフィルタリングします。このオペレーションのレスポンスサイズが不明な場合、Amazon S3 Select はレスポンスを一連のメッセージとしてストリーミングし、レスポンスに値として **chunked** を含む Transfer-Encoding ヘッダーを含めます。

Amazon S3 Select の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[オブジェクトからのコンテンツの選択](#)」を参照してください。

Amazon S3 Select で SQL を使用方法の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[Amazon S3 Select および Amazon Glacier Select の SQL リファレンス](#)」を参照してください。

## レスポンス

Amazon S3 Select オペレーションが成功すると、200 OK ステータスコードが返されます。

### レスポンスヘッダー

このオペレーションの実装はほとんどのレスポンスに共通のレスポンスヘッダーのみを使用します。詳細については、「[共通のレスポンスヘッダー](#)」を参照してください。

### レスポンス本文

この Amazon S3 Select のレスポンスサイズは不明なため、Amazon S3 はレスポンスを一連のメッセージとしてストリーミングし、レスポンスに値として **chunked** を含む Transfer-Encoding ヘッダーを含めます。次の例は、トップレベルのレスポンス形式を示しています。

```
<Message 1>
<Message 2>
<Message 3>
.....
<Message n>
```

各メッセージは、prelude と data の 2 つのセクションで構成されています。prelude セクションは、1) メッセージの全バイト長と、2) すべてのヘッダーの合計バイト長で構成されています。data セクションには、1) ヘッダーと 2) ペイロードが含まれています。

各セクションは、4 バイトのビッグエンディアン整数チェックサム (CRC) で終わります。Amazon S3 Select は、CRC32 (別名 GZIP CRC32) を使用して、両方の CRC を計算します。CRC32 の詳細については、「[GZIP ファイル形式仕様バージョン 4.3](#)」を参照してください。

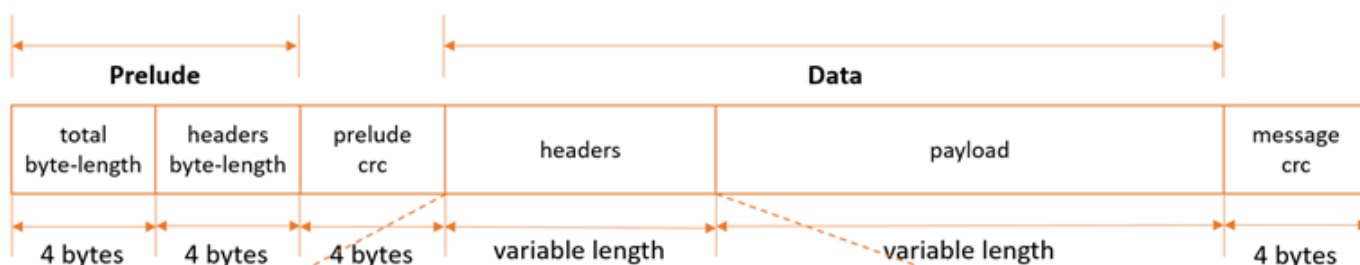
合計メッセージオーバーヘッド (prelude と両方のチェックサムを含む) は 16 バイトです。

### Note

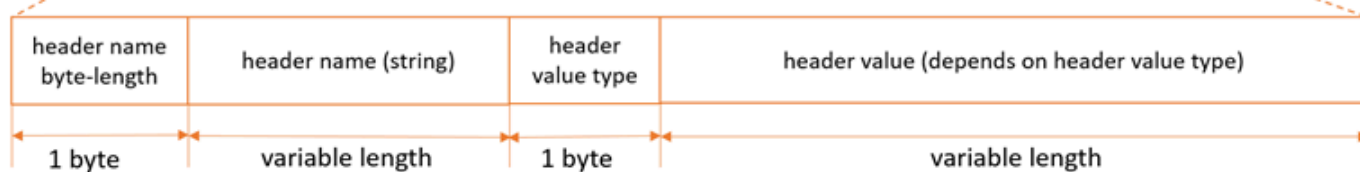
メッセージ内のすべての integer 値は、ネットワークバイト順、つまりビッグエンディアン順です。

次の図は、メッセージとヘッダーを構成するコンポーネントを示します。メッセージごとに複数のヘッダーがあることに注意してください。

#### Message:



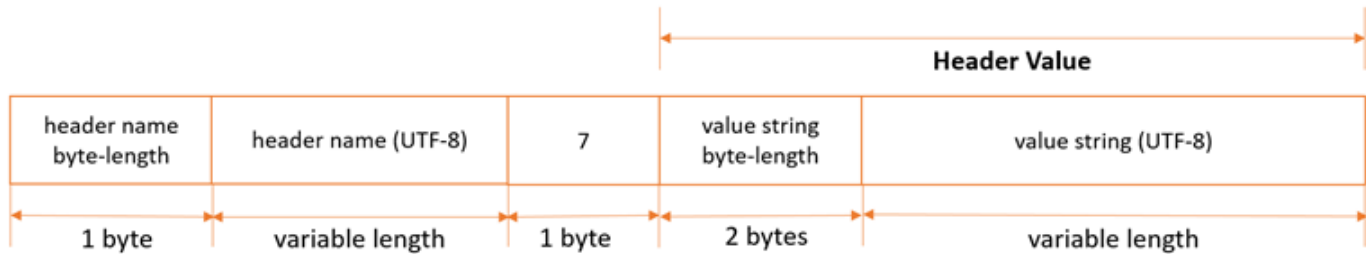
#### Headers (multiple headers per message):



### Note

Amazon S3 Select の場合、ヘッダー値の型は常に 7 (type=String) です。この型では、ヘッダー値は 2 バイトのビッグエンディアン整数長、およびそのバイト長の UTF-8 文字列の 2 つのコンポーネントで構成されます。次の図は、Amazon S3 Select ヘッダーを構成するコンポーネントを示しています。

## Amazon S3 Select Headers (type=String):



ペイロードのバイト長の計算 (この 2 つの計算は同等です):

- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - \text{sizeof}(\text{total\_length}) - \text{sizeof}(\text{header\_length}) - \text{sizeof}(\text{prelude\_crc}) - \text{sizeof}(\text{message\_crc})$
- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - 16$

各メッセージには、以下のコンポーネントが含まれます。

- Prelude: 常に 8 バイト の固定サイズ (4 バイト × 2 フィールド)。
  - 最初の 4 バイト: 全バイト長: メッセージ全体のビッグエンディアン整数バイト長 (4 バイトの合計長フィールド自体を含む)。
  - 次の 4 バイト: ヘッダーバイト長: メッセージのヘッダー部分のビッグエンディアン整数バイト長 (ヘッダー長フィールド自体を除く)。
- Prelude CRC: メッセージの prelude 部分 (CRC 自体を除く) の 4 バイトのビッグエンディアン整数チェックサム (CRC)。異常なバッファリング動作を引き起こすことなく、破損したバイト長の情報を即時に検出できるように、prelude にはメッセージ CRC とは別の CRC があります (下記を参照)。
- ヘッダー: メッセージの種類、ペイロードの形式など、メッセージに注釈を付けるメタデータのセット。メッセージには複数のヘッダーを含めることができるため、メッセージのこの部分はメッセージタイプに応じて異なるバイト長を持つことができます。ヘッダーはキーと値のペアで、キーと値はどちらも UTF-8 文字列です。ヘッダーは、メッセージのヘッダー部分に任意の順序で表示することができ、同じタイプのヘッダーは一度だけしか表示できません。

Amazon S3 Select の場合、以下はメッセージタイプに応じたヘッダー名と有効な値のセットのリストです。

- MessageType ヘッダー:

- HeaderName => ":message-type"
- 有効な HeaderValues => "error"、"event"
- EventType ヘッダー:
  - HeaderName => ":event-type"
  - 有効な HeaderValues => "Records"、"Cont"、"Progress"、"Stats"、"End"
- ErrorCode ヘッダー:
  - HeaderName => ":error-code"
  - 有効な HeaderValues => [SELECT Object Content エラーコードのリスト](#) セクションのテーブルからのエラーコード。
- ErrorMessage ヘッダー:
  - HeaderName => ":error-message"
  - 有効な HeaderValues => リクエストレベルのエラーの診断に役立つ、サービスによって返されるエラーメッセージ。
- ペイロード: 何でも構いません。
- メッセージ CRC: メッセージの先頭からチェックサム先頭までの 4 バイトのビッグエンディアン整数チェックサム (CRC) (つまり、メッセージ CRC 自体を除くメッセージ内のすべて)。

各ヘッダーには、以下のコンポーネントが含まれます。メッセージごとに複数のヘッダーが存在する場合があります。

- ヘッダー名のバイト長: ヘッダー名のバイトの長さ。
- ヘッダー名: ヘッダーのタイプを示すヘッダーの名前。有効な値: ":message-type" ":event-type" ":error-code" ":error-message"
- ヘッダー値のタイプ: ヘッダー値のタイプを示す列挙型。Amazon S3 Select の場合、これは常に 7 です。
- 値の文字列のバイト長: (Amazon S3 Select の場合) ヘッダー値の文字列のバイト長。
- ヘッダー値文字列: (Amazon S3 Select の場合) ヘッダー値の文字列のバイト長。このフィールドの有効な値は、ヘッダーのタイプによって異なります。各ヘッダーのタイプとメッセージタイプの有効な値については、以下のセクションを参照してください。

Amazon S3 Select の場合、レスポンスは次のタイプのメッセージになります。



- レコードメッセージ: 単一のレコード、部分的なレコード、または複数のレコードが含まれる場合があります。結果のサイズに応じて、レスポンスにはこれらのメッセージが 1 つ以上含まれる場合があります。
- 継続メッセージ: Amazon S3 は、TCP 接続を維持するため、定期的にこのメッセージを送信します。これらのメッセージは、レスポンスにランダムに表示されます。クライアントはメッセージタイプを検出し、それに応じて処理する必要があります。
- 進行状況メッセージ: Amazon S3 は、リクエストに応じて定期的にこのメッセージを送信します。これには、開始されたものの、まだ完了していないクエリの進行状況に関する情報が含まれます。
- 統計情報メッセージ: Amazon S3 は、リクエストの最後にこのメッセージを送信します。これには、クエリに関する統計情報が含まれます。
- 終了メッセージ: リクエストが完了し、それ以上メッセージが送信されないことを示します。クライアントが End メッセージを受信するまで、リクエストが完了したとみなさないでください。
- RequestLevelError メッセージ: Amazon S3 は、何らかの理由でリクエストが失敗した場合にこのメッセージを送信します。このメッセージには、失敗した際のエラーコードとエラーメッセージが含まれています。Amazon S3 が RequestLevelError メッセージを送信する場合、End メッセージは送信されません。

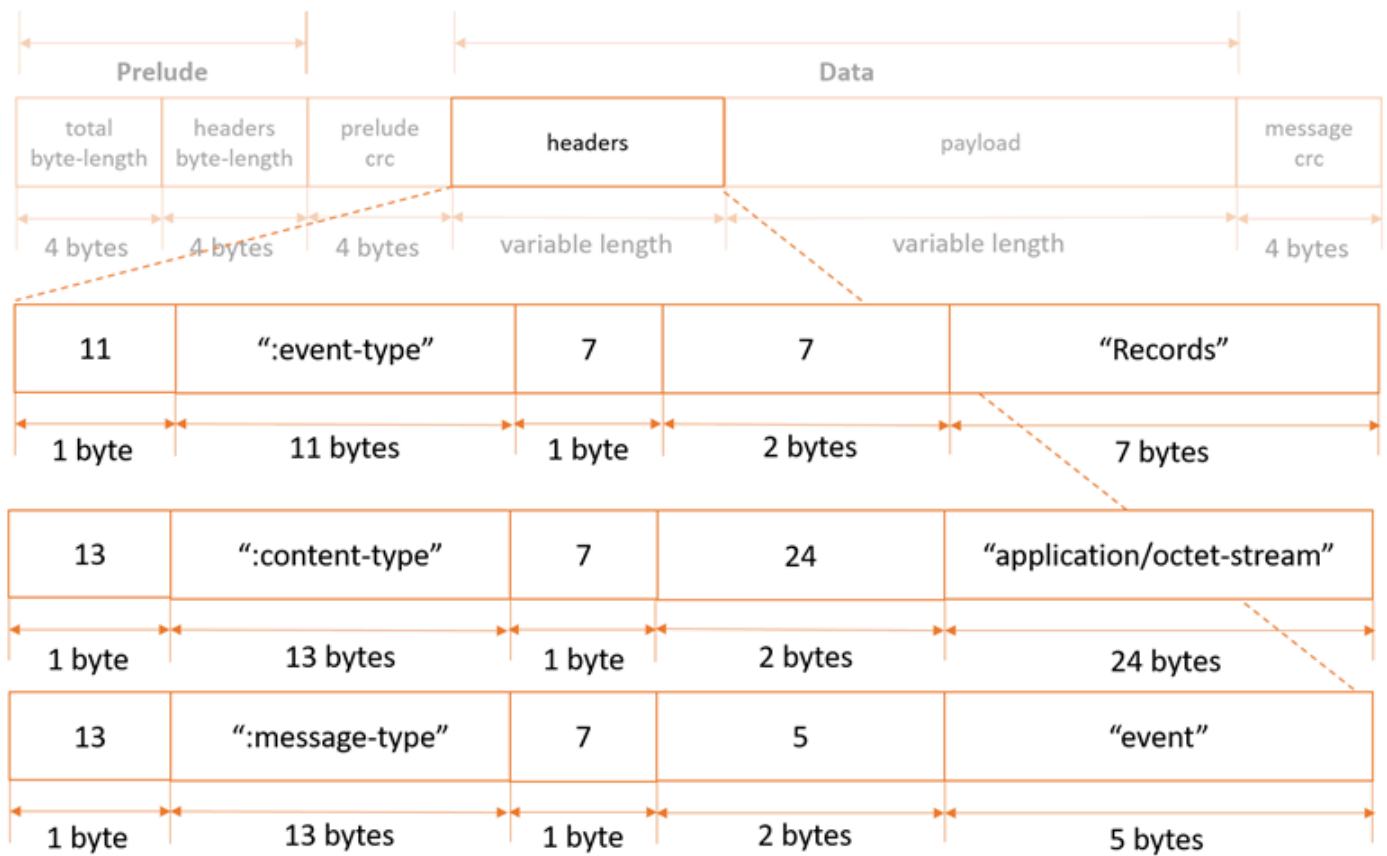
以下のセクションでは、各メッセージタイプの構造について詳しく説明します。

このプロトコルを使用するサンプルコードとユニットテストについては、GitHub ウェブサイトの「[AWS C Event Stream](#)」を参照してください。

## レコードメッセージ

### ヘッダー仕様

レコードメッセージには、以下のように 3 つのヘッダーが含まれています。



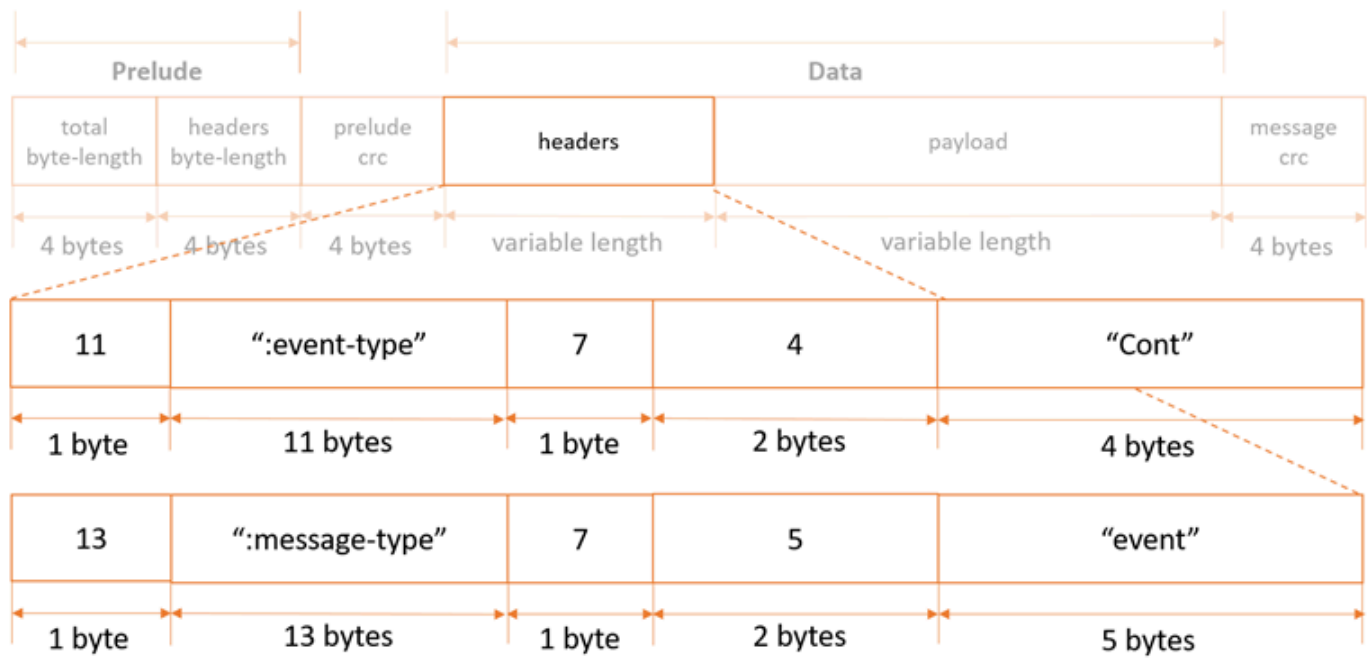
## ペイロード仕様

レコードメッセージのペイロードには、単一のレコード、部分的なレコード、または複数のレコードが含まれる場合があります。

## 継続メッセージ

## ヘッダー仕様

継続メッセージには、以下のように 2 つのヘッダーが含まれています。



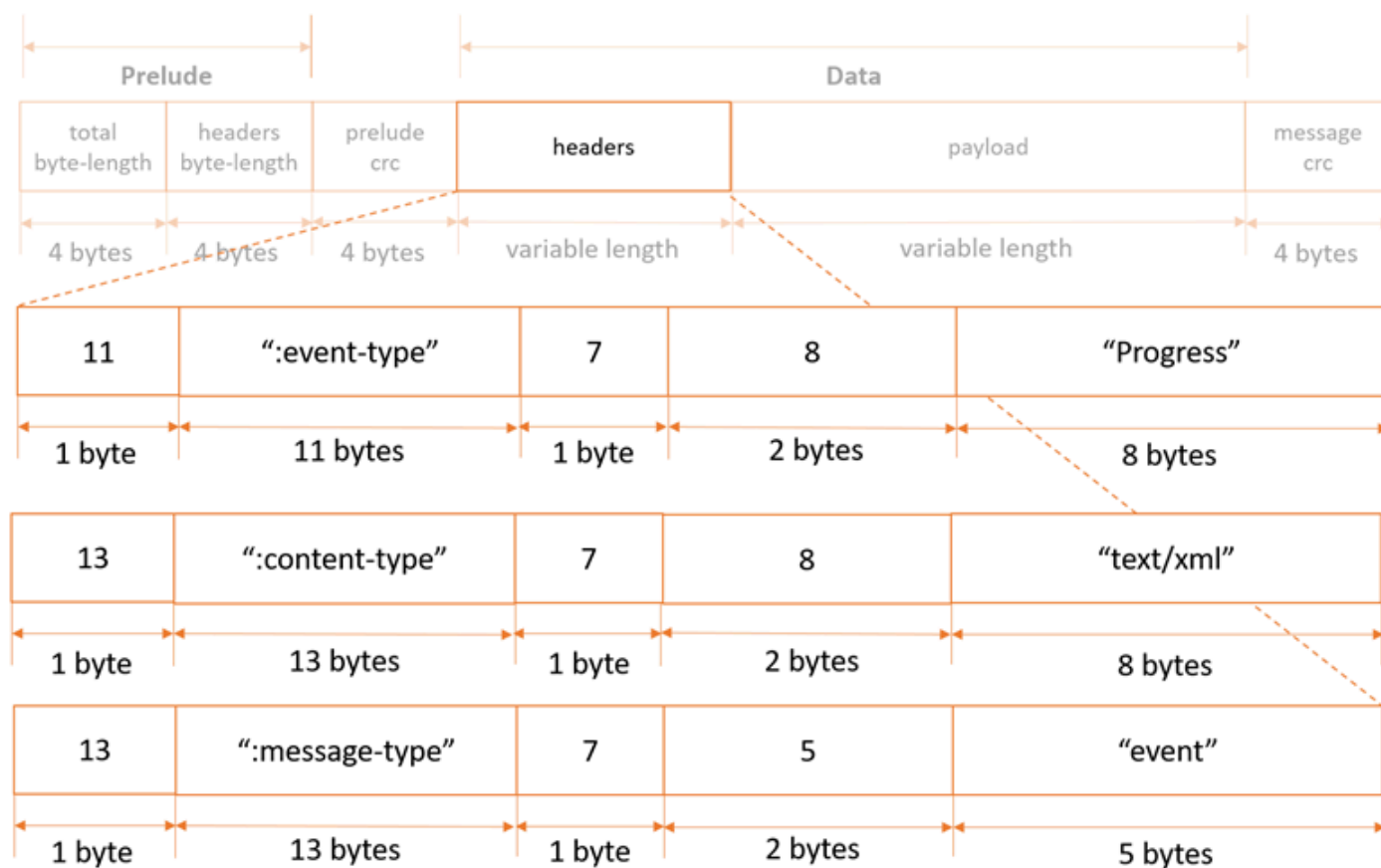
## ペイロード仕様

継続メッセージにはペイロードがありません。

## 進行状況メッセージ

## ヘッダー仕様

進行状況メッセージには、以下のように 3 つのヘッダーが含まれています。



## ペイロード仕様

進行状況メッセージのペイロードは、リクエストの進行状況に関する情報を含む XML ドキュメントです。

- BytesScanned => 圧縮解除前に処理されたバイト数 (ファイルが圧縮されている場合)。
- BytesProcessed => 圧縮解除後に処理されたバイト数 (ファイルが圧縮されている場合)。
- BytesReturned => Amazon S3 によって返されるレコードペイロードデータの現在のバイト数。

非圧縮ファイルの場合、BytesScanned と BytesProcessed は等しくなります。

例:

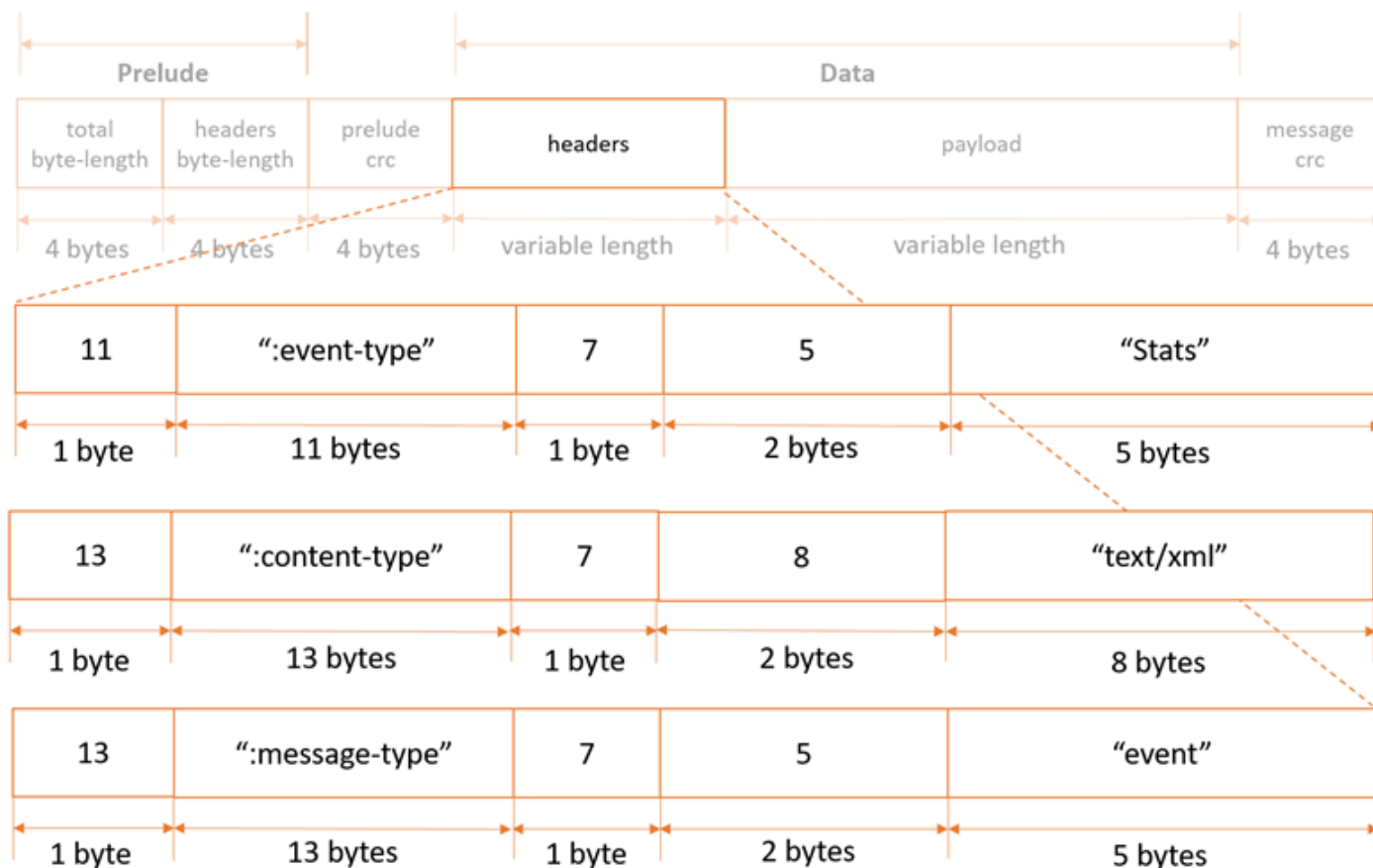
```
<?xml version="1.0" encoding="UTF-8"?>
<Progress>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
```

```
<BytesReturned>1024</BytesReturned>
</Progress>
```

## 統計情報メッセージ

### ヘッダー仕様

統計情報メッセージには、以下のように 3 つのヘッダーが含まれています。



### ペイロード仕様

統計情報メッセージのペイロードは、処理が完了した時点でのリクエストの統計に関する情報を含む XML ドキュメントです。

- BytesScanned => 圧縮解除前に処理されたバイト数 (ファイルが圧縮されている場合)。
- BytesProcessed => 圧縮解除後に処理されたバイト数 (ファイルが圧縮されている場合)。
- BytesReturned => Amazon S3 によって返されるレコードペイロードデータの合計バイト数。

非圧縮ファイルの場合、BytesScanned と BytesProcessed は等しくなります。

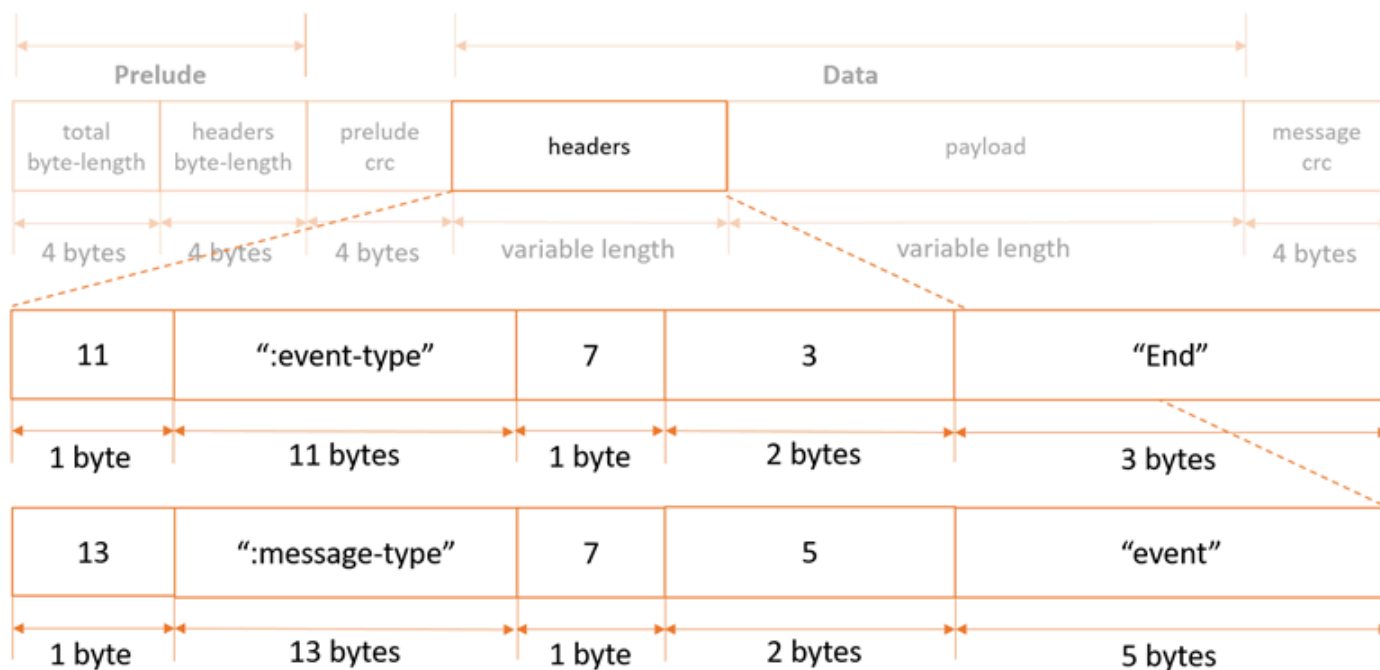
例:

```
<?xml version="1.0" encoding="UTF-8"?>
<Stats>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Stats>
```

## 終了メッセージ

### ヘッダー仕様

終了メッセージには、以下のように 2 つのヘッダーが含まれています。



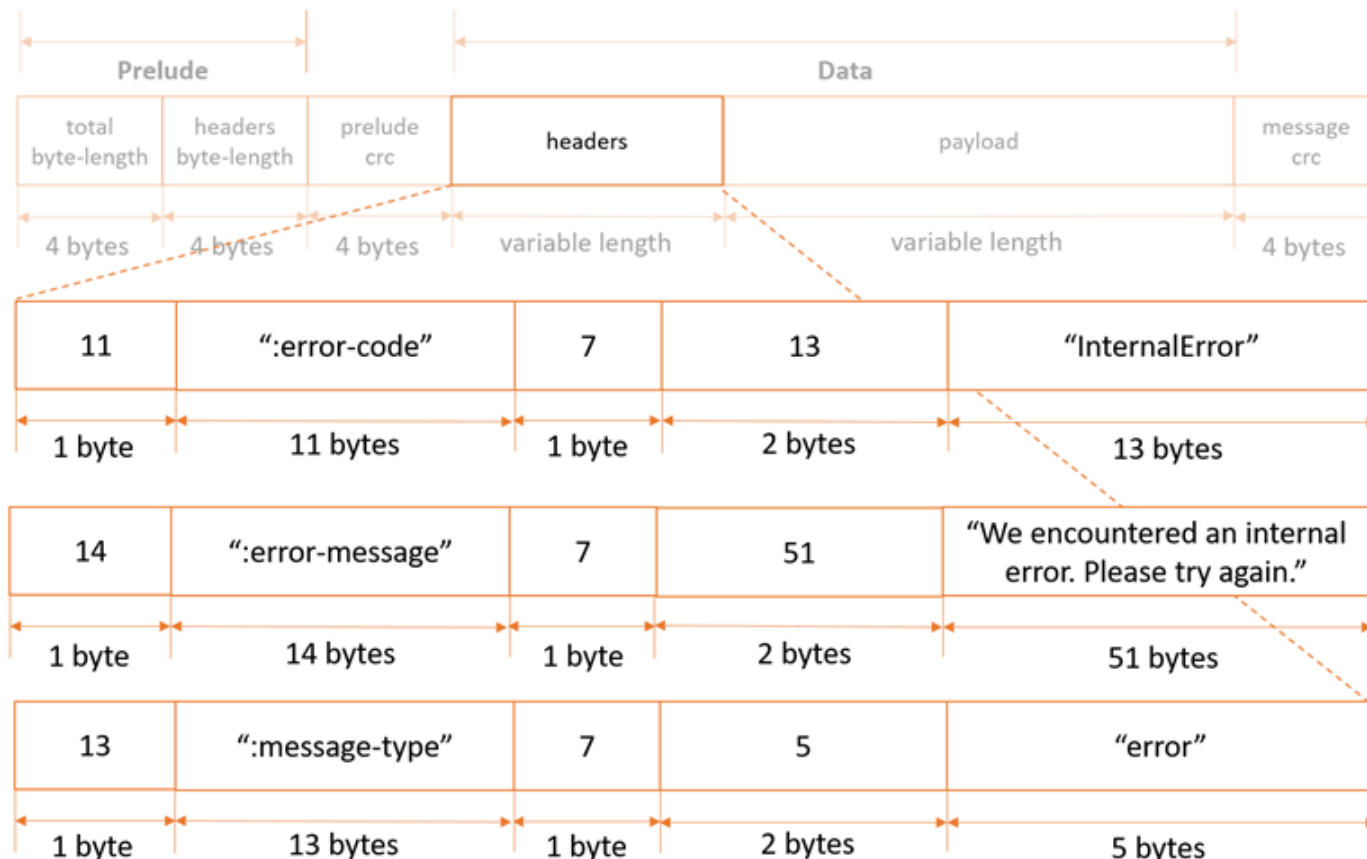
### ペイロード仕様

終了メッセージにはペイロードがありません。

## リクエストレベルのエラーメッセージ

### ヘッダー仕様

リクエストレベルのエラーメッセージには、以下のように 3 つのヘッダーが含まれています。



考えられるエラーコードとエラーメッセージのリストについては、「[SELECT Object Content エラーコードのリスト](#)」を参照してください。

### ペイロード仕様

リクエストレベルのエラーメッセージにはペイロードがありません。

## 関連リソース

- [SelectObjectContent](#)
- [GetObject](#)
- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)





## 付録: OPTIONS オブジェクト

### 説明

ブラウザは、このプリフライトリクエストを Amazon S3 に送信して、特定のオリジン、HTTP メソッド、ヘッダーを使用して実際のリクエストを送信できるかどうかを判断できます。

Amazon S3 は、バケットに cors サブリソースを追加できるようにすることで、クロスオリジンリソース共有 (CORS) をサポートしています。ブラウザがこのプリフライトリクエストを送信すると、Amazon S3 は cors 設定で定義されたルールを評価して応答します。

バケットで cors が有効になっていない場合、Amazon S3 は 403 Forbidden レスポンスを返します。

CORS の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[Cross-Origin Resource Sharing の有効化](#)」を参照してください。

### リクエスト

#### 構文

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

### リクエストパラメータ

このオペレーションでは、特定のリクエストパラメータは導入されませんが、実際のリクエストに必要なリクエストパラメータが含まれる場合があります。

#### リクエストヘッダー

| 名前     | 説明                                                                                  | 必須 |
|--------|-------------------------------------------------------------------------------------|----|
| Origin | Amazon S3 へのクロスオリジンリクエストの発信元を識別します。例えば、http://www.example.com などです。<br><br>タイプ: 文字列 | はい |

| 名前                             | 説明                                                                                                                                                                                                           | 必須  |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|                                | デフォルト: なし                                                                                                                                                                                                    |     |
| Access-Control-Request-Method  | <p>実際のリクエストで使用される HTTP メソッドを識別します。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p>                                                                                                                                      | はい  |
| Access-Control-Request-Headers | <p>実際のリクエストで送信される HTTP ヘッダーのカンマ区切りリスト。</p> <p>例えば、サーバー側の暗号化を使用してオブジェクトを配置する場合、このプリフライトリクエストは、リクエストに <code>x-amz-server-side-encryption</code> ヘッダーを含めることができるかどうかを判断します。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> | いいえ |

## リクエストの要素

このオペレーションの実装では、リクエストパラメータは使用しません。

## レスポンス

### レスポンスヘッダー

| ヘッダー                        | 説明                                                                        |
|-----------------------------|---------------------------------------------------------------------------|
| Access-Control-Allow-Origin | リクエストで送信されたオリジン。リクエストで指定されたオリジンが許可されていない場合、Amazon S3 はレスポンスにこのヘッダーを含めません。 |

| ヘッダー                          | 説明                                                                                                                                                                          |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                               | タイプ: 文字列                                                                                                                                                                    |
| Access-Control-Max-Age        | <p>プリフライトリクエストの結果をキャッシュできる時間 (秒単位)。</p> <p>タイプ: 文字列</p>                                                                                                                     |
| Access-Control-Allow-Methods  | <p>元のリクエストで送信された HTTP メソッド。リクエストのメソッドが許可されていない場合、Amazon S3 はレスポンスにこのヘッダーを含めません。</p> <p>タイプ: 文字列</p>                                                                         |
| Access-Control-Allow-Headers  | <p>ブラウザが実際のリクエストで送信できる HTTP ヘッダーのカンマ区切りリスト。リクエストされたヘッダーのいずれかが許可されていない場合、Amazon S3 はレスポンスにそのヘッダーを含めません。また、レスポンスには Access-Control プレフィックスが付いたヘッダーは含まれません。</p> <p>タイプ: 文字列</p> |
| Access-Control-Expose-Headers | <p>HTTP ヘッダーのカンマ区切りリスト。このヘッダーにより、JavaScript クライアントは実際のリクエストへのレスポンスでこれらのヘッダーにアクセスできるようになります。</p> <p>タイプ: 文字列</p>                                                             |

## レスポンス要素

このオペレーションの実装では、レスポンス要素は返されません。

## 例

### 例: **cors** が有効なバケットにプリフライト OPTIONS リクエストを送信する

ブラウザは、このプリフライトリクエストを Amazon S3 に送信して、<http://www.example.com> オリジンから `examplebucket` という名前の Amazon S3 バケットに実際の PUT リクエストを送信できるかどうかを判断できます。

#### リクエスト例

```
OPTIONS /exampleobject HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

#### レスポンス例

```
HTTP/1.1 200 OK
x-amz-id-2: 6SvaESv3VULYPLik5LL171SPPtSnBvDdGmnk1X1HfU17uS2m1DF6td6KWKNjYMXZ
x-amz-request-id: BDC4B83DF5096BBE
Date: Wed, 21 Aug 2012 23:09:55 GMT
Etag: "1f1a1af1f111111111111111c11aed1da1"
Access-Control-Allow-Origin: http://www.example.com
Access-Control-Allow-Methods: PUT
Access-Control-Expose-Headers: x-amz-request-id
Content-Length: 0
Server: AmazonS3
```

## 関連リソース

- [GetBucketCors](#)
- [DeleteBucketCors](#)
- [PutBucketCors](#)

## 付録: SOAP API

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

このセクションでは、サービス、バケット、およびオブジェクトオペレーションに関する SOAP API について説明します。SOAP リクエストは、認証済みの場合も匿名の場合も、SSL を使用して Amazon S3 に送信する必要があることに注意してください。HTTP を介して SOAP リクエストが送信された場合、Amazon S3 はエラーを返します。

最新の Amazon S3 WSDL は、[docs.aws.amazon.com/2006-03-01/AmazonS3.wSDL](https://docs.aws.amazon.com/2006-03-01/AmazonS3.wSDL) で入手できます。

### トピック

- [サービスに対するオペレーション \(SOAP API\)](#)
- [バケットに対するオペレーション \(SOAP API\)](#)
- [オブジェクトに対するオペレーション \(SOAP API\)](#)
- [SOAP リクエストの認証方法](#)
- [SOAP のアクセスポリシーの設定](#)
- [一般的な要素](#)
- [SOAP エラーレスポンス](#)

## サービスに対するオペレーション (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

このセクションでは、Amazon S3 サービスで実行できるオペレーションについて説明します。

### トピック

- [ListAllMyBuckets \(SOAP API\)](#)

## ListAllMyBuckets (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

ListAllMyBuckets オペレーションは、リクエスト送信者が所有するすべてのバケットのリストを返します。

### Example

#### Sample Request

```
<ListAllMyBuckets xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListAllMyBuckets>
```

#### Sample Response

```
<ListAllMyBucketsResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Owner>
    <ID>bcaf1ffd86f41161ca5fb16fd081034f</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes</Name>
      <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>samples</Name>
      <CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
    </Bucket>
  </Buckets>
```

```
</ListAllMyBucketsResult>
```

## レスポンス本文

- Owner:

これは、Amazon S3 が認証とアクセスコントロールの目的でユーザーの ID を表すために使用する情報を提供します。ID は、リクエストを行った開発者を識別する、一意で永続的な識別子です。DisplayName は、リクエストを行った開発者を表す、人間が読める名前です。これは一意ではなく、時間の経過とともに変更される可能性があります。DisplayName はフォーラム名と一致させることをお勧めします。

- Name:

バケットの名前。バケットの 1 つが最近削除された場合でも、削除されたバケットの名前は一定期間このリストに表示される可能性があることに注意してください。

- CreationDate:

バケットが作成された時間。

## アクセスコントロール

有効な AWS アクセスキー ID で認証する必要があります。匿名リクエストでバケットを一覧表示することは一切許可されず、自分が所有者であるバケットのみ一覧表示できます。

## バケットに対するオペレーション (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

このセクションでは、Amazon S3 バケットで実行できるオペレーションについて説明します。

### トピック

- [CreateBucket \(SOAP API\)](#)
- [DeleteBucket \(SOAP API\)](#)

- [ListBucket \(SOAP API\)](#)
- [GetBucketAccessControlPolicy \(SOAP API\)](#)
- [SetBucketAccessControlPolicy \(SOAP API\)](#)
- [GetBucketLoggingStatus \(SOAP API\)](#)
- [SetBucketLoggingStatus \(SOAP API\)](#)

## CreateBucket (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

CreateBucket オペレーションは、バケットを作成します。すべての文字列がバケット名として使用できるわけではありません。バケットの命名に関する制限については、「[Amazon S3 バケットの使用](#)」を参照してください。

### Note

バケット名が存在するかどうかを判断するには、ListBucket を使用し、MaxKeys を 0 に設定します。NoSuchBucket レスポンスは、バケットが使用可能であることを示します。AccessDenied レスポンスは、他のユーザーがバケットを所有していることを示します。Success レスポンスは、自分がバケットを所有しているか、バケットへのアクセス許可を持っていることを示します。

Example 「quotes」という名前のバケットを作成する

### Sample Request

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```



## Sample Response

```
<CreateBucketResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <CreateBucketResponse>
    <Bucket>quotes</Bucket>
  </CreateBucketResponse>
</CreateBucketResponse>
```

### 要素

- **Bucket**: 作成しようとしているバケットの名前。
- **AccessControlList**: 新しいバケットのアクセスコントロールリスト。この要素はオプションです。指定しない場合、バケットはリクエストに FULL\_CONTROL アクセスを許可するアクセスポリシーで作成されます。

### アクセスコントロール

有効な AWS アクセスキー ID で認証する必要があります。匿名リクエストでバケットを作成することは一切許可されていません。

### 関連リソース

- [ListBucket \(SOAP API\)](#)

## DeleteBucket (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

DeleteBucket オペレーションはバケットを削除します。バケット自体を削除する前に、バケット内のすべてのオブジェクトを削除する必要があります。

### Example

この例では、「quotes」バケットを削除します。

## Sample Request

```
<DeleteBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteBucket>
```

## Sample Response

```
<DeleteBucketResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <DeleteBucketResponse>
    <Code>204</Code>
    <Description>No Content</Description>
  </DeleteBucketResponse>
</DeleteBucketResponse>
```

### 要素

- Bucket: 削除するバケットの名前。

### アクセスコントロール

バケットのアクセスコントロールポリシーに関係なく、バケットの所有者のみがバケットを削除できます。

## ListBucket (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

ListBucket オペレーションは、バケット内の一部の項目に関する情報を返します。

リストオペレーションの一般的な概要については、「[オブジェクトキーのリスト化](#)」を参照してください。

## リクエスト

この例では、プレフィックス「notes」を持つ「quotes」バケット内のキー最大 1,000 個まで一覧表示します。

### 構文

```
<ListBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Prefix>notes/</Prefix>
  <Delimiter></Delimiter>
  <MaxKeys>1000</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

### パラメータ

| 名前     | 説明                                                                                                                                                       | 必須  |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| prefix | <p>指定されたプレフィックスで始まるキーのみにレスポンスを制限します。プレフィックスを使用すると、ファイルシステムがフォルダを使用するのと同様の方法で、バケットを異なるキーセットに分割できます。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p>                  | いいえ |
| marker | <p>バケット内でリスト表示を開始する場所を示します。リストには、マーカの後に辞書順で出現するキーのみが含まれます。これはページ分割に便利です。結果の次のページを取得するには、現在のページの最後のキーをマーカースとして使用します。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> | いいえ |

| 名前        | 説明                                                                                                                                                 | 必須  |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| max-keys  | レスポンス本文に表示するキーの最大数。サーバーは、この数よりも少ないキーを返す場合がありますが、それ以上の数は返しません。<br><br>タイプ: 文字列<br><br>デフォルト: なし                                                     | いいえ |
| delimiter | プレフィックスと区切り文字の最初の出現箇所の間に同じ文字列を含むキーを、CommonPrefixes コレクション内の単一の結果要素にロールアップします。ロールアップされたこれらのキーは、レスポンスの他の箇所では返されません。<br><br>タイプ: 文字列<br><br>デフォルト: なし | いいえ |

## 成功時のレスポンス

このレスポンスは、バケットに次のキーが含まれていることを前提としています。

```
notes/todos.txt
notes/2005-05-23/customer_mtg_notes.txt
notes/2005-05-23/phone_notes.txt
notes/2005-05-28/sales_notes.txt
```

## 構文

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>backups</Name>
  <Prefix>notes</Prefix>
  <MaxKeys>1000</MaxKeys>
  <Delimiter></Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>notes/todos.txt</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
```

```
<Size>5126</Size>
<StorageClass>STANDARD</StorageClass>
<Owner>
  <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
  <DisplayName>webfile</DisplayName>
</Owner>
<StorageClass>STANDARD</StorageClass>
</Contents>
<CommonPrefixes>
  <Prefix>notes/2005-05-23/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
  <Prefix>notes/2005-05-28/</Prefix>
</CommonPrefixes>
</ListBucketResult>
```

ご覧のとおり、レスポンスの多くのフィールドはリクエストパラメータをエコーします。IsTruncated、Contents、および CommonPrefixes は、新しい情報を含めることができる唯一のレスポンス要素です。

レスポンス要素

| 名前             | 説明                                                                                                                                                                                                                                                                                                                                  |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Contents       | 返された各オブジェクトに関するメタデータ。<br><br>型: XML メタデータ<br><br>祖先: ListBucketResult                                                                                                                                                                                                                                                               |
| CommonPrefixes | delimiter を指定した場合のみ、レスポンスに CommonPrefixes を含めることができます。これを行うと、CommonPrefixes には、Prefix と delimiter で指定された文字列の次の出現箇所との間のすべてのキー (存在する場合) が含まれます。実際には、CommonPrefixes は Prefix で指定されたディレクトリ内のサブディレクトリのように動作するキーを一覧表示します。例えば、prefix が notes/ で、delimiter がスラッシュ (/) の場合、notes/summer/july では、共通プレフィックスは notes/summer/ です。<br><br>タイプ: 文字列 |

| 名前          | 説明                                                                                                                                                                   |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | 祖先: ListBucketResult                                                                                                                                                 |
| Delimiter   | <p>プレフィックスと区切り文字の最初の出現箇所の間に同じ文字列を含むキーを、CommonPrefixes コレクション内の単一の結果要素にロールアップします。ロールアップされたこれらのキーは、レスポンスの他の箇所では返されません。</p> <p>タイプ: 文字列</p> <p>祖先: ListBucketResult</p> |
| IsTruncated | <p>すべての結果が返されたか (true) または返されなかったか (false) どうかを指定します。結果の数が MaxKeys で指定した数を超えた場合、すべての結果が返されないことがあります。</p> <p>タイプ: 文字列</p> <p>祖先: boolean</p>                          |
| Marker      | <p>バケット内でリスト表示を開始する場所を示します。</p> <p>タイプ: 文字列</p> <p>祖先: ListBucketResult</p>                                                                                          |
| MaxKeys     | <p>レスポンス本文で返されるキーの最大数。</p> <p>タイプ: 文字列</p> <p>祖先: ListBucketResult</p>                                                                                               |
| Name        | <p>バケットの名前。</p> <p>タイプ: 文字列</p> <p>祖先: ListBucketResult</p>                                                                                                          |

| 名前     | 説明                                                              |
|--------|-----------------------------------------------------------------|
| Prefix | 指定されたプレフィックスで始まるキー。<br><br>タイプ: 文字列<br><br>祖先: ListBucketResult |

## レスポンス本文

リストレスポンスの詳細については、「[キーレスポンスのリスト化](#)」を参照してください。

## アクセスコントロール

バケットのキーを一覧表示するには、そのバケットに対する READ アクセス権が付与されている必要があります。

## GetBucketAccessControlPolicy (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

GetBucketAccessControlPolicy オペレーションは、バケットのアクセスコントロールポリシーを取得します。

## Example

この例では、「quotes」バケットのアクセスコントロールポリシーを取得します。

## Sample Request

```
<GetBucketAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetBucketAccessControlPolicy>
```

## Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd2441bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6f41bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

## レスポンス本文

レスポンスには、バケットのアクセスコントロールポリシーが含まれます。このレスポンスの説明については、「[SOAP アクセスポリシー](#)」を参照してください。

## アクセスコントロール

バケットのアクセスコントロールポリシーを取得するには、バケットに対する READ\_ACP 権限が必要です。

## SetBucketAccessControlPolicy (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。



SetBucketAccessControlPolicy オペレーションは、既存のバケットのアクセスコントロールポリシーを設定します。成功すると、バケットの以前のアクセスコントロールポリシーは、指定されたアクセスコントロールポリシーに完全に置き換えられます。

## Example

指定されたユーザー (通常は所有者) に「quotes」バケットへの FULL\_CONTROL アクセスを許可します。

## Sample Request

```
<SetBucketAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b8863000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</SetBucketAccessControlPolicy >
```

## Sample Response

```
<GetBucketAccessControlPolicyResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetBucketAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </GetBucketAccessControlPolicyResponse>
</GetBucketAccessControlPolicyResponse>
```

## アクセスコントロール

バケットのアクセスコントロールポリシーを設定するには、バケットに対する WRITE\_ACP 権限が必要です。

## GetBucketLoggingStatus (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

GetBucketLoggingStatus は、既存のバケットのログ記録ステータスを取得します。

この機能の一般的な概要については、「[サーバーログ](#)」を参照してください。

### Example

#### Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
    <soap:Body>
      <GetBucketLoggingStatus xmlns="https://doc.s3.amazonaws.com/2006-03-01">
        <Bucket>mybucket</Bucket>
        <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
        <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
        <Signature>YOUR_SIGNATURE_HERE</Signature>
      </GetBucketLoggingStatus>
    </soap:Body>
  </soap:Envelope>
```

#### Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <GetBucketLoggingStatusResponse xmlns="https://s3.amazonaws.com/
doc/2006-03-01">
```

```
<GetBucketLoggingStatusResponse>
  <LoggingEnabled>
    <TargetBucket>mylogs</TargetBucket>
    <TargetPrefix>mybucket-access_log-</TargetPrefix>
  </LoggingEnabled>
</GetBucketLoggingStatusResponse>
</GetBucketLoggingStatusResponse>
</soapenv:Body>
</soapenv:Envelope>
```

## アクセスコントロール

このオペレーションを呼び出すことができるのは、バケットの所有者のみです。

## SetBucketLoggingStatus (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

SetBucketLoggingStatus オペレーションは、既存のバケットのログ記録ステータスを更新します。

この機能の一般的な概要については、「[サーバーログ](#)」を参照してください。

### Example

このサンプルリクエストは、「mybucket」バケットのサーバーアクセスのログ記録を有効にし、ログが「access\_log-」のプレフィックスで「mylogs」に配信されるように設定します。

### Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
    <soap:Body>
      <SetBucketLoggingStatus xmlns="https://doc.s3.amazonaws.com/2006-03-01">
```

```
<Bucket>myBucket</Bucket>
<AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>YOUR_SIGNATURE_HERE</Signature>
<BucketLoggingStatus>
  <LoggingEnabled>
    <TargetBucket>mylogs</TargetBucket>
    <TargetPrefix>mybucket-access_log-</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
</SetBucketLoggingStatus>
</soap:Body>
:</soap:Envelope>
```

## Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <SetBucketLoggingStatusResponse xmlns="https://s3.amazonaws.com/
doc/2006-03-01"/>
    </soapenv:Body>
  </soapenv:Envelope>
```

## アクセスコントロール

このオペレーションを呼び出すことができるのは、バケットの所有者のみです。

## オブジェクトに対するオペレーション (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

このセクションでは、Amazon S3 オブジェクトで実行できるオペレーションについて説明します。

## トピック

- [PutObjectInline \(SOAP API\)](#)
- [PutObject \(SOAP API\)](#)
- [CopyObject \(SOAP API\)](#)
- [GetObject \(SOAP API\)](#)
- [GetObjectExtended \(SOAP API\)](#)
- [DeleteObject \(SOAP API\)](#)
- [GetObjectAccessControlPolicy \(SOAP API\)](#)
- [SetObjectAccessControlPolicy \(SOAP API\)](#)

## PutObjectInline (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

PutObjectInline オペレーションはバケットにオブジェクトを追加します。オブジェクトのデータは、SOAP メッセージの本文に含まれています。

バケット内に既にオブジェクトが存在する場合、Amazon S3 は最後の書き込みリクエストを保存しているため、新しいオブジェクトはそれを上書きします。ただし、Amazon S3 は分散システムです。Amazon S3 が同じオブジェクトに対する複数の書き込みリクエストをほぼ同時に受信した場合、最終的に優先されるのは 1 つだけだとしても、すべてのオブジェクトが保存される可能性があります。Amazon S3 は Object Lock 機能を提供していません。必要な場合は、必ずアプリケーションレイヤーに構築してください。

オブジェクトがネットワーク上で破損しないようにするには、オブジェクトの MD5 を計算し、Amazon S3 に PUT して、返された Etag を計算された MD5 値と比較します。

PutObjectInline は、大きなオブジェクトでの使用には適していません。このオペレーションは、1 MB 以下のオブジェクトを操作する場合に限定されています。データパラメータが 1 MB を超えるオ

オブジェクトをエンコードすると、PutObjectInline は InlineDataTooLargeError ステータスコードで失敗します。大きなオブジェクトをアップロードするには、代わりに非インライン PutObject API または REST API の使用を検討してください。

## Example

この例では、「quotes」バケットの「Nelson」オブジェクトにテキストとメタデータを書き込み、ユーザー (通常は所有者) にオブジェクトへの FULL\_CONTROL アクセスを許可し、匿名のユーザーでもオブジェクトを読み取れるようにします。

## Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fde241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
```

```
</PutObjectInline>
```

## Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <LastModified>2006-01-01T12:00:00.000Z</lastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

## 要素

- **Bucket**: オブジェクトを追加するバケット。
- **Key**: オブジェクトに割り当てるキー。
- **Metadata**: メタデータ要素で名前と値のメタデータのペアを指定できます。これらはオブジェクトと共に保存されます。
- **Data**: データの Base64 エンコード形式。
- **ContentLength**: データの長さ (バイト単位)。
- **AccessControlList**: リソースに対するアクセスコントロールリスト。この要素はオプションです。省略すると、リクエストにオブジェクトへの FULL\_CONTROL アクセスが付与されます。オブジェクトが既に存在する場合、既存のアクセスコントロールポリシーが置き換えられます。

## レスポンス

- **ETag**: エンティティタグは、GetObjectExtended を使用してオブジェクトの条件付きフェッチを実行するために使用できる、オブジェクトの MD5 ハッシュです。ETag は、オブジェクトのコンテンツに加えた変更のみを反映し、メタデータに加えた変更は反映しません。
- **LastModified**: 保存されたオブジェクトの Amazon S3 タイムスタンプ。

## アクセスコントロール

オブジェクトをバケットに配置するには、バケットへの WRITE アクセス権が必要です。

## 関連リソース

- [PutObject \(SOAP API\)](#)
- [CopyObject \(SOAP API\)](#)

## PutObject (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

PutObject オペレーションはバケットにオブジェクトを追加します。オブジェクトのデータは DIME 添付ファイルとしてアタッチされます。

オブジェクトがネットワーク上で破損しないようにするには、オブジェクトの MD5 を計算し、Amazon S3 に PUT して、返された Etag を計算された MD5 値と比較します。

バケット内に既にオブジェクトが存在する場合、Amazon S3 は最後の書き込みリクエストを保存しているため、新しいオブジェクトはそれを上書きします。ただし、Amazon S3 は分散システムです。Amazon S3 が同じオブジェクトに対する複数の書き込みリクエストをほぼ同時に受信した場合、最終的に優先されるのは 1 つだけだとしても、すべてのオブジェクトが保存される可能性があります。Amazon S3 は Object Lock 機能を提供していません。必要な場合は、必ずアプリケーションレイヤーに構築してください。

## Example

この例では、「quotes」バケットの「Nelson」オブジェクトにデータとメタデータを配置し、ユーザー (通常は所有者) にオブジェクトへの FULL\_CONTROL アクセスを許可し、匿名のユーザーでもオブジェクトを読み取れるようにします。このサンプルでは、実際の添付ファイルは表示されません。

## Sample Request

```
<PutObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
```



```

    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2007-05-11T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObject>

```

## Sample Response

```

<PutObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2006-03-01T12:00:00.183Z</LastModified>
  </PutObjectResponse>
</PutObjectResponse>

```

## 要素

- **Bucket:** オブジェクトを追加するバケット。
- **Key:** オブジェクトに割り当てるキー。
- **Metadata:** メタデータ要素で名前と値のメタデータのペアを指定できます。これらはオブジェクトと共に保存されます。

- **ContentLength**: データの長さ (バイト単位)。
- **AccessControlList**: リソースに対するアクセスコントロールリスト。この要素はオプションです。省略すると、リクエストにオブジェクトへの **FULL\_CONTROL** アクセスが付与されます。オブジェクトが既に存在する場合、既存のアクセスコントロールポリシーが置き換えられます。

## レスポンス

- **ETag**: エンティティタグは、**GetObjectExtended** を使用してオブジェクトの条件付きフェッチを実行するために使用できる、オブジェクトの MD5 ハッシュです。ETag は、オブジェクトのコンテンツに加えた変更のみを反映し、メタデータに加えた変更は反映しません。
- **LastModified**: 保存されたオブジェクトの Amazon S3 タイムスタンプ。

## アクセスコントロール

オブジェクトをバケットに配置するには、バケットへの **WRITE** アクセス権が必要です。

## 関連リソース

- [CopyObject \(SOAP API\)](#)

## CopyObject (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

## 説明

ソースオブジェクトのキーとバケット、およびターゲット送信先のキーとバケットを指定すると、**CopyObject** オペレーションによってオブジェクトのコピーが作成されます。

オブジェクトをコピーするときは、すべてのメタデータを保持する (デフォルト) か、新しいメタデータを指定できます。ただし、ACL は保持されず、リクエストを行ったユーザーに対して **private** に設定されます。デフォルトの ACL 設定を上書きするには、コピーリクエストを生成するときに新しい ACL を指定します。詳細については、「[ACL によるアクセス管理](#)」を参照してください。

すべてのコピーリクエストには認証が必要です。さらに、ソースオブジェクトへの read アクセスと、送信先バケットへの write アクセスが必要です。詳細については、「[認証アクセスの使用](#)」を参照してください。

Etag が一致するかどうか、オブジェクトが指定された日付の前後に変更されたかどうかなど、特定の条件下でのみオブジェクトをコピーするには、リクエストパラメータ CopySourceIfUnmodifiedSince、CopyIfUnmodifiedSince、CopySourceIfMatch、または CopySourceIfNoneMatch を使用します。

#### Note

大きなオブジェクトをコピーするには、SOAP スタックソケットのタイムアウトを設定する必要がある場合があります。

## リクエストの構文

```
<CopyObject xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <SourceBucket>source_bucket</SourceBucket>
  <SourceObject>source_object</SourceObject>
  <DestinationBucket>destination_bucket</DestinationBucket>
  <DestinationObject>destination_object</DestinationObject>
  <MetadataDirective>{REPLACE | COPY}</MetadataDirective>
  <Metadata>
    <Name>metadata_name</Name>
    <Value>metadata_value</Value>
  </Metadata>
  ...
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="user_type">
        <ID>user_id</ID>
        <DisplayName>display_name</DisplayName>
      </Grantee>
      <Permission>permission</Permission>
    </Grant>
    ...
  </AccessControlList>
  <CopySourceIfMatch>etag</CopySourceIfMatch>
  <CopySourceIfNoneMatch>etag</CopySourceIfNoneMatch>
  <CopySourceIfModifiedSince>date_time</CopySourceIfModifiedSince>
```

```
<CopySourceIfUnmodifiedSince>date_time</CopySourceIfUnmodifiedSince>
<AWSAccessKeyId>AWSAccessKeyId</AWSAccessKeyId>
<Timestamp>TimeStamp</Timestamp>
<Signature>Signature</Signature>
</CopyObject>
```

リクエストパラメータ

| 名前                | 説明                                                                                        | 必須 |
|-------------------|-------------------------------------------------------------------------------------------|----|
| SourceBucket      | ソースバケットの名前。<br><br>タイプ: 文字列<br><br>デフォルト: なし<br><br>制約: 有効なソースバケット。                       | はい |
| SourceKey         | ソースオブジェクトのキーの名前。<br><br>タイプ: 文字列<br><br>デフォルト: なし<br><br>制約: READ アクセスを持つ有効なソースオブジェクトのキー。 | はい |
| DestinationBucket | 宛先 バケットの名前。<br><br>タイプ: 文字列<br><br>デフォルト: なし<br><br>制約: 送信先バケットへの WRITE アクセスが必要です。        | はい |
| DestinationKey    | 送信先オブジェクトのキー。<br><br>タイプ: 文字列<br><br>デフォルト: なし                                            | はい |

| 名前                | 説明                                                                                                                                                                                                                                                       | 必須  |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|                   | 制約: 送信先バケットへの WRITE アクセスが必要です。                                                                                                                                                                                                                           |     |
| MetadataDirective | <p>メタデータをソースオブジェクトからコピーするか、リクエストで指定されたメタデータに置き換えるかを指定します。</p> <p>タイプ: 文字列</p> <p>デフォルト: COPY</p> <p>有効な値: COPY   REPLACE</p> <p>制約: COPY または REPLACE 以外の値は即時エラーになります。MetadataDirective ヘッダーが指定され、その値が REPLACE に設定されていない限り、オブジェクトをそれ自体にコピーすることはできません。</p> | いいえ |
| Metadata          | <p>オブジェクトに設定するメタデータの名前と値のペアを指定します。MetadataDirective が COPY に設定されている場合、すべてのメタデータは無視されます。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> <p>制約: なし。</p>                                                                                                              | いいえ |

| 名前                    | 説明                                                                                                                                                                    | 必須  |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| AccessControlList     | <p>E メールアドレスまたは正規ユーザー ID でユーザーにアクセス権を付与します。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> <p>制約: なし。</p>                                                                       | いいえ |
| CopySourceIfMatch     | <p>エンティティタグ (ETag) が指定されたタグと一致する場合はオブジェクトをコピーします。一致しない場合は、PreconditionFailed を返します。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> <p>制約: なし。Etag が一致しない場合、オブジェクトはコピーされません。</p> | いいえ |
| CopySourceIfNoneMatch | <p>エンティティタグ (ETag) が指定された Etag と異なる場合はオブジェクトをコピーします。それ以外の場合はエラーを返します。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> <p>制約: なし。</p>                                             | いいえ |

| 名前                          | 説明                                                                                                         | 必須  |
|-----------------------------|------------------------------------------------------------------------------------------------------------|-----|
| CopySourceIfUnmodifiedSince | 指定された時刻以降に変更されていない場合は、オブジェクトをコピーします。それ以外の場合は、PreconditionFailed を返します。<br><br>型: dateTime<br><br>デフォルト: なし | いいえ |
| CopySourceIfModifiedSince   | 指定された時刻以降に変更された場合は、オブジェクトをコピーします。それ以外の場合は、エラーを返します。<br><br>型: dateTime<br><br>デフォルト: なし                    | いいえ |

## レスポンスの構文

```
<CopyObjectResponse xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">  
  <CopyObjectResponse>  
    <ETag>"etag"</ETag>  
    <LastModified>timestamp</LastModified>  
  </CopyObjectResponse>  
</CopyObjectResponse>
```

## レスポンス要素

以下は、レスポンス要素のリストです。

### Note

SOAP API は余分な空白を返しません。追加の空白は REST API によってのみ返されます。

| 名前           | 説明                                                                                                                           |
|--------------|------------------------------------------------------------------------------------------------------------------------------|
| Etag         | <p>新しいオブジェクトの ETag を返します。ETag は、オブジェクトのコンテンツに加えた変更のみを反映し、メタデータに加えた変更は反映しません。</p> <p>タイプ: 文字列</p> <p>祖先: CopyObjectResult</p> |
| LastModified | <p>オブジェクトの最終変更日時を返します。</p> <p>タイプ: 文字列</p> <p>祖先: CopyObjectResult</p>                                                       |

一般的なレスポンス要素については、「[REST エラーレスポンスヘッダーの使用](#)」を参照してください。

### 特殊なエラー

このオペレーションには特別なエラーはありません。一般的な Amazon S3 のエラーについては、「[エラーコードのリスト](#)」を参照してください。

### 例

このサンプルは、flotsam オブジェクト (pacific バケット) を jetsam オブジェクト (atlantic バケット) にコピーし、そのメタデータを維持します。

### リクエスト例

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>pacific</SourceBucket>
  <SourceObject>flotsam</SourceObject>
  <DestinationBucket>atlantic</DestinationBucket>
  <DestinationObject>jetsam</DestinationObject>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```



## レスポンス例

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

この例では、ワンダーランドバケットから「tweedledee」オブジェクトをワンダーランドバケットの「tweedledum」オブジェクトにコピーし、メタデータを置き換えます。

## リクエスト例

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>wonderland</SourceBucket>
  <SourceObject>tweedledee</SourceObject>
  <DestinationBucket>wonderland</DestinationBucket>
  <DestinationObject>tweedledum</DestinationObject>
  <MetadataDirective>REPLACE</MetadataDirective>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>relationship</Name>
    <Value>twins</Value>
  </Metadata>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```

## レスポンス例

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

## 関連リソース

- [PutObject \(SOAP API\)](#)
- [PutObjectInline \(SOAP API\)](#)

## GetObject (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

GetObject オペレーションは、オブジェクトの現在のバージョンを返します。削除マーカが設定されているオブジェクトを現在のバージョンとして GetObject を実行しようとする、S3 は 404 エラーを返します。SOAP API を使用して、指定されたバージョンのオブジェクトを取得することはできません。そのためには、REST API を使用します。詳細については、「[バージョンニング](#)」を参照してください。その他のオプションについては、[GetObjectExtended \(SOAP API\)](#) オペレーションを使用します。

### Note

値「soap」を持つオブジェクトキー名は、[仮想ホスト形式のリクエスト](#)についてサポートされていません。「soap」が使用されるオブジェクトキー名値の場合、[パス形式の URL](#) を代わりに使用する必要があります。

## Example

この例では、「quotes」バケットから「Nelson」オブジェクトを取得します。

## Sample Request

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
```

```
<InlineData>true</InlineData>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

## Sample Response

```
<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>aGEtaGE=</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

## 要素

- Bucket: オブジェクトを取得するバケット。
- Key: オブジェクトを識別するキー。
- GetMetadata: これが true の場合、メタデータはオブジェクトとともに返されます。
- GetData: これが true の場合、オブジェクトデータが返されます。
- InlineData: これが true の場合、レスポンスの SOAP 本文の一部として、Base64 でエンコードされたデータが返されます。false の場合、データは SOAP アタッチメントとして返されます。InlineData オプションは、大きなオブジェクトでの使用には適していません。このオペレーションは、1 MB 以下のデータを操作する場合に限定されています。InlineData フラグが設定された GetObject リクエストは、結果のデータパラメータが 1MB を超えてエンコードされた場合、InlineDataTooLargeError ステータスコードで失敗します。大きなオブジェクトをダウ

ンロードするには、InlineData フラグを設定せずに GetObject を呼び出すか、代わりに REST API を使用することを検討してください。

## 返される要素

- Metadata: オブジェクトに保存されている名前と値のペアのメタデータ。
- Data: リクエストで InlineData が true の場合、これには Base64 エンコードされたオブジェクトデータが含まれます。
- LastModified: オブジェクトが Amazon S3 に保存された時刻。
- ETag: オブジェクトのエンティティタグ。これは、条件付きの GET を行うために使用できるオブジェクトのハッシュ値です。ETag は、オブジェクトのコンテンツに加えた変更のみを反映し、メタデータに加えた変更は反映しません。

## アクセスコントロール

オブジェクトへの READ アクセス権が付与されている場合にのみ、オブジェクトを読み取ることができます。

## SOAP チャンクダウンロードと再開可能なダウンロード

GET の柔軟性を提供するために、Amazon S3 はチャンクダウンロードと再開可能なダウンロードをサポートしています。

以下から選択します。

- 大きなオブジェクトをダウンロードする場合は、それらを小さなチャンクに分割できます。詳細については、[レンジ GET](#)を参照してください。
- GET オペレーションが失敗した場合、ファイル全体ではなく残りの部分をダウンロードするようにアプリケーションを設計できます。詳細については、[REST GET エラー回復](#)を参照してください。

## レンジ GET

一部のクライアントでは、大きなダウンロードを小さなダウンロードに分割する場合があります。GET を小さな単位に分割するには、Range を使用します。

GET を小さな単位に分割する前に、そのサイズを決定する必要があります。例えば、次のリクエストは bigfile オブジェクトのサイズを取得します。

```
<ListBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Prefix>bigfile</Prefix>
  <MaxKeys>1</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Amazon S3 は以下のレスポンスを返します。

```
<ListBucketResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Name>quotes</Name>
  <Prefix>N</Prefix>
  <MaxKeys>1</MaxKeys>
  <IsTruncated>>false</IsTruncated>
  <Contents>
    <Key>bigfile</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <Size>2023276</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>bcaf1ffd86f41161ca5fb16fd081034f</ID>
      <DisplayName>bigfile</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

以下は、bigfile オブジェクトから最初の 1 メガバイトをダウンロードするリクエストです。

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>0</ByteRangeStart>
  <ByteRangeEnd>1048576</ByteRangeEnd>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
```

```
</GetObject>
```

Amazon S3 は、ファイルの最初の 1 メガバイトとファイルの Etag を返します。

```
<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>--first megabyte of bigfile--</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

前の部分がダウンロードされてからファイルが変更されないようにするには、IfMatch 要素を指定します。IfMatch 要素は必須ではありませんが、変更される可能性が高いコンテンツには推奨されます。

以下は、IfMatch リクエストヘッダーを使用してファイルの残りの部分を取得するリクエストです。

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>10485761</ByteRangeStart>
  <ByteRangeEnd>2023276</ByteRangeEnd>
  <IfMatch>"828ef3fdfa96f00ad9f27c383fc9ac7f"</IfMatch>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
```

```
</GetObject>
```

Amazon S3 は、次のレスポンスとファイルの残りの部分を返します。

```
<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>>Muntz</Value>
    </Metadata>
    <Data>--remainder of bigfile--</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

## バージョンングされた GetObject

次のリクエストは、バケット内のオブジェクトの指定されたバージョンを返します。

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

## レスポンス例

```
<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
```

```
<GetObjectResponse>
<Status>
<Code>200</Code>
<Description>OK</Description>
</Status>
<Metadata>
<Name>Content-Type</Name>
<Value>text/plain</Value>
</Metadata>
<Metadata>
<Name>family</Name>
<Value>Muntz</Value>
</Metadata>
<Data>aGEtaGE=</Data>
<LastModified>2006-01-01T12:00:00.000Z</LastModified>
<ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
</GetObjectResponse>
</GetObjectResponse>
```

## REST GET エラー回復

オブジェクトの GET が失敗した場合、ダウンロードする範囲を指定してファイルの残りの部分を取得できます。そのためには、ListBucket を使用してオブジェクトのサイズを取得し、ファイルの残りの部分でレンジ GET を実行する必要があります。詳細については、「[GetObjectExtended \(SOAP API\)](#)」を参照してください。

## 関連リソース

### [オブジェクトに対するオペレーション \(SOAP API\)](#)

### GetObjectExtended (SOAP API)

#### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

GetObjectExtended は [GetObject \(SOAP API\)](#) とまったく同じですが、HTTP GET ヘッダーが提供するのと同じ機能の多くを実現するために使用できる以下の追加要素がサポートされています (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html> を参照)。



GetObjectExtended は、GetObject でサポートされている要素に加えて、次の要素をサポートしています。

- ByteRangeStart, ByteRangeEnd: これらの要素は、オブジェクトデータの一部のみを取得するように指定します。これらは HTTP バイト範囲の動作に従います (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35> を参照)。
- IfModifiedSince: オブジェクトのタイムスタンプが指定されたタイムスタンプより後である場合にのみ、オブジェクトを返します (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.25>)。
- IfUnmodifiedSince: オブジェクトのタイムスタンプが指定されたタイムスタンプより前か等しい場合にのみ、オブジェクトを返します (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.28> を参照)。
- IfMatch: ETag が指定されたタグ (複数可) と一致する場合にのみ、オブジェクトを返します (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24> を参照)。
- IfNoneMatch: ETag が指定されたタグ (複数可) と一致しない場合にのみ、オブジェクトを返します (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26> を参照)。
- ReturnCompleteObjectOnConditionFailure: ReturnCompleteObjectOnConditionFailure: true の場合、リクエストに範囲要素と IfUnmodifiedSince/IfMatch 要素のいずれかまたは両方が含まれ、条件が失敗した場合、フォルトではなくオブジェクト全体を返します。これにより、If-Range 機能が有効になります (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27> を参照)。

## DeleteObject (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

DeleteObject オペレーションは、Amazon S3 から指定されたオブジェクトを削除します。削除されると、オブジェクトを復元または削除解除する方法はありません。

 Note

存在しないオブジェクトを削除すると、Amazon S3 は成功を返します (エラーメッセージは返しません)。

## Example

この例では、「quotes」バケットから「Nelson」オブジェクトを削除します。

## Sample Request

```
<DeleteObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteObject>
```

## Sample Response

```
<DeleteObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <DeleteObjectResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </DeleteObjectResponse>
</DeleteObjectResponse>
```

## 要素

- Bucket: オブジェクトを格納するバケット。
- Key: オブジェクトを識別するキー。

## アクセスコントロール

オブジェクトは、オブジェクトの所有者や付与される権限に関係なく、バケットへの WRITE アクセスを持っている場合にのみ削除できます。

## GetObjectAccessControlPolicy (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

GetObjectAccessControlPolicy オペレーションは、オブジェクトのアクセスコントロールポリシーを取得します。

### Example

この例では、「quotes」バケットから「Nelson」オブジェクトのアクセスコントロールポリシーを取得します。

### Sample Request

```
<GetObjectAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObjectAccessControlPolicy>
```

### Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd24a541bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b841bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
```

```
<Grant>
  <Grantee xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
</AccessControlList>
</AccessControlPolicy>
```

## レスポンス本文

レスポンスには、バケットのアクセスコントロールポリシーが含まれます。このレスポンスの説明については、「[SOAP アクセスポリシー](#)」を参照してください。

## アクセスコントロール

オブジェクトのアクセスコントロールポリシーを取得するには、オブジェクトに対する READ\_ACP 権限が必要です。

## SetObjectAccessControlPolicy (SOAP API)

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

SetObjectAccessControlPolicy オペレーションは、既存のオブジェクトのアクセスコントロールポリシーを設定します。成功すると、オブジェクトの以前のアクセスコントロールポリシーは、指定されたアクセスコントロールポリシーに完全に置き換えられます。

## Example

この例では、指定されたユーザー (通常は所有者) に「quotes」バケットから「Nelson」オブジェクトへの FULL\_CONTROL アクセスを許可します。

## Sample Request

```
<SetObjectAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
```

```
<AccessControlList>
  <Grant>
    <Grantee xsi:type="CanonicalUser">
      <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
      <DisplayName>chriscustomer</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
</AccessControlList>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</SetObjectAccessControlPolicy>
```

## Sample Response

```
<SetObjectAccessControlPolicyResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <SetObjectAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </SetObjectAccessControlPolicyResponse>
</SetObjectAccessControlPolicyResponse>
```

## キー

### アクセスコントロール

バケットのアクセスコントロールポリシーを設定するには、オブジェクトに対する WRITE\_ACP 権限が必要です。

## SOAP リクエストの認証方法

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

リクエスト元となるプリンシパルのアイデンティティを確立するには、すべての非匿名リクエストは認証情報を含んでいる必要があります。SOAP では、認証情報は SOAP リクエストの以下の要素に置かれています。

- お客様の AWS アクセスキー ID。

 Note

認証済みの SOAP リクエストを行う場合、一時セキュリティ証明書はサポートされません。証明書の種類の詳細については、[リクエストの実行](#) を参照してください。

- Timestamp: これは、2009-01-01T12:00:00.000Z など、協定世界時 (グリニッジ標準時) タイムゾーンの dateTime (<http://www.w3.org/TR/xmlschema-2/#dateTime> を参照) である必要があります。このタイムスタンプが Amazon S3 サーバーの時刻から 15 分を過ぎている場合、認可は失敗します。
- Signature: AWS のシークレットアクセスキーをキーとして使用し、AmazonS3 + オペレーション + タイムスタンプの連結文字で生成される RFC 2104 の HMAC-SHA1 ダイジェスト (<http://www.ietf.org/rfc/rfc2104.txt> を参照)。たとえば、以下の CreateBucket サンプルリクエストでは、署名要素に「AmazonS3CreateBucket2009-01-01T12:00:00.000Z」という値の HMAC-SHA1 ダイジェストが含まれています。

たとえば、以下の CreateBucket サンプルリクエストでは、署名要素に「AmazonS3CreateBucket2009-01-01T12:00:00.000Z」という値の HMAC-SHA1 ダイジェストが含まれています。

### Example

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Acl>private</Acl>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-01-01T12:00:00.000Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

 Note

SOAP リクエストは、認証済みの場合も匿名の場合も、SSL を使用して Amazon S3 に送信する必要があります。HTTP を介して SOAP リクエストが送信された場合、Amazon S3 はエラーを返します。

**⚠ Important**

時間の精度の桁数をどこで切り捨てるかにはさまざまな解釈があるため、.NET ユーザーは過度に詳細なタイムスタンプを Amazon S3 に送信しないようにする必要があります。ミリ秒での精度で DateTime オブジェクトを手動で構築することによって、これを達成することができます。

## SOAP のアクセスポリシーの設定

**i Note**

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

アクセスコントロールは、バケットまたはオブジェクトの記述時に、「AccessControlList」要素と一緒に CreateBucket、PutObjectInline、または PutObject へのリクエストを含めることによって設定できます。AccessControlList 要素は、「[Amazon S3 用 Identity and Access Management](#)」で説明されています。これらのオペレーションでアクセスコントロールリストが指定されていない場合、リクエストに FULL\_CONTROL アクセスを付与するデフォルトのアクセスポリシーでリソースを生成します (これは該当のリクエストが、既に存在するオブジェクトへの PutObjectInline または PutObject リクエストである場合にも当てはまります)。

以下に示すリクエストは、オブジェクトにデータを書き込み、匿名のプリンシパルによるオブジェクトの読み取りを可能にし、特定のユーザーにバケットへの FULL\_CONTROL の権限を付与します (ほとんどの開発者は、自分で自分のバケットに FULL\_CONTROL アクセスを付与したいと考えるでしょう)。

### Example

以下に示すリクエストは、オブジェクトにデータを書き込み、匿名のプリンシパルによるオブジェクトの読み取りを可能にします。

### Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
```

```

<Key>Nelson</Key>
<Metadata>
  <Name>Content-Type</Name>
  <Value>text/plain</Value>
</Metadata>
<Data>aGEtaGE=</Data>
<ContentLength>5</ContentLength>
<AccessControlList>
  <Grant>
    <Grantee xsi:type="CanonicalUser">
      <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
      <DisplayName>chriscustomer</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
    <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
    </Grantee>
    <Permission>READ</Permission>
  </Grant>
</AccessControlList>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2009-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>

```

## Sample Response

```

<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>

```

既存のバケットまたはオブジェクトのアクセスコントロールポリシー

は、GetBucketAccessControlPolicy、GetObjectAccessControlPolicy、SetBucketAccessControlPolicy メソッドを使って読み込みまたは設定できます。詳細については、これらのメソッドの詳しい説明を参照してください。



## 一般的な要素

いずれの SOAP リクエストにも、認可に関連する次の要素を含めることができます。

- `AWSSecretAccessKey`: リクエストの AWS アクセスキー ID
- `Timestamp`: システムの現在の時刻
- `Signature`: リクエストの署名

## SOAP エラーレスポンス

### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

SOAP の場合、エラー結果は SOAP フォルトとして HTTP レスポンスコード 500 と共にクライアントに返されます。SOAP フォルトを受け取っていない場合、リクエストは成功しています。Amazon S3 SOAP フォルトコードは、標準の SOAP 1.1 フォルトコード (「Server」または「Client」) に Amazon S3 固有のエラーコードが連結した構成になっています。たとえば、「Server.InternalError」や「Client.NoSuchBucket」などです。SOAP フォルト文字列要素には、人が読むことができる英語の汎用エラーメッセージが含まれます。また、SOAP フォルト詳細要素には、エラーに関連するさまざまな情報が含まれます。

たとえば、「Fred」というオブジェクトを削除しようとして、このオブジェクトが存在しなかった場合、SOAP レスポンスの本文には「NoSuchKey」という SOAP エラーが含まれます。

以下の例は、SOAP エラーレスポンスのサンプルです。

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
    <Faultstring>The specified key does not exist.</Faultstring>
    <Detail>
      <Key>Fred</Key>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

次の表に、SOAP エラーレスポンスの要素を示します。

| 名前          | 説明                                                                                                                                                                                                                         |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Detail      | エラーに関するキーのコンテナ<br><br>型: コンテナ<br><br>祖先: Body.Fault                                                                                                                                                                        |
| Fault       | エラー情報のコンテナ。<br><br>型: コンテナ<br><br>祖先: Body                                                                                                                                                                                 |
| Faultcode   | フォルトコードは、エラー状態を個別に識別する文字列です。エラーを検出してタイプ別に処理するプログラムによって読み取りおよび解釈されるためのものです。詳細については、「 <a href="#">エラーコードのリスト</a> 」を参照してください。<br><br>タイプ: 文字列<br><br>祖先: Body.Fault                                                           |
| Faultstring | フォルト文字列には、エラー状態の一般的な説明が英語で含まれます。これは人が理解できるようにするためのものです。シンプルなプログラムの場合、処理できない、または処理されないエラー状態が発生すると、メッセージがエンドユーザーに直接表示されます。より徹底したエラー処理を備え、適切に国際化されている洗練されたプログラムでは、フォルト文字列が無視される傾向にあります。<br><br>タイプ: 文字列<br><br>祖先: Body.Fault |
| Key         | エラーに関連するキーを識別します。<br><br>タイプ: 文字列<br><br>祖先: Body.Fault                                                                                                                                                                    |

## 付録: リクエストの認証 (AWS 署名バージョン 2)

### Important

このセクションでは AWS 署名バージョン 2 を使用してリクエストを認証する方法を説明します。署名バージョン 2 は無効 (非推奨) にされているため、Amazon S3 は署名バージョン 4 を使用して署名された API リクエストのみを受け入れます。詳細については、「[Amazon S3 用の AWS 署名バージョン 2 がオフになっている \(廃止\)](#)」を参照してください。

すべての AWS リージョン は署名バージョン 4 をサポートし、新しいリージョンでサポートされるのはこのバージョンのみです。詳細については、Amazon Simple Storage Service API リファレンスの「[リクエストの認証 \(AWS 署名バージョン 4\)](#)」を参照してください。

Amazon S3 では、リクエストの署名にどの API 署名バージョンが使用されているかを識別できます。ビジネスへの影響を防ぐために、いずれかのワークフローが署名バージョン 2 の署名を利用しているかどうかを識別し、それらを署名バージョン 4 を使用するようにアップグレードすることが重要です。

- CloudTrail のイベントログ (推奨オプション) を使用している場合、そのようなリクエストをクエリおよび識別する方法については、「[CloudTrail を使用した Amazon S3 Signature Version 2 リクエストの識別](#)」を参照してください。
- Amazon S3 サーバーアクセスログを使用している場合は、「[Amazon S3 サーバーアクセスログを使用したリクエストの識別](#)」を参照してください。

### トピック

- [REST API を使用したリクエストの認証 \(AWS 署名バージョン 2\)](#)
- [REST リクエストの署名と認証 \(AWS 署名バージョン 2\)](#)
- [POST \(AWS 署名バージョン 2\) を使用したブラウザベースのアップロード](#)

## REST API を使用したリクエストの認証(AWS 署名バージョン 2)

REST を使用して Amazon S3 にアクセスする場合は、リクエストが認証されるように、リクエストに以下の項目を指定する必要があります。

### リクエストの要素

- AWS アクセスキー ID – 各リクエストには、リクエストの送信に使用するアイデンティティのアクセスキー ID を含める必要があります。
- 署名 – 各リクエストには、リクエストの有効な署名を含める必要があります。含まれていない場合、リクエストは拒否されます。

リクエスト署名は、シークレットアクセスキーを使用して計算されます。シークレットアクセスキーは、自分と AWS のみが知っている共有の秘密です。

- タイムスタンプ – 各リクエストには、リクエストを作成した日時 (UTC) を文字列として含める必要があります。
- 日付 – 各リクエストには、リクエストのタイムスタンプを含める必要があります。

使用する API アクションによっては、タイムスタンプを追加するかわりに、リクエストの有効期限を示す日時を指定できます。特定のアクションに必要な項目を確認するには、そのアクションの認証のトピックを参照してください。

Amazon S3 へのリクエストを認証する一般的なステップを以下に示します。この説明では、必要なセキュリティ認証情報、アクセスキー ID、およびシークレットアクセスキーをお客様が持っていることを前提としています。

**You****1 Create a request:****Request**

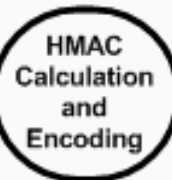
AccessKeyId = ...  
Action = ...  
Timestamp = ...  
ParameterA = ...

**2 Create an HMAC-SHA1 signature:**

String based on  
request contents

+

Your Secret Access Key



Your Signature

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

**3 Send the request and signature to AWS:****Request**

AccessKeyId = ...  
Action = ...  
Timestamp = ...  
ParameterA = ...

Your Signature

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

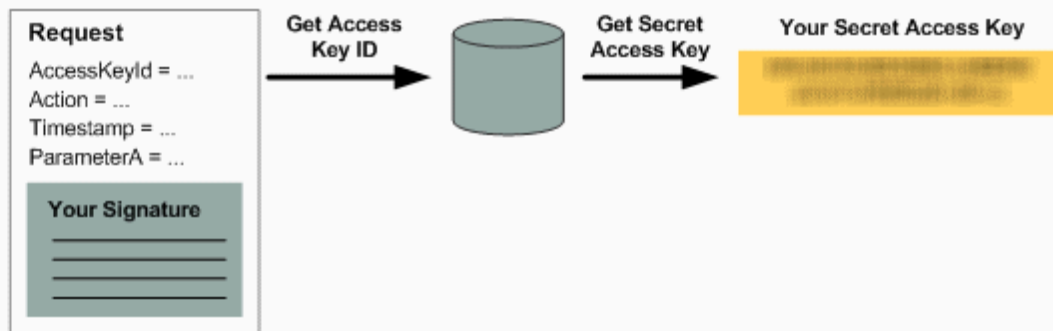


**AWS**

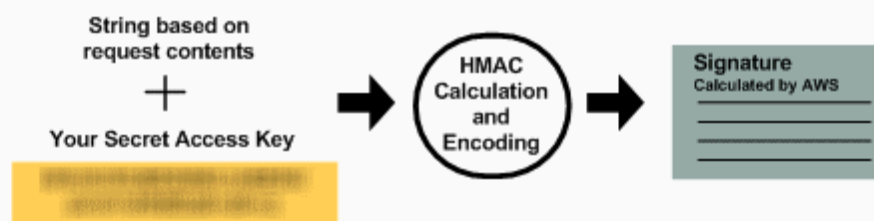
- 1 AWSにリクエストします。
- 2 シークレットアクセスキーを使用して署名を計算します。
- 3 Amazon S3 にリクエストを送信します。リクエストにアクセスキー ID と署名を含めます。次の 3 つのステップは、Amazon S3 で実行されます。

## AWS

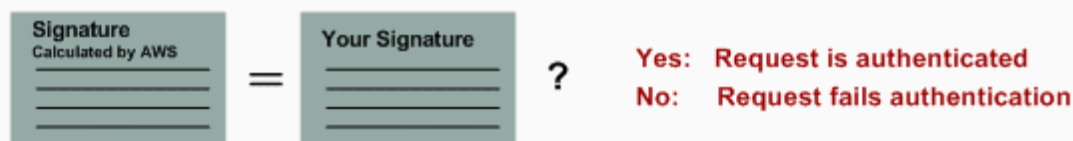
## 4 Retrieve your Secret Access Key:



## 5 Create an HMAC-SHA1 signature:



## 6 Compare the two signatures:



4 Amazon S3 がアクセスキー ID を使用してシークレットアクセスキーを調べます。

5 リクエストで送信した署名を計算するために使用したものと同一アルゴリズムを使用して、Amazon S3 がリクエストのデータとシークレットアクセスキーから署名を計算します。

6 Amazon S3 が生成した署名がリクエストで送信したものと一致した場合、リクエストは本物とみなされます。署名が一致しなかった場合、リクエストは破棄され、Amazon S3 はエラーレスポンスを返します。

## 詳細な認証情報

REST 認証の詳細については、「[REST リクエストの署名と認証\(AWS 署名バージョン 2\)](#)」を参照してください。

## REST リクエストの署名と認証(AWS 署名バージョン 2)

### トピック

- [一時的なセキュリティ認証情報の使用](#)
- [認証ヘッダー](#)
- [署名のためのリクエストの標準化](#)
- [CanonicalizedResource 要素の作成](#)
- [CanonicalizedAmzHeaders 要素の作成](#)
- [位置ヘッダーおよび指定 HTTP ヘッダーの StringToSign 要素](#)
- [タイムスタンプの要件](#)
- [認証の例](#)
- [REST リクエストの署名に関する問題](#)
- [クエリ文字列による代替リクエスト認証](#)

### Note

このトピックでは、署名バージョン 2 を使用してリクエストを認証する方法を説明します。Amazon S3 では、現在、最新の署名バージョン 4 がサポートされています。この最新の署名バージョンはすべてのリージョンでサポートされており、2014 年 1 月 30 日以降は、新しいリージョンではバージョン 4 の署名のみがサポートされます。詳細については、Amazon Simple Storage Service API リファレンスの「[リクエストの認証 \(AWS 署名バージョン 4\)](#)」を参照してください。

認証とは、自身のアイデンティティをシステムに証明するプロセスのことをいいます。アイデンティティは、Amazon S3 のアクセス管理の判断における重要な要因です。リクエストの許可または拒否は、リクエストのアイデンティティに部分的に基づいています。たとえば、バケットを作成する権利は、登録開発者用に予約されています。また、バケットにオブジェクトを作成する権利は、対象のバケット所有者用に (デフォルトで) 予約されています。開発者としてこれらの権限を実行するリクエ

ストを行うので、そのリクエストを認証することで、自身のアイデンティティをシステムに対して証明する必要があります。このセクションでは、その方法を説明します。

 Note

このセクションの内容は HTTP POST には適用されません。詳細については、「[POST \(AWS 署名バージョン 2\) を使用したブラウザベースのアップロード](#)」を参照してください。

Amazon S3 の REST API では、キーを使用して生成される HMAC (ハッシュメッセージ認証コード) に基づくカスタムの HTTP スキーマが認証に使用されます。リクエストを認証するには、まず、選択されているリクエストの要素を連結し、文字列を作成します。次に、AWS シークレットアクセスキーを使用して、その文字列の HMAC を計算します。正式な呼び名ではありませんが、このプロセスのことを「リクエストへの署名」と呼び、HMAC アルゴリズムの出力を署名と呼びます。これは、このプロセスが実際の署名のセキュリティプロパティを真似ているからです。最後に、このセクションで説明する構文を使用して、この署名をリクエストのパラメータとして追加します。

システムは、認証済みリクエストを受け取るとき、リクエスト送信者が所有している AWS シークレットアクセスキーを取得し、同様の方法でそのアクセスキーを使用して、受信したメッセージの署名を計算します。そして、計算した署名と、リクエストから提示された署名を比較します。2 つの署名が一致したら、リクエストは AWS シークレットアクセスキーにアクセスできると判断されるため、システムはそのキーの発行先となるプリンシパルの権限をサポートします。2 つの署名が一致しない場合、リクエストは中断し、エラーメッセージが返されます。

Example 認証された Amazon S3 の REST リクエスト

```
GET /photos/puppy.jpg HTTP/1.1
Host: awsexamplebucket1.us-west-1.s3.amazonaws.com
Date: Tue, 27 Mar 2007 19:36:42 +0000
```

```
Authorization: AWS AKIAIOSFODNN7EXAMPLE:
qgk2+6Sv9/oM7G3qLEjTH1a1l1g=
```

## 一時的なセキュリティ認証情報の使用

一時的なセキュリティ認証情報を使用してリクエストに署名する場合は (「[リクエストの実行](#)」を参照)、x-amz-security-token ヘッダーを追加して、対応するセキュリティトークンをリクエストに含める必要があります。



AWS Security Token Service API を使用して一時的なセキュリティ認証情報を取得すると、そのレスポンスには、一時的なセキュリティ認証情報とセッショントークンが含まれます。リクエストを Amazon S3 に送信するときに、x-amz-security-token ヘッダーでセッショントークン値を指定します。IAM が提供する AWS Security Token Service API については、AWS Security Token Service API リファレンスガイドの「[アクション](#)」を参照してください。

## 認証ヘッダー

Amazon S3 の REST API は、標準の HTTP ヘッダーの Authorization を使用して認証情報を渡します。(標準ヘッダーの名前とは異なり、ヘッダーに含まれるのは認可ではなく認証情報です。) Amazon S3 の認証スキームにおける認可ヘッダーの形式は次のとおりです。

```
Authorization: AWS AWSAccessKeyId:Signature
```

デベロッパーには、登録時に AWS アクセスキー ID と AWS シークレットアクセスキーが発行されます。リクエストを認証するために、AWSAccessKeyId 要素は、署名の計算に使用されたアクセスキー ID を識別するほか、リクエストを行っている開発者も間接的に識別します。

Signature 要素は、リクエストから選択した要素の RFC 2104HMAC-SHA1 です。したがって、Authorization ヘッダーの Signature 部分はリクエストによって異なります。システムによって計算されたリクエストの署名が、リクエストに含まれる Signature と一致する場合は、リクエストが AWS シークレットアクセスキーを所有していることになります。その後、リクエストは、キーの発行対象者である開発者のアイデンティティと権限に従って処理されます。

以下は擬似文法で、Authorization リクエストヘッダーの構文例を示しています。(例中の \n は Unicode のコードポイント U+000A を意味しています。これは通常改行と呼ばれています)。

```
Authorization = "AWS" + " " + AWSAccessKeyId + ":" + Signature;
```

```
Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of(YourSecretAccessKey), UTF-8-Encoding-Of( StringToSign ) ) );
```

```
StringToSign = HTTP-Verb + "\n" +  
Content-MD5 + "\n" +  
Content-Type + "\n" +  
Date + "\n" +  
CanonicalizedAmzHeaders +  
CanonicalizedResource;
```

```
CanonicalizedResource = [ "/" + Bucket ] +  
    <HTTP-Request-URI, from the protocol name up to the query string> +  
    [ subresource, if present. For example "?acl", "?location", or "?logging"];  
  
CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1 は、[RFC 2104 の Keyed-Hashing for Message Authentication](#) で定義されているアルゴリズムです。このアルゴリズムは、キーとメッセージの 2 つのバイト文字列を入力として取ります。Amazon S3 のリクエストの認証では、AWS のシークレットアクセスキー (YourSecretAccessKey) をキーとして使用し、StringToSign の UTF-8 エンコーディングをメッセージとして使用します。また、HMAC-SHA1 の出力もバイト文字列で、これはダイジェストと呼ばれます。Signature リクエストパラメータは、このダイジェストをエンコードする Base64 によって作成されます。

## 署名のためのリクエストの標準化

既に説明したように、システムは認証済みリクエストを受け取るときに、計算されたリクエスト署名と StringToSign のリクエストで指定された署名を比較します。そのため、署名は Amazon S3 で使用されているものと同じ方法で計算する必要があります。署名のためにリクエストを承認済み形式にするプロセスは正規化と言います。

## CanonicalizedResource 要素の作成

CanonicalizedResource は、リクエストの対象となる Amazon S3 のリソースを表します。REST リクエストのこのリソースは次のように作成します。

1. 空の文字列 ("" ) で開始します。
2. HTTP ホストヘッダー (仮想ホスト形式) を使用してバケットが指定されているリクエストについては、バケット名の前に "/" を付けます (例: 「/bucketname」)。パス形式のリクエスト、およびバケットを処理しないリクエストの場合は、何も行いません。仮想ホスト形式のリクエストの詳細については、「[バケットの仮想ホスティング](#)」を参照してください。

仮想ホスティング形式のリクエスト「https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg」の場合、CanonicalizedResource は「/awsexamplebucket1」です。

パス形式のリクエスト「https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg」の場合、CanonicalizedResource は "" です。

3. デコードされていない HTTP リクエスト URI のパス部分を追加します (クエリ文字列まで、ただし、その文字列は含みません)。

仮想ホスト形式のリクエスト「`https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg`」の場合、CanonicalizedResource は「`/awsexamplebucket1/photos/puppy.jpg`」です。

パス形式のリクエスト「`https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg`」の場合、CanonicalizedResource は「`/awsexamplebucket1/photos/puppy.jpg`」です。この時点で、CanonicalizedResource は、仮想ホスト形式とパス形式の両方のリクエストで同じです。

[GET サービス](#)などのバケットを指定しないリクエストの場合は、「`/`」を追加します。

4. `?versioning`、`?location`、`?acl`、`?lifecycle`、`?versionid`などのサブリソースを処理するリクエストの場合は、サブリソース、そのサブリソースの値 (存在する場合)、および疑問符を追加します。複数のサブリソースは、名前のアルファベット順に並べ替えて `"&"` で区切る必要があります。たとえば、`?acl&versionId=value` のようにする必要があります。

CanonicalizedResource 要素を作成するときに含める必要があるサブリソース

は、`acl`、`lifecycle`、`location`、`logging`、`notification`、`partNumber`、`policy`、`requestPayment`、`uploadId`、`website` です。

リクエストでレスポンスヘッダーの値を上書きするクエリ文字列パラメータを指定する場合 (「[GetObject](#)」を参照) は、クエリ文字列パラメータとその値を追加します。これらのパラメータ値は署名時にエンコードする必要はありませんが、リクエストの実行時にはエンコードする必要があります。GET リクエストのクエリ文字列パラメータには、`response-content-type`、`response-content-language`、`response-expires`、`response-cache-control`、`response-content-disposition`、`response-content-encoding` があります。

複数のオブジェクトの Delete リクエストに対して CanonicalizedResource を作成する際には、`delete` クエリ文字列パラメータを含める必要があります。

HTTP リクエスト URI の CanonicalizedResource の要素は、URL エンコーディングメタ文字を含め、HTTP リクエストに表示されるとおりに署名する必要があります。

CanonicalizedResource は、HTTP リクエスト URI とは異なる場合があります。特に、リクエストが HTTP Host ヘッダーを使用してバケットを指定する場合、そのバケットは HTTP リクエスト URI に表示されませんが、CanonicalizedResource にもバケットが含まれます。クエリ文字

列パラメータは、リクエスト URI に表示される可能性があります、CanonicalizedResource には含まれません。詳細については、「[バケットの仮想ホスティング](#)」を参照してください。

## CanonicalizedAmzHeaders 要素の作成

StringToSign の CanonicalizedAmzHeaders 部分を作成するには、次の手順に従って、「x-amz-」で始まるすべての HTTP リクエストヘッダーを選択します (大文字と小文字は区別されません)。

1. 各 HTTP ヘッダー名を小文字に変換します。たとえば、「X-Amz-Date」は「x-amz-date」に変換します。
2. ヘッダーのコレクションを辞書と同じ順序でヘッダー名ごとに並べ替えます。
3. RFC 2616 のセクション 4.2 の説明に従って、同じ名前のヘッダーフィールドを 1 つの「ヘッダー名:カンマ区切りの値のリスト」にまとめます。値の間にスペースは入れません。たとえば、2 つのメタデータヘッダー「x-amz-meta-username: fred」および「x-amz-meta-username: barney」が 1 つのヘッダーに結合されると、「x-amz-meta-username: fred,barney」になります。
4. (RFC 2616 のセクション 4.2 で許可されている) 複数行にわたる長いヘッダーの折り返しのスペース (改行を含む) を 1 つのスペースに置き換えて、折り返しをなくします。
5. ヘッダーのコロンの前後のスペースをすべて削除します。例えば、ヘッダー「x-amz-meta-username: fred,barney」は「x-amz-meta-username:fred,barney」になります。
6. 最後に、改行文字 (U+000A) を、結果の一覧の各標準化ヘッダーに追加します。この一覧のすべてのヘッダーを 1 つの文字列に連結することで、CanonicalizedResource 要素を作成します。

## 位置ヘッダーおよび指定 HTTP ヘッダーの StringToSign 要素

StringToSign の最初のヘッダー要素 (Content-Type、Date、および Content-MD5) は位置を示します。StringToSign には、これらのヘッダーの名前は含まれません。含まれるのはリクエストの値のみです。一方、「x-amz-」要素には名前が付いています。ヘッダーの名前と値は両方とも StringToSign に含まれます。

StringToSign の定義で呼び出される位置ヘッダーがリクエストにない場合は (たとえば、Content-Type または Content-MD5 は、PUT リクエストのオプションであり、GET リクエストには無意味です)、空の文字列 ("" ) をその位置に代入します。

## タイムスタンプの要件

認証済みリクエストには有効なタイムスタンプ (HTTP Date ヘッダーまたは x-amz-date 代替) が必ず必要です。また、認証済みリクエストに含まれているクライアントのタイムスタンプは、リクエストの受信時の Amazon S3 システムの時刻から 15 分以内である必要があります。そうでない場合、リクエストは失敗し、RequestTimeTooSkewed エラーコードが返されます。このように制限することで、攻撃者によって傍受されたリクエストが繰り返される可能性を限定します。傍受に対する保護をさらに強化するには、認証済みリクエストに対して HTTPS 転送を使用します。

### Note

リクエスト日の検証の制約は、クエリ文字列認証を使用しない認証済みリクエストにのみ適用されます。詳細については、「[クエリ文字列による代替リクエスト認証](#)」を参照してください。

HTTP クライアントライブラリによっては、リクエストの Date ヘッダーを設定する機能が公開されていないことがあります。標準化ヘッダーの「Date」ヘッダーの値を含めることができない場合は、代わりに「x-amz-date」ヘッダーを使用してリクエストのタイムスタンプを設定します。x-amz-date ヘッダーの値は、RFC 2616 (<http://www.ietf.org/rfc/rfc2616.txt>) のいずれかの形式である必要があります。x-amz-date ヘッダーがリクエストに存在する場合は、すべての Date ヘッダーがリクエスト署名の計算時に無視されます。したがって、x-amz-date ヘッダーを含める場合、Date を作成するときは、空の文字列を StringToSign に対して使用します。例については、次のセクションを参照ください。

## 認証の例

このセクションの例では、次の表の証明書 (実際は機能していません) を使用しています。

| パラメータ              | 値                                        |
|--------------------|------------------------------------------|
| AWSAccessKeyId     | AKIAIOSFODNN7EXAMPLE                     |
| AWSSecretAccessKey | wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY |

StringToSign の例では、書式設定はそれほど複雑ではありません。\\n は Unicode コードポイント U+000A を意味します。これは一般的には改行と呼ばれています。また、例では「+0000」を使用

してタイムゾーンを指定しています。代わりに「GMT」を使用してタイムゾーンを指定することもできますが、署名はこの例に示したものと異なることになります。

## オブジェクト GET

この例では、awsexamplebucket1 バケットからオブジェクトを取得します。

| リクエスト                                                                                                                                                                                                     | StringToSign                                                                                 |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| <pre>GET /photos/puppy.jpg HTTP/1.1 Host: awsexamplebucket1.us-west-1.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:36:42 +0000  Authorization: AWS AKIAIOSFODNN7EXAMPLE: qgk2+6Sv9/oM7G3qLEjTH1a1l1g=</pre> | <pre>GET\n \n \n Tue, 27 Mar 2007 19:36:42 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre> |

バケット名は CanonicalizedResource には含まれますが、HTTP リクエスト URI には含まれないことに注意してください。(これはホストヘッダーによって指定されます。)

### Note

以下の Python スクリプトは、渡されたパラメータを使用して、先程の署名を計算します。このスクリプトを使用し、必要に応じてキーおよび StringToSign を置き換えて、独自の署名を作成できます。

```
import base64
import hmac
from hashlib import sha1

access_key = 'AKIAIOSFODNN7EXAMPLE'.encode("UTF-8")
secret_key = 'wJa1rXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY'.encode("UTF-8")

string_to_sign = 'GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n/awsexamplebucket1/
photos/puppy.jpg'.encode("UTF-8")
signature = base64.b64encode(
    hmac.new(
        secret_key, string_to_sign, sha1
```

```

        ).digest()
    ).strip()

print(f"AWS {access_key.decode()}:{signature.decode()}")

```

## Object PUT

この例では、オブジェクトを `awsexamplebucket1` バケットに配置します。

| リクエスト                                                                                                                                                                                                                                                      | StringToSign                                                                                             |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| <pre> PUT /photos/puppy.jpg HTTP/1.1 Content-Type: image/jpeg Content-Length: 94328 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:15:45 +0000  Authorization: AWS AKIAIOSFODNN7EXAMPLE: iqRzw+ileNPu1fhspnRs8n0jjIA= </pre> | <pre> PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /awsexamplebucket1/photos/puppy.jpg </pre> |

リクエスト内および StringToSign 内の Content-Type ヘッダーに注意してください。また、Content-MD5 はリクエストにないので、StringToSign では空白のままであることを注意してください。

## リスト

この例では、`awsexamplebucket1` バケットのコンテンツのリストを取得します。

| リクエスト                                                                                                                                                                                       | StringToSign                                               |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <pre> GET /?prefix=photos&amp;max-keys=50&amp;marker=puppy HTTP/1.1 User-Agent: Mozilla/5.0 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 19:42:41 +0000 </pre> | <pre> GET\n \n \n Tue, 27 Mar 2007 19:42:41 +0000\n </pre> |



| リクエスト                                                                            | StringToSign        |
|----------------------------------------------------------------------------------|---------------------|
| <i>Authorization: AWS AKIAIOSFODNN7EXAMPLE:<br/>m0WP8eCtspQl5Ahe6L1SozdX9YA=</i> | /awsexamplebucket1/ |

CanonicalizedResource の後ろにスラッシュがあること、そしてクエリ文字列パラメータがないことに注意してください。

## 取得

この例では、「awsexamplebucket1」バケットのアクセスコントロールポリシーのサブリソースを取得します。

| リクエスト                                                                                                                                                                                                    | StringToSign                                                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| GET /?acl HTTP/1.1<br>Host: awsexamplebucket1.s3.us-west-1.amazonaws.com<br>Date: Tue, 27 Mar 2007 19:44:46 +0000<br><br><i>Authorization: AWS AKIAIOSFODNN7EXAMPLE:<br/>82ZHiFIjc+WbcwFKGUVQspPn+0=</i> | GET\n\n\nTue, 27 Mar 2007 19:44:46<br>+0000\n/awsexamplebucket1/?acl |

サブリソースクエリ文字列パラメータがどのように CanonicalizedResource に挿入されているかに注意してください。

## 削除

この例では、パス形式および Date 代替を使用して、「awsexamplebucket1」バケットからオブジェクトを削除します。

| リクエスト                                                                                                         | StringToSign                                  |
|---------------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| DELETE /awsexamplebucket1/photos/puppy.jpg HTTP/1.1<br>User-Agent: dotnet<br>Host: s3.us-west-1.amazonaws.com | DELETE\n\n\nTue, 27 Mar 2007 21:20:26 +0000\n |



| リクエスト                                                                                                                                                                 | StringToSign                                   |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|
| <pre>Date: Tue, 27 Mar 2007 21:20:27 +0000  x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE LE:XbyTlbQdu9Xw5o8P4iMwPktxQd8=</pre> | <pre>/awsexamplebucket1/photos/puppy.jpg</pre> |

代替の「x-amz-date」方式を使用して日付を指定する方法に注意してください (クライアントライブラリにより、日付を設定できない場合など)。このような場合、x-amz-date が Date ヘッダーより優先されます。このため、署名内の日付エントリに x-amz-date ヘッダーの値を含める必要があります。

## アップロード

この例では、オブジェクトを、メタデータが含まれる CNAME スタイルの仮想ホストバケットにアップロードします。

| リクエスト                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | StringToSign                                                                                                                                                                                                                                                                                            |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>PUT /db-backup.dat.gz HTTP/1.1 User-Agent: curl/7.15.5 Host: static.example.com:8080 Date: Tue, 27 Mar 2007 21:06:08 +0000  x-amz-acl: public-read content-type: application/x-download Content-MD5: 4gJE4saaMU4BqNR0kLY+lw== X-Amz-Meta-ReviewedBy: joe@example.com X-Amz-Meta-ReviewedBy: jane@example.com X-Amz-Meta-FileChecksum: 0x02661779 X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment;   filename=database.dat Content-Encoding: gzip Content-Length: 5913339  Authorization: AWS AKIAIOSFODNN7EXAMPLE LE:</pre> | <pre>PUT\n 4gJE4saaMU4BqNR0kLY+lw==\n application/x-download\n Tue, 27 Mar 2007 21:06:08 +0000\n  x-amz-acl:public-read\n x-amz-meta-checksumalgorithm:crc32\n x-amz-meta-filechecksum:0x02661779\n x-amz-meta-reviewedby:joe@example.com,jane@example.com\n /static.example.com/db-backup.dat.gz</pre> |

| リクエスト                               | StringToSign |
|-------------------------------------|--------------|
| <i>jtBQa0Aq+DkULFI8qrpwIjGEx0E=</i> |              |

「x-amz-」ヘッダーが並べ替えられ、スペースが削除されて小文字に変換されていることに注意してください。また、同じ名前を持つ複数のヘッダーが値の区切りとしてカンマを使用して結合されている様子も確認してください。

Content-Type および Content-MD5 HTTP エンティティヘッダーのみが StringToSign に表示されていることに注意してください。もう一方の Content-\* エンティティヘッダーは表示されていません。

また、バケット名は CanonicalizedResource には含まれますが、HTTP リクエスト URI には含まれないことに注意してください。(これはホストヘッダーによって指定されます。)

自分のすべてのバケットのリストを取得する

| リクエスト                                                                                                                                                                   | StringToSign                                               |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <pre>GET / HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:29:59 +0000  Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdE RIC03wnaRNKh60qZehG9s=</pre> | <pre>GET\n \n \n Wed, 28 Mar 2007 01:29:59 +0000\n /</pre> |

Unicode キー

| リクエスト                                                                                                                                                                                                             | StringToSign                                                                                          |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <pre>GET /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:49:49 +0000 Authorization: AWS AKIAIOSFODNN7EXAMP LE:DNEZGsoieTZ92F3bUfSPQcbGm1M=</pre> | <pre>GET\n \n \n Wed, 28 Mar 2007 01:49:49 +0000\n /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re</pre> |

**Note**

リクエスト URI から派生した StringToSign の要素は、URL エンコーディングおよび大文字/小文字の設定を含め、文字通りに取得されます。

## REST リクエストの署名に関する問題

REST リクエスト認証が失敗すると、システムは、そのリクエストに対して XML エラードキュメントで応答します。このエラードキュメントに含まれる情報は、開発者が問題を診断するのに役立ちます。特に、StringToSign エラードキュメントの SignatureDoesNotMatch 要素を確認すると、システムで使用されているリクエスト標準化が正確にわかります。

ツールキットの中には、知らないヘッダーを通知なしで事前に挿入するものがあります。たとえば、PUT の実行中にヘッダー Content-Type が追加されることがあります。この場合、挿入されたヘッダーの値は一定であることがほとんどで、Ethereal、tcpmon などのツールを使用すると、不足しているヘッダーを検出できます。

## クエリ文字列による代替リクエスト認証

Authorization HTTP ヘッダーを使用する代わりに、必要な情報をクエリ文字列パラメータとして渡すことで、特定の種類のリクエストを認証できます。これは、サードパーティーのブラウザで、リクエストのプロキシを行わずにプライベートの Amazon S3 データに直接アクセスさせる場合に便利です。これを行うには、「署名付き」のリクエストを作成し、エンドユーザーのブラウザが取得できる URL としてエンコードします。さらに、署名付きのリクエストは、有効期限を指定することで制限できます。

クエリパラメータを使用するリクエストの認証の詳細については、Amazon Simple Storage Service API リファレンスの「[リクエストの認証: クエリパラメータの使用 \(AWS 署名バージョン 4\)](#)」を参照してください。AWS SDK を使用して署名付き URL を生成する例については、「[署名付き URL を使用したオブジェクトの共有](#)」を参照してください。

### 署名の作成

クエリ文字列で認証済みの Amazon S3 の REST リクエストの例を次に示します。

```
GET /photos/puppy.jpg
?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=1141889120&Signature=vjbyPxybdZaNmGa
%2ByT272YEAiv4%3D HTTP/1.1
```

Host: awsexamplebucket1.s3.us-west-1.amazonaws.com

Date: Mon, 26 Mar 2007 19:37:58 +0000

クエリ文字列リクエスト認証方法には、特別な HTTP ヘッダーは必要ありません。代わりに、必要な認証要素は、クエリ文字列パラメータとして指定します。

| クエリ文字列パラメータ名   | 値の例                              | 説明                                                                                                |
|----------------|----------------------------------|---------------------------------------------------------------------------------------------------|
| AWSAccessKeyId | AKIAIOSFODNN7EXAMPLE             | AWS アクセスキー ID。リクエストへの署名に使用する AWS シークレットアクセスキーを指定します。また、リクエストを行うデベロッパーのアイデンティティを間接的に指定します。         |
| Expires        | 1141889120                       | 署名の有効期限。エポック (1970 年 1 月 1 日の 00:00:00 UTC) からの時間が秒単位で指定されます。この時刻 (サーバーの時刻) より後に受信したリクエストは拒否されます。 |
| Signature      | vjbyPxybdZaNmGa%2ByT272YEAiv4%3D | StringToSign の HMAC-SHA1 の Base64 エンコーディングの URL エンコーディング。                                         |

クエリ文字列リクエスト認証方法は、通常の方法とは若干異なりますが、Signature リクエストパラメータおよび StringToSign 要素の形式だけが異なります。クエリ文字列リクエスト認証方法を示す疑似文法を次に示します。

```
Signature = URL-Encode( Base64( HMAC-SHA1( YourSecretAccessKey, UTF-8-Encoding-Of( StringToSign ) ) ) );
```

```
StringToSign = HTTP-VERB + "\n" +
  Content-MD5 + "\n" +
  Content-Type + "\n" +
  Expires + "\n" +
```

```
CanonicalizedAmzHeaders +  
CanonicalizedResource;
```

YourSecretAccessKey は、アマゾン ウェブ サービスのデベロッパーとしてサインアップするときに、Amazon によって割り当てられる AWS シークレットアクセスキー ID です。Signature が、クエリ文字列内に適切に配置されるために URL エンコードされている点に注目してください。また、HTTP では StringToSign であった位置要素が、Date では Expires に置き換えられていることに注意してください。CanonicalizedAmzHeaders および CanonicalizedResource は同じです。

#### Note

クエリ文字列認証メソッドでは、署名する文字列を計算する際に、Date または x-amz-date request ヘッダーを使用しません。

## クエリ文字列リクエスト認証

| リクエスト                                                                                                                                                                                                                                    | StringToSign                                                          |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| <pre>GET /photos/puppy.jpg?AWSAccess<br/>KeyId=AKIAIOSFODNN7EXAMPLE&amp;<br/>Signature=NpgCjnDzrM%2BWFzo<br/>ENXmpNDUsSn8%3D&amp;<br/>Expires=1175139620 HTTP/1.1<br/><br/>Host: awsexamplebucket1.s3.us-wes<br/>t-1.amazonaws.com</pre> | <pre>GET\n\n\n1175139620\n\n/awsexamplebucket1/photos/puppy.jpg</pre> |

ブラウザが GET リクエストを行うとき、Content-MD5 または Content-Type ヘッダーを提供しないこと、また、x-amz- ヘッダーを設定しないことが前提となっています。したがって、StringToSign のこの部分は空白のままです。

## Base64 エンコーディングの使用

HMAC リクエスト署名は、Base64 エンコードされている必要があります。Base64 エンコーディングでは、署名を、リクエストにアタッチできるシンプルな ASCII 文字列に変換します。プラス

(+)、スラッシュ (/)、等号 (=) などの署名文字列に含めることができる文字は、URI で使用する場合には、エンコードされている必要があります。たとえば、認証コードにプラス記号 (+) が含まれる場合は、リクエストで %2B としてエンコードします。スラッシュは %2F、等号は %3D でエンコードします。

Base64 エンコーディングの例については、Amazon S3 の「[認証の例](#)」を参照してください。

## POST (AWS 署名バージョン 2) を使用したブラウザベースのアップロード

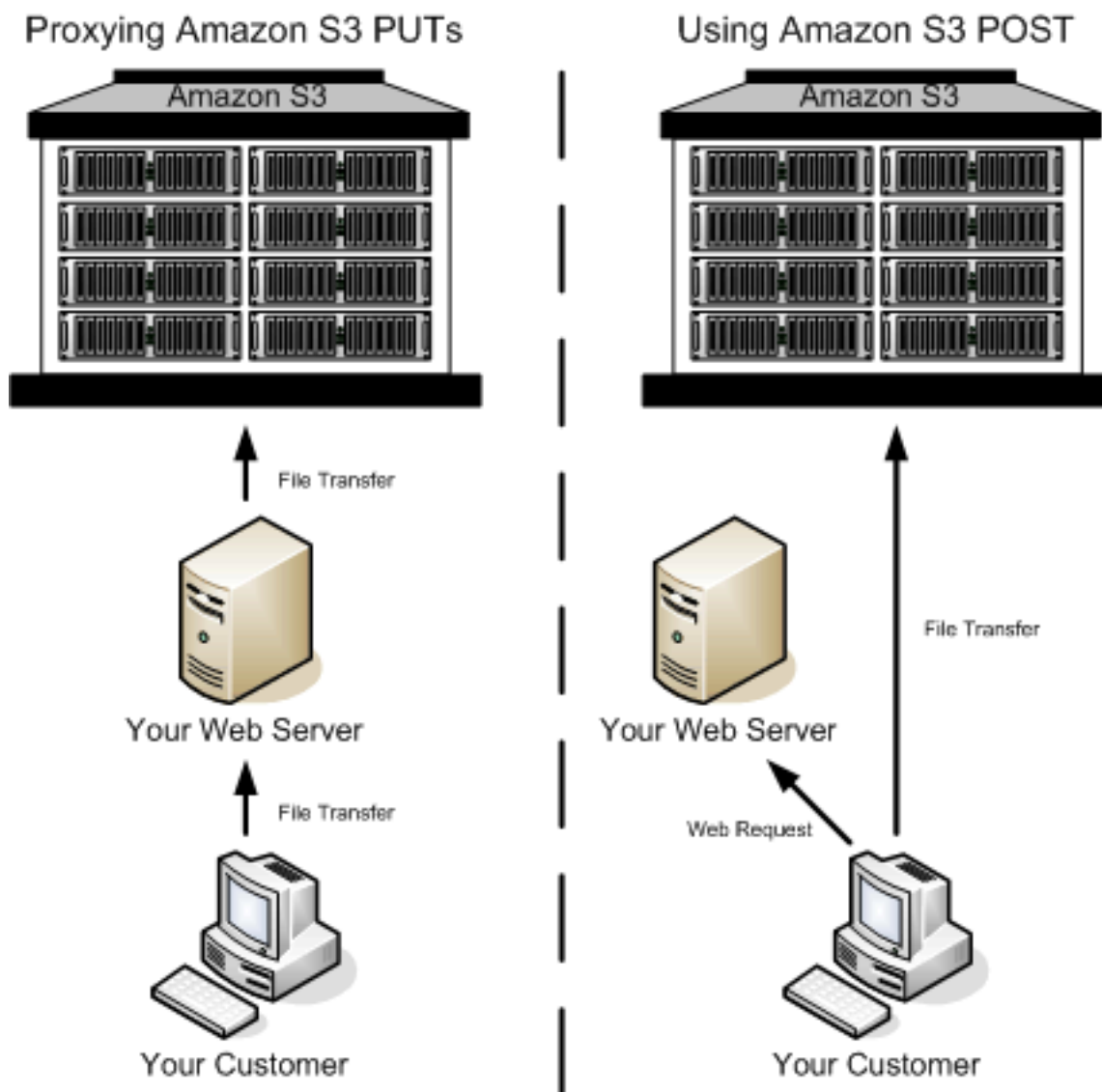
Amazon S3 では、POST がサポートされています。POST を使用すると、ユーザーがコンテンツを Amazon S3 に直接アップロードできます。POST は、アップロードを簡素化し、アップロードの待ち時間を短縮するように設計されています。また、ユーザーが Amazon S3 にデータをアップロードして保存するためのアプリケーションの費用を節約することもできます。

### Note

このセクションで説明するリクエスト認証は、AWS 署名バージョン 2 に基づいています。これは、AWS のサービスへのインバウンドの API リクエストを認証するためのプロトコルです。

Amazon S3 では、現在、署名バージョン 4 がサポートされています。署名バージョン 4 は、すべての AWS リージョンで AWS のサービスに対するインバウンドの API リクエストを認証するためのプロトコルです。現時点では、2014 年 1 月 30 日以前に作成された AWS リージョンは、以前のプロトコルである署名バージョン 2 のサポートを継続します。2014 年 1 月 30 日以降の新しいリージョンでは、署名バージョン 4 のみがサポートされます。そのため、これらのリージョンに対するすべてのリクエストは署名バージョン 4 で実行される必要があります。詳細については、Amazon Simple Storage Service API リファレンスの「[POST を使用したブラウザベースのアップロードでのリクエストの認証 \(AWS 署名バージョン 4\)](#)」を参照してください。

次の図は、Amazon S3 の POST を使用したアップロードを示しています。



### POST を使用したアップロード

- 1 ユーザーは、ウェブブラウザを開いて、開発者のウェブページにアクセスします。
- 2 ウェブページには、ユーザーがコンテンツを Amazon S3 にアップロードするために必要なすべての情報が含まれている HTTP フォームがあります。
- 3 ユーザーは、コンテンツを Amazon S3 に直接アップロードします。

#### Note

POST では、クエリ文字列認証がサポートされていません。

## HTML フォーム (AWS 署名バージョン 2)

### トピック

- [HTML フォームのエンコーディング](#)
- [HTML フォーム宣言](#)
- [HTML フォームフィールド](#)
- [ポリシーの作成](#)
- [署名の作成](#)
- [リダイレクト](#)

Amazon S3 と通信する場合、通常は REST API または SOAP API を使用して、put、get、delete などのオペレーションを実行します。POST では、ユーザーはデータをブラウザから Amazon S3 に直接アップロードします。SOAP API の処理や REST の PUT リクエストの作成は行えません。

#### Note

SOAP API for Amazon S3 は新規顧客には利用できず、2025 年 8 月 31 日にサポート終了 (EOL) となります。REST API か AWS SDK を使用することをお勧めします。

ユーザーがブラウザを使用してコンテンツを Amazon S3 にアップロードできるようにするには、HTML フォームを使用します。HTML フォームは、フォーム宣言とフォームフィールドで構成されます。フォーム宣言には、リクエストの概要情報が含まれています。フォームフィールドには、リクエストの詳細情報と、リクエストを認証し、指定した条件をそのリクエストが確実に満たすようにするためのポリシーが含まれています。

#### Note

フォームデータと境界 (ファイルのコンテンツは除く) は 20 KB を超えることはできません。

このセクションでは、HTML フォームを使用する方法について説明します。



## HTML フォームのエンコーディング

フォームとポリシーは UTF-8 エンコーディングされている必要があります。UTF-8 エンコードをフォームに適用するには、そのエンコードを HTML 見出しで指定するリクエストヘッダーとして指定します。

### Note

HTML フォーム宣言は、クエリ文字列認証パラメータを受け入れません。

HTML 見出しの UTF-8 エンコードの例を示します。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

リクエストヘッダーの UTF-8 エンコードの例を示します。

```
Content-Type: text/html; charset=UTF-8
```

## HTML フォーム宣言

フォーム宣言には、アクション、メソッド、エンクロージャタイプの 3 つのコンポーネントが含まれます。これらの値が適切に設定されていないと、リクエストは失敗します。

アクションは、リクエストを処理する URL を指定します。これはバケットの URL に設定されていなければなりません。例えば、バケットの名前が `awsexamplebucket1` で、リージョンが米国西部 (北カリフォルニア) の場合、URL は `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/` になります。

**Note**

キー名はフォームフィールドで指定されます。

メソッドは POST である必要があります。

エンクロージャタイプ (enctype) を指定する必要がある、ファイルアップロードとテキストエリアアップロードの両方に対して、multipart/form-data が設定されていなければなりません。詳細については、[RFC 1867](#) を参照してください。

**Example**

以下の例は、「awsexamplebucket1」バケットのフォーム宣言を示しています。

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

**HTML フォームフィールド**

以下の表に、HTML フォーム内で使用できるフィールドを示します。

**Note**

変数 \${filename} は、ユーザーによって指定されたファイル名に自動的に置き換えられ、すべてのフォームフィールドで認識されます。ブラウザまたはクライアントがファイルへの完全または部分的なパスを提供した場合、最後のスラッシュ (/) またはバックスラッシュ (\) 以降のテキストのみが使用されます。たとえば、「C:\Program Files\directory1\file.txt」は「file.txt」と解釈されます。ファイルまたはファイル名が指定されていない場合、変数は空の文字列に置き換えられます。

| フィールド名             | 説明                                                               | 必須   |
|--------------------|------------------------------------------------------------------|------|
| AWSSecretAccessKey | ポリシー内の一連の制約を満たすリクエストに対して匿名ユーザーアクセスを付与するバケット所有者の AWS アクセスキー ID。リク | 条件付き |

| フィールド名                                                                      | 説明                                                                                                                                                                                                                                                                                                                | 必須  |
|-----------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|                                                                             | エストにポリシードキュメントが含まれる場合、このフィールドは必須です。                                                                                                                                                                                                                                                                               |     |
| acl                                                                         | <p>Amazon S3 のアクセスコントロールリスト (ACL)。無効なアクセスコントロールリストが指定されると、エラーが生成されます。</p> <p>タイプ: 文字列</p> <p>デフォルト: プライベート</p> <p>有効な値: private   public-read   public-read-write   aws-exec-read   authenticated-read   bucket-owner-read   bucket-owner-full-control</p>                                                        | いいえ |
| Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires | REST 固有ヘッダー。詳細については、「 <a href="#">Put Object</a> 」を参照してください。                                                                                                                                                                                                                                                      | いいえ |
| key                                                                         | <p>アップロードされたキーの名前。</p> <p>ユーザーによって指定されたファイル名を使用するには、<code>\${filename}</code> 変数を使用します。たとえば、Betty がファイル lolcatz.jpg をアップロードし、開発者が <code>/user/betty/\${filename}</code> を指定すると、そのファイルは <code>/user/betty/lolcatz.jpg</code> として保存されます。</p> <p>詳細については、<a href="#">Working with Object Metadata</a> を参照してください。</p> | はい  |

| フィールド名                            | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 必須  |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| policy                            | リクエストで許可されているものについて説明するセキュリティポリシー。セキュリティポリシーがないリクエストは匿名と見なされ、誰でも書き込むことができるバケットでのみ動作します。                                                                                                                                                                                                                                                                                                                                                                                                                                                             | いいえ |
| success_action_redirect, redirect | <p>アップロードが成功したときにクライアントがリダイレクトされる URL。Amazon S3 は、バケット、キー、ETag の値をクエリ文字列パラメータとして URL に追加します。</p> <p>success_action_redirect が指定されていない場合、Amazon S3 は、success_action_status フィールドで指定されている空のドキュメントタイプを返します。</p> <p>Amazon S3 が URL を解釈できない場合、フィールドは無視されます。</p> <p>アップロードが失敗した場合、Amazon S3 はエラーを表示し、ユーザーを URL にリダイレクトしません。</p> <p>詳細については、「<a href="#">リダイレクト</a>」を参照してください。</p> <div><p> <b>Note</b></p><p>リダイレクトフィールド名は廃止されています。このフィールド名のサポートは削除される予定です。</p></div> | いいえ |

| フィールド名                                                                                                                                                                                                                                                      | 説明                                                                                                                                                                                                                                                                                                                                                                                                                 | 必須  |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| success_action_status                                                                                                                                                                                                                                       | <p>アップロードが正常に行われたとき、success_action_redirect が指定されていない場合にクライアントに返されるステータスコード。</p> <p>有効な値は、200、201、または 204 (デフォルト) です。</p> <p>値が 200 または 204 に設定されている場合、Amazon S3 は空のドキュメントとステータスコード 200 または 204 を返します。</p> <p>値が 201 に設定されている場合、Amazon S3 は XML ドキュメントとステータスコード 201 を返します。XML ドキュメントの内容については、「<a href="#">POST Object</a>」を参照してください。</p> <p>値が設定されていないか無効な値が設定されている場合は、Amazon S3 は空のドキュメントとステータスコード 204 を返します。</p> | いいえ |
| <div> <b>Note</b></div> <p>Adobe Flash Player のバージョンによっては、本文が空の HTTP レスポンスが適切に処理されないことがあります。Adobe Flash でアップロードをサポートするには、success_action_status を 201 に設定することをお勧めします。</p> |                                                                                                                                                                                                                                                                                                                                                                                                                    |     |

| フィールド名                                        | 説明                                                                                                                                                                                                                                                                                                 | 必須   |
|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| signature                                     | <p>HMAC 署名は、提供された <code>AWSSignatureVersion</code> に対応するシークレットアクセスキーを使用して構築されます。このフィールドは、ポリシードキュメントがリクエストに含まれる場合に必要です。</p> <p>詳細については、「<a href="#">Amazon S3 用 Identity and Access Management</a>」を参照してください。</p>                                                                                   | 条件付き |
| x-amz-security-token                          | <p>セッション認証情報で使用するセキュリティトークン</p> <p>リクエストで Amazon DevPay を使用する場合は、製品トークン用とユーザートークン用の 2 つの <code>x-amz-security-token</code> フォームフィールドが必要です。</p> <p>リクエストでセッション認証情報が使用される場合は、単一の <code>x-amz-security-token</code> フォームが必要です。詳細については、IAM ユーザーガイドの「<a href="#">IAM の一時的なセキュリティ認証情報</a>」を参照してください。</p> | いいえ  |
| プレフィックスが <code>x-amz-meta-</code> のその他のフィールド名 | <p>ユーザーが指定したメタデータ。</p> <p>Amazon S3 では、このデータは検証または使用されません。</p> <p>詳細については、「<a href="#">PutObject</a>」を参照してください。</p>                                                                                                                                                                                | いいえ  |

| フィールド名 | 説明                                                                                                                            | 必須 |
|--------|-------------------------------------------------------------------------------------------------------------------------------|----|
| file   | <p>ファイルまたはテキストコンテンツ。</p> <p>ファイルまたはコンテンツはフォームの最後のフィールドでなければなりません。その下のフィールドはすべて無視されます。</p> <p>複数のファイルを一度にアップロードすることはできません。</p> | はい |

## ポリシーの作成

### トピック

- [有効期限](#)
- [条件](#)
- [条件一致](#)
- [文字のエスケープ](#)

ポリシーは、UTF-8 および Base64 エンコードされた JSON ドキュメントで、リクエストが満たさなければならない条件を指定します。また、コンテンツを認証するときにも使用されます。ポリシードキュメントは、どのように設計するかに応じて、アップロードごと、ユーザーごと、すべてのアップロードに対して、またはニーズに応じたその他の設計にしたがって使用できます。

#### Note

ポリシードキュメントはオプションですが、バケットを誰でも書き込むことができるようにするよりも、ポリシードキュメントを使用することを強くお勧めします。

ポリシードキュメントの例を次に示します。

```
{ "expiration": "2007-12-01T12:00:00.000Z",  
  
  "conditions": [
```

```
{ "acl": "public-read" },  
  
{ "bucket": "awsexamplebucket1" },  
  
[ "starts-with", "$key", "user/eric/" ],  
  
]  
  
}
```

ポリシードキュメントには、有効期限と条件が含まれます。

### 有効期限

有効期限要素は、ポリシーの有効期限日を ISO 8601 UTC 日付形式で指定します。たとえば「2007-12-01T12:00:00.000Z」は、2007 年 12 月 1 日の 12:00 UTC にポリシーが無効になることを指定します。ポリシーには有効期限が必ず必要です。

### 条件

ポリシードキュメントの条件は、アップロードされたオブジェクトのコンテンツを検証します。条件の一覧には、フォームで指定した各フォームフィールドが含まれていなければなりません (AWSAccessKeyId、署名、ポリシー、および x-ignore- プレフィックスが付いているフィールド名を除く)。

#### Note

同じ名前のフィールドが複数ある場合、そのフィールドの値はコンマで区切る必要があります。たとえば、「x-amz-meta-tag」という名前のフィールドが 2 つあり、最初のフィールドの値が「Ninja」、2 つ目のフィールドの値が「Stallman」の場合、ポリシードキュメントは Ninja,Stallman に設定します。

フォーム内のすべての変数は、ポリシーの検証前に展開されます。したがって、条件との照合は必ず、展開されたフィールドに対して行われます。たとえば、キーフィールドを user/betty/\${filename} に設定した場合、ポリシーは [ "starts-with", "\$key", "user/betty/" ] になる可能性があります。[ "starts-with", "\$key", "user/betty/\${filename}" ] は入力しないでください。詳細については、「[条件一致](#)」を参照してください。

以下の表にポリシードキュメント条件を示します。



| 要素名                                                                     | 説明                                                                                                                                                         |
|-------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| acl                                                                     | ACL が満たす必要がある条件を指定します。<br><br>完全一致および starts-with をサポートします。                                                                                                |
| content-length-range                                                    | アップロードされたコンテンツの最小サイズと最大サイズを指定します。<br><br>範囲一致をサポートします。                                                                                                     |
| Cache-Control、Content-Type、Content-Disposition、Content-Encoding、Expires | REST 固有ヘッダー。<br><br>完全一致および starts-with をサポートします。                                                                                                          |
| key                                                                     | アップロードされたキーの名前。<br><br>完全一致および starts-with をサポートします。                                                                                                       |
| success_action_redirect、redirect                                        | アップロードが成功したときにクライアントがリダイレクトされる URL。<br><br>完全一致および starts-with をサポートします。                                                                                   |
| success_action_status                                                   | アップロードが正常に行われたとき、success_action_redirect が指定されていない場合にクライアントに返されるステータスコード。<br><br>完全一致をサポートします。                                                             |
| x-amz-security-token                                                    | Amazon DevPay のセキュリティトークン。<br><br>Amazon DevPay を使用する各リクエストでは、製品トークン用とユーザートークン用の 2 つの x-amz-security-token フォームフィールドが必要です。結果的に、値はコンマで区切られていなければなりません。たとえば、 |

| 要素名                                           | 説明                                                                                                                                                                         |
|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                               | ユーザートークンが <code>eW91dHViZQ==</code> で、製品トークンが <code>b0hnNVNKWVJIQTA=</code> の場合、ポリシーエントリは <code>{ "x-amz-security-token": "eW91dHViZQ==,b0hnNVNKWVJIQTA=" }</code> に設定します。 |
| プレフィックスが <code>x-amz-meta-</code> のその他のフィールド名 | <p>ユーザーが指定したメタデータ。</p> <p>完全一致および <code>starts-with</code> をサポートします。</p>                                                                                                   |

### Note

ツールキットにより追加のフィールドが追加されている場合は (Flash では filename が追加される)、そのフィールドをポリシードキュメントに追加する必要があります。この機能を管理できる場合は、そのフィールドの機能が Amazon S3 に無視されるようにプレフィックスの `x-ignore-` をフィールドに追加します。これは、そのフィールドの機能の将来のバージョンには影響しません。

## 条件一致

以下の表に、条件一致タイプを示します。フォーム内で指定したフォームフィールドごとに 1 つの条件を指定する必要がありますが、1 つのフォームフィールドに複数の条件を指定し、より複雑な一致条件を作成することもできます。

| 条件   | 説明                                                                                                                                                                                                         |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 完全一致 | <p>完全一致では、フィールドが特定の値と一致するかどうかを確認します。この例は、ACL が <code>public-read</code> に設定される必要があることを示しています:</p> <pre>{"acl": "public-read" }</pre> <p>この例は、ACL が <code>public-read</code> に設定される必要があることを別の方法で示しています:</p> |

| 条件          | 説明                                                                                                                                                                      |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <pre>[ "eq", "\$acl", "public-read" ]</pre>                                                                                                                             |
| 先頭が一致       | <p>先頭の値を指定する場合は、starts-with を使用します。次の例は、キーが user/betty で開始する必要があることを示しています。</p> <pre>[ "starts-with", "\$key", "user/betty/" ]</pre>                                    |
| 任意のコンテンツに一致 | <p>フィールド内の任意のコンテンツを許可するようにポリシーを設定するには、starts-with に空の値を使用します。次の例は、任意の success_action_redirect を許可します。</p> <pre>[ "starts-with", "\$success_action_redirect", "" ]</pre> |
| 範囲を指定       | <p>範囲指定できるフィールドについては、範囲の上限値と下限値をコンマで区切ります。次の例は、1～10 メガバイトのサイズのファイルを許可します。</p> <pre>[ "content-length-range", 1048579, 10485760 ]</pre>                                  |

## 文字のエスケープ

以下の表に、ポリシードキュメント内でエスケープする必要のある文字を示します。

| エスケープシーケンス | 説明       |
|------------|----------|
| \\         | バックスラッシュ |
| \\\$       | ドル記号     |

| エスケープシーケンス          | 説明              |
|---------------------|-----------------|
| <code>\b</code>     | Backspace       |
| <code>\f</code>     | フォームフィード        |
| <code>\n</code>     | 改行              |
| <code>\r</code>     | キャリッジリターン       |
| <code>\t</code>     | 水平タブ            |
| <code>\v</code>     | 垂直タブ            |
| <code>\uxxxx</code> | すべての Unicode 文字 |

## 署名の作成

| ステップ | 説明                                        |
|------|-------------------------------------------|
| 1    | UTF-8 を使用してポリシーをエンコードします。                 |
| 2    | Base64 を使用して、それらの UTF-8 バイトをエンコードします。     |
| 3    | HMAC SHA-1 を使用して、シークレットアクセスキーでポリシーに署名します。 |
| 4    | Base64 を使用して、SHA-1 署名をエンコードします。           |

認証の一般的な情報については、「[Amazon S3 用 Identity and Access Management](#)」を参照してください。

## リダイレクト

このセクションでは、リダイレクトを処理する方法について説明します。

### 一般的なリダイレクト

POST リクエストの完了時に、ユーザーは、フィールドで指定した場所にリダイレクトされます。success\_action\_redirect Amazon S3 が URL を解釈できない場合、success\_action\_redirect フィールドは無視されます。

success\_action\_redirect が指定されていない場合、Amazon S3 は success\_action\_status フィールドで指定されている空のドキュメントタイプを返します。

POST リクエストが失敗した場合、Amazon S3 はエラーを表示し、リダイレクトを行いません。

### アップロード前のリダイレクト

<CreateBucketConfiguration> を使用してバケットが作成された場合、エンドユーザーをリダイレクトしなければならない可能性があります。これが発生した場合、ブラウザによっては、リダイレクトが正しく処理されないことがあります。比較的まれにですが、バケットの作成直後に発生する可能性があります。

## アップロードの例 (AWS 署名バージョン 2)

### トピック

- [ファイルのアップロード](#)
- [テキストエリアのアップロード](#)

#### Note

このセクションで説明するリクエスト認証は、AWS 署名バージョン 2 に基づいています。これは、AWS のサービスへのインバウンドの API リクエストを認証するためのプロトコルです。

Amazon S3 では、現在、署名バージョン 4 がサポートされています。署名バージョン 4 は、すべての AWS リージョンで AWS のサービスに対するインバウンドの API リクエストを認証するためのプロトコルです。現時点では、2014 年 1 月 30 日以前に作成された AWS リージョンは、以前のプロトコルである署名バージョン 2 のサポートを継続します。2014 年 1 月 30 日以降の新しいリージョンでは、署名バージョン 4 のみがサポートされます。そ

のため、これらのリージョンに対するすべてのリクエストは署名バージョン 4 で実行される必要があります。詳細については、Amazon Simple Storage Service API リファレンスの「[例: HTTP POST を使用したブラウザベースのアップロード \(AWS 署名バージョン 4 を使用\)](#)」を参照してください。

## ファイルのアップロード

この例は、ポリシーおよびフォームを作成し、添付ファイルをアップロードするプロセスを示しています。

### ポリシーとフォームの作成

以下のポリシーは、Amazon S3 の `awsexamplebucket1` バケットへのアップロードをサポートしています。

```
{  "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

このポリシーでは、以下のように規定されています。

- アップロードは、2007 年 12 月 1 日の 12:00 UTC より前に行う必要があります。
- コンテンツは `awsexamplebucket1` バケットにアップロードする必要があります。
- キーの先頭は「`user/eric/`」でなければなりません。
- ACL は `public-read` に設定されています。
- `success_action_redirect` が `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html` に設定されています。
- オブジェクトはイメージファイルです。

- x-amz-meta-uuid タグが 14365123651274 に設定されている必要があります。
- x-amz-meta-tag には任意の値を含めることができます。

このポリシーの Base64 エンコードされたバージョンを次に示します。

eyJ0eXhwaXJhdGlubiI6ICIyMDA3LTExLTAxVDEyOjAwMjAwLjAwMFoiLAogICJjb25kaXRpb25zIjoqWwogICAgeyJidWN

認証情報を使用すると署名が作成されます。たとえば、0RavWzkygo6QX9caELEqKi9kDbU= は、前述のポリシードキュメントの署名です。

以下のフォームは、このポリシーを使用する `amzn-s3-demo-bucket` バケットへの POST リクエストをサポートします。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
    ...
    <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
enctype="multipart/form-data">
      Key to upload: <input type="input" name="key" value="user/eric/" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html" />
      Content-Type: <input type="input" name="Content-Type" value="image/jpeg" /><br />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
      <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
      <input type="hidden" name="Policy" value="POLICY" />
      <input type="hidden" name="Signature" value="SIGNATURE" />
      File: <input type="file" name="file" /> <br />
      <!-- The elements after this will be ignored -->
      <input type="submit" name="submit" value="Upload to Amazon S3" />
    </form>
    ...
  </html>
```

## リクエスト例

このリクエストでは、アップロードされたイメージが 117,108 バイト (イメージデータを除く) であることを前提としています。

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
  Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: 118698

--9431149156168
Content-Disposition: form-data; name="key"

user/eric/MyPicture.jpg
--9431149156168
Content-Disposition: form-data; name="acl"

public-read
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html
--9431149156168
Content-Disposition: form-data; name="Content-Type"

image/jpeg
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

Some,Tag,For,Picture
--9431149156168
```



```
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--9431149156168
Content-Disposition: form-data; name="Policy"

eyJhZXBwaXJhdGlvbiI6ICIyMDA3LTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgJidWN
--9431149156168
Content-Disposition: form-data; name="Signature"

0RavWzkygo6QX9caELEqKi9kDbU=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

...file content...
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

## レスポンス例

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html?bucket=awsexamplebucket1&key=user/eric/
MyPicture.jpg&etag="39d459dfbc0faabbb5e179358dfb94c3"
Server: AmazonS3
```

## テキストエリアのアップロード

### トピック

- [ポリシーとフォームの作成](#)
- [リクエスト例](#)
- [レスポンス例](#)

この例は、ポリシーおよびフォームを作成し、テキストエリアをアップロードするプロセスを示しています。テキストエリアのアップロードは、ブログ投稿のようなユーザー作成コンテンツの送信に便利です。

## ポリシーとフォームの作成

以下のポリシーは、Amazon S3 の `awsexamplebucket1` バケットへのテキストのアップロードをサポートしています。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html"},
    ["eq", "$Content-Type", "text/html"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

このポリシーでは、以下のように規定されています。

- アップロードは 2007 年 12 月 1 日 12:00 GMT よりも前に行う必要があります。
- コンテンツは `awsexamplebucket1` バケットにアップロードする必要があります。
- キーの先頭は「`user/eric/`」でなければなりません。
- ACL は `public-read` に設定されています。
- `success_action_redirect` が `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html` に設定されています。
- オブジェクトは HTML テキストです。
- `x-amz-meta-uuid` タグが 14365123651274 に設定されている必要があります。
- `x-amz-meta-tag` には任意の値を含めることができます。

このポリシーの Base64 エンコードされたバージョンを次に示します。

```
eyJhZiZlXhwaXJhdGlvbiI6IClYMDA3LDEyLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXR
```

```
pb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXR0In0sCiAgICBbInN0YXJ0cy13aXR0IiwgIiRrZXkiLCaidXNlci
LAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2p
C5zMy5hbWV6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSI9ICJ0ZXh0L2h0
CAgIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXR0IiwgIiR4LWFtei1t
IsICIiXQogIF0KfQo=
```

認証情報を使用すると署名が作成されます。たとえば、qA7FWXKq6VvU68lI9KdveT1cWgF= は、前述のポリシードキュメントの署名です。

以下のフォームは、このポリシーを使用する amzn-s3-demo-bucket バケットへの POST リクエストをサポートします。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
    ...
    <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
    enctype="multipart/form-data">
      Key to upload: <input type="input" name="key" value="user/eric/" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="https://
      awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html" />
      <input type="hidden" name="Content-Type" value="text/html" />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
      <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
      <input type="hidden" name="Policy" value="POLICY" />
      <input type="hidden" name="Signature" value="SIGNATURE" />
      Entry: <textarea name="file" cols="60" rows="10">

      Your blog post goes here.

      </textarea><br />
      <!-- The elements after this will be ignored -->
      <input type="submit" name="submit" value="Upload to Amazon S3" />
    </form>
    ...
  </html>
```

## リクエスト例

このリクエストでは、アップロードされたイメージが 117,108 バイト (イメージデータを除く) であることを前提としています。

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
  Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=178521717625888
Content-Length: 118635

-178521717625888
Content-Disposition: form-data; name="key"

ser/eric/NewEntry.html
--178521717625888
Content-Disposition: form-data; name="acl"

public-read
--178521717625888
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html
--178521717625888
Content-Disposition: form-data; name="Content-Type"

text/html
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-tag"

Interesting Post
--178521717625888
```

```
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--178521717625888
Content-Disposition: form-data; name="Policy"
eyJhZlZxhwaXJhdGlvbiI6IClYMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidW
--178521717625888
Content-Disposition: form-data; name="Signature"

qA7FWXKq6VvU68lI9KdveT1cWgF=
--178521717625888
Content-Disposition: form-data; name="file"

...content goes here...
--178521717625888
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--178521717625888--
```

## レスポンス例

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html?
bucket=awsexamplebucket1&key=user/eric/
NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4
Server: AmazonS3
```

## Adobe Flash を使用した POST (AWS 署名バージョン 2)

このセクションでは、Adobe Flash で POST を使用する方法について説明します。

### Adobe Flash Player のセキュリティ

デフォルトでは、Adobe Flash Player のセキュリティモデルにより、Adobe Flash Player が、SWF ファイルを提供するドメイン外のサーバーにネットワーク接続することはできません。

このデフォルトの設定を上書きするには、公開されている `crossdomain.xml` ファイルを、POST アップロードを受け入れるバケットにアップロードする必要があります。`crossdomain.xml` ファイルの例を次に示します。

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```

#### Note

Adobe Flash セキュリティモデルの詳細については、Adobe のウェブサイトを参照してください。

`crossdomain.xml` ファイルをバケットに追加すると、Adobe Flash Player はバケット内の `crossdomain.xml` に接続できるようになります。ただし、実際の Amazon S3 バケットへのアクセス許可は付与されません。

## Adobe Flash に関する考慮事項

Adobe Flash の FileReference API により、Filename フォームフィールドが POST リクエストに追加されます。FileReference API アクションを使用して Amazon S3 にアップロードを行う Adobe Flash アプリケーションを作成する場合は、次の条件をポリシーに含めます。

```
['starts-with', '$Filename', '']
```

Adobe Flash Player のバージョンによっては、本文が空の HTTP レスポンスが適切に処理されないことがあります。本文が空でないレスポンスを返すように POST を設定するには、`success_action_status` を 201 に設定します。これにより、Amazon S3 はステータスコード 201 で XML ドキュメントを返します。XML ドキュメントの内容については、「[POST Object](#)」を参照してください。フォームフィールドの詳細については、「[HTML フォームフィールド](#)」を参照してください。

## 付録: ライフサイクル設定 API (廃止)

バケットライフサイクル設定が更新され、オブジェクトタグに基づくフィルタがサポートされるようになりました。つまり、キー名のプレフィックス、1 つ以上のオブジェクトタグ、またはその両方を

指定するルールを指定して、ルールを適用するオブジェクトのサブセットを選択できるようになりました。API もそれに応じて更新されています。以下のトピックでは、下位互換性のための PUT および GET バケットライフサイクルオペレーションの以前のバージョンについて説明します。

## トピック

- [PUT バケットライフサイクル \(廃止\)](#)
- [GET バケットライフサイクル \(廃止\)](#)

# PUT バケットライフサイクル (廃止)

## 説明

### Important

この API の更新バージョンについては、[PutBucketLifecycleConfiguration](#) を参照してください。このバージョンは非推奨です。既存のライフサイクル設定は引き続き使用できます。新しいライフサイクル設定については、更新された API を使用してください。

バケットの新しいライフサイクル設定を作成するか、既存のライフサイクル設定を置き換えます。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[オブジェクトのライフサイクルの管理](#)」を参照してください。

## アクセス許可

デフォルトでは、バケット、オブジェクト、関連サブリソース (例:ライフサイクル設定やウェブサイト設定) を含むすべての Amazon S3 リソースはプライベートです。リソースの所有者 (リソースを作成した AWS アカウント) のみが、リソースにアクセスできます。リソースの所有者は、アクセスポリシーを作成することにより、オプションで他のユーザーにアクセス許可を付与できます。このオペレーションでは、ユーザーは `s3:PutLifecycleConfiguration` アクセス許可を取得する必要があります。

アクセス許可を明示的に拒否することもできます。明示的な拒否は、他のあらゆるアクセス許可よりも優先されます。ユーザーまたはアカウントがバケットからオブジェクトを削除できないようにするには、次のアクションのアクセス許可を拒否する必要があります。

- `s3:DeleteObject`
- `s3:DeleteObjectVersion`
- `s3:PutLifecycleConfiguration`

アクセス許可の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[Amazon S3 リソースへのアクセス許可の管理](#)」を参照してください。



# リクエスト

## 構文

```
PUT /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Content-Length: length
Date: date
Authorization: authorization string
Content-MD5: MD5
```

*Lifecycle configuration in the request body*

認可文字列の詳細については、「[リクエストの認証 \(AWS 署名バージョン 4\)](#)」を参照してください。

## リクエストパラメータ

オペレーションの実装では、リクエストパラメータを使用しません。

## リクエストヘッダー

| 名前          | 説明                                                                                                                                                                                      | 必須 |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Content-MD5 | <p>データの Base64 エンコードされた 128 ビット MD5 ダイジェスト。このヘッダーをメッセージ整合性チェックとして使用して、リクエスト本文が転送中に破損していないことを確認する必要があります。詳細については、<a href="#">RFC 1864</a>を参照してください。</p> <p>タイプ: 文字列</p> <p>デフォルト: なし</p> | はい |

## リクエスト本文

リクエストでは、リクエスト本文でライフサイクル設定を指定します。ライフサイクル設定は XML 形式で指定します。以下は、基本的なライフサイクル設定の一例です。1 つのルールを指定します。ルール内の Prefix は、ルールが適用されるオブジェクトを特定します。このルールでは、2 つのア

クシオン (Transition と Expiration) も指定します。各アクションは、Amazon S3 がアクションを実行するタイムラインを指定します。Status は、ルールが有効が無効かを示します。

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>key-prefix</Prefix>
    <Status>rule-status</Status>
    <Transition>
      <Date>value</Date>
      <StorageClass>storage class</StorageClass>
    </Transition>
    <Expiration>
      <Days>value</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

バケットの状態が、バージョニングが有効か、またはバージョニングが停止している場合、同じオブジェクトの複数のバージョン (現行バージョン 1 つと 非現行バージョン 0 個以上) を保持できます。次のライフサイクル設定では、現在のものではないオブジェクトバージョンに固有のアクション (NoncurrentVersionTransition、NoncurrentVersionExpiration) を指定します。

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>key-prefix</Prefix>
    <Status>rule-status</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>value</NoncurrentDays>
      <StorageClass>storage class</StorageClass>
    </NoncurrentVersionTransition>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>value</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

マルチパートアップロード API を使用すると、大容量オブジェクトをいくつかに分けてアップロードできます。マルチパートアップロードの詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[マルチパートアップロードの概要](#)」を参照してください。ライフサイクル設定を

使用すると、マルチパートアップロードが指定の日数以内に完了しなかった場合、不完全なマルチパートアップロード (ルールで指定されたキー名プレフィックスによって識別される) を停止するように Amazon S3 に指示できます。Amazon S3 は、マルチパートアップロードをキャンセルするとき、アップロードに関連付けられているすべてのパートを削除します。これにより、Amazon S3 に一部のパートが保存されたままの不完全なマルチパートアップロードは存在しなくなるため、それらに対してストレージコストを支払う必要はありません。AbortIncompleteMultipartUpload アクションにルールを指定するライフサイクル設定の例を次に示します。このアクションは、開始から 7 日後に不完全なマルチパートアップロードをキャンセルするように Amazon S3 に指示します。

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>SomeKeyPrefix</Prefix>
    <Status>rule-status</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

以下のセクションでは、ライフサイクル設定における XML 要素について説明します。

| 名前                             | 説明                                                                                                              | 必須                             |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------|--------------------------------|
| AbortIncompleteMultipartUpload | 不完全なマルチパートアップロードが中止オペレーションの対象となるタイミングを指定するためのコンテナ。<br><br>子: DaysAfterInitiation<br><br>型: コンテナ<br><br>祖先: Rule | はい (ルールに他のアクションが指定されていない場合)    |
| Date                           | Amazon S3 がアクションを実行する日付。詳細については、「Amazon Simple Storage Service ユーザーガイド」中の「 <a href="#">ライフサイクル</a> 」            | はい (Days と ExpiredObjectDelete |

| 名前                  | 説明                                                                                                                                                                                        | 必須                                                |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|
|                     | <p><a href="#">ルール: 特定の日付に基づく</a>」を参照してください。</p> <p>日付の値は、ISO 8601 形式に従う必要があります。この時刻は常に午前 0 時 (UTC) となります。</p> <p>タイプ: 文字列</p> <p>祖先: Expiration または Transition</p>                       | eMarker<br>がない場合)                                 |
| Days                | <p>オブジェクトの作成後、特定のルールアクションが有効になるまでの日数を指定します。</p> <p>型: Transition で使用する場合は非負整数、Expiration で使用する場合は正整数</p> <p>祖先: Expiration 、 Transition</p>                                               | はい<br>(Date と ExpiredObjectDeleteMarker<br>がない場合) |
| DaysAfterInitiation | <p>マルチパートアップロードを開始してから、そのマルチパートアップロードを完了する必要がある日数を指定します。指定した日数で完了しない場合、中止オペレーションの対象となり、Amazon S3 は不完全なマルチパートアップロードをキャンセルします。</p> <p>型: 正の整数</p> <p>祖先: AbortIncompleteMultipartUpload</p> | はい (親タグが指定されている場合)                                |

| 名前         | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 必須                         |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| Expiration | <p data-bbox="634 258 1284 531">このアクションは、オブジェクトの有効期間内で、Amazon S3 が適切な失効アクションを実行する期間を指定します。Amazon S3 が実行するアクションは、バケットでバージョンングが有効になっているかどうかによって異なります。</p> <ul data-bbox="634 583 1284 1161" style="list-style-type: none"><li data-bbox="634 583 1284 741">• バケットでバージョンングが有効になっていない場合、Amazon S3 はオブジェクトの唯一のコピーを完全に削除します。</li><li data-bbox="634 772 1284 1161">• バージョニングが有効になっているバケット (またはバージョンングが停止されている) の場合、アクションは現在のバージョンのオブジェクトにのみ適用されます。バージョンングが有効なバケットは、同じオブジェクトの複数のバージョン (1 個の現行バージョンと 0 個以上の非現行バージョン) を持つことができます。</li></ul> <p data-bbox="664 1203 1271 1381">Amazon S3 は、現在のバージョンを削除する代わりに、削除マーカを新しい現在のバージョンとして追加することで、そのバージョンを非現行バージョンとします。</p> <div data-bbox="667 1455 1304 1785"><p data-bbox="695 1497 881 1528"> Important</p><p data-bbox="743 1549 1263 1780">バケットがバージョンング停止状態にある場合、Amazon S3 はバージョン ID として null を使用する削除マーカを作成します。バージョン ID が null のバージョンがある場</p></div> | はい (Rule に他のアクションが存在しない場合) |

| 名前 | 説明                                                                                                                         | 必須  |
|----|----------------------------------------------------------------------------------------------------------------------------|-----|
|    | <div>合、Amazon S3 はそのバージョンを上書きします。</div> <div>Note<br/>非現行オブジェクトの有効期限を設定するには、NoncurrentVersionExpiration アクションを使用します。</div> |     |
|    | 型: コンテナ<br><br>子: Days または Date<br><br>祖先: Rule                                                                            |     |
| ID | ルール用の一意の識別子。値は 255 文字以下にする必要があります。<br><br>タイプ: 文字列<br><br>祖先: Rule                                                         | いいえ |

| 名前                        | 説明                                                                                                                                                                                                                                                                                                                                                                                                                      | 必須                     |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| LifecycleConfiguration    | <p>ライフサイクルルールのコンテナ。最大 1,000 個のルールを追加できます。</p> <p>型: コンテナ</p> <p>子: Rule</p> <p>祖先: なし</p>                                                                                                                                                                                                                                                                                                                               | はい                     |
| ExpiredObjectDeleteMarker | <p>バージョン管理されたバケット (バージョニングが有効なバケットまたはバージョニングが停止されたバケット) では、この要素をライフサイクル設定に追加して、期限切れのオブジェクト削除マーカを削除するように Amazon S3 に指示できます。例については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">例 8: 期限切れオブジェクト削除マーカの削除</a>」を参照してください。バージョン管理されていないバケットには追加しないでください。このタイプのバケットには削除マーカを含めることができないためです。</p> <p>タイプ: 文字列</p> <p>有効な値: true   false (値 false は許可されますが、no-op です。つまり、Amazon S3 はアクションを実行しません)</p> <p>祖先: Expiration</p> | はい (Date と Days がない場合) |

| 名前             | 説明                                                                                                                                                                                                                                                                                                                                                                            | 必須 |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| NoncurrentDays | <p>Amazon S3 で関連付けられたアクションが実行されるまでオブジェクトが最新でなくなつてからの日数を指定します。非現行になつてからの日数の計算の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">オブジェクトが最新でなくなつてからの期間を Amazon S3 が計算する方法</a>」を参照してください。</p> <p>型: NoncurrentVersionTransition</p> <p>で使用する場合は非負整数、NoncurrentVersionExpiration で使用する場合は正整数</p> <p>祖先: NoncurrentVersionExpiration または NoncurrentVersionTransition</p> | はい |



| 名前                          | 説明                                                                                                                                                                                                                                                                               | 必須                         |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| NoncurrentVersionExpiration | <p>最新ではないオブジェクトのバージョンが期限切れになる時期を指定します。期限切れになると、Amazon S3 により、最新ではないオブジェクトのバージョンが完全に削除されます。</p> <p>バージョニングが有効 (または停止) になっているバケットにこのライフサイクル設定アクションを構成し、オブジェクトのライフサイクル内の特定の期間に非現行オブジェクトのバージョンを削除するように Amazon S3 に指示します。</p> <p>型: コンテナ</p> <p>子: NoncurrentDays</p> <p>祖先: Rule</p> | はい (Rule に他のアクションが存在しない場合) |
| NoncurrentVersionTransition | <p>非現行オブジェクトが、STANDARD_IA 、ONEZONE_IA 、または ストレージクラスに移行する時期を記述する移行ルールのコンテナ。</p> <p>バケットでバージョニングが有効な場合 (またはバージョニングが停止している場合)、このアクションを設定して、オブジェクトの有効期間内の特定の期間にオブジェクトの非現行バージョンを移行するように Amazon S3 に指示できます。</p> <p>型: コンテナ</p> <p>子: NoncurrentDays と StorageClass</p> <p>祖先: Rule</p> | はい (Rule に他のアクションが存在しない場合) |

| 名前           | 説明                                                                                                                                                                 | 必須                                               |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| Prefix       | <p>ルールの適用対象となるオブジェクトを 1 つ以上特定するオブジェクトキープレフィックス。</p> <p>タイプ: 文字列</p> <p>祖先: Rule</p>                                                                                | はい                                               |
| Rule         | <p>ライフサイクルルールのコンテナ。ライフサイクル設定には、最大 1,000 個のルールを含めることができます。</p> <p>型: コンテナ</p> <p>祖先: LifecycleConfiguration</p>                                                     | はい                                               |
| Status       | <p>有効にすると、Amazon S3 はスケジュールどおりにルールを実行します。無効にすると、Amazon S3 はルールを無視します。</p> <p>タイプ: 文字列</p> <p>祖先: Rule</p> <p>有効な値: Enabled、Disabled</p>                            | はい                                               |
| StorageClass | <p>オブジェクトの移行先となる Amazon S3 ストレージクラスを指定します。</p> <p>タイプ: 文字列</p> <p>祖先: Transition と NoncurrentVersionTransition</p> <p>有効な値: STANDARD_IA   ONEZONE_IA   GLACIER</p> | <p>はい</p> <p>この要素は、一方または両方の祖先を指定する場合にのみ必要です。</p> |

| 名前         | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 必須                         |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| Transition | <p>このアクションは、オブジェクトの有効期間内で Amazon S3 がオブジェクトを STANDARD_IA 、ONEZONE_IA 、またはストレージクラスに移行する期間を指定します。このアクションが有効な場合、Amazon S3 の動作は、バケットのバージョニングが有効になっているかどうかによって異なります。</p> <ul style="list-style-type: none"><li>• バケットでバージョニングが有効になっていない場合、Amazon S3 はオブジェクトの唯一のコピーを指定されたストレージクラスに移行します。</li><li>• バケットでバージョニングが有効になっている (またはバージョニングが停止されている) 場合、Amazon S3 はルールで識別されたオブジェクトの現在のバージョンのみを移行します。</li></ul> <div><p> <b>Note</b></p><p>バージョニングが有効なバケットでは、オブジェクトの複数のバージョンを保持できます。このアクションは、オブジェクトの非現行バージョンには影響しません。非現行オブジェクトを移行するには、NoncurrentVersionTransition アクションを使用する必要があります。</p></div> | はい (Rule に他のアクションが存在しない場合) |

| 名前 | 説明                                | 必須 |
|----|-----------------------------------|----|
|    | 型: コンテナ                           |    |
|    | 子: Days または Date、および StorageClass |    |
|    | 祖先: Rule                          |    |

## レスポンス

### レスポンスヘッダー

このオペレーションの実装はほとんどのレスポンスに共通のレスポンスヘッダーのみを使用します。詳細については、「[共通のレスポンスヘッダー](#)」を参照してください。

### レスポンス要素

このオペレーションの実装では、レスポンス要素は返されません。

### 特殊なエラー

このオペレーションの実装では、特殊なエラーは返されません。Amazon S3 のエラーとエラーコードのリストの一般的な情報については、「[エラーレスポンス](#)」を参照してください。

## 例

### 例 1: バージョニングが有効になっていないバケットにライフサイクル設定を追加する

以下のライフサイクル設定では、それぞれ 1 つのアクションを持つ 2 つのルールが指定されています。

- Transition アクションは、作成から 30 日後に「documents/」プレフィックスを持つオブジェクトをストレージクラスに移行するように Amazon S3 に指示します。
- Expiration アクションは、作成から 365 日後に「logs/」プレフィックスを持つオブジェクトを削除するように Amazon S3 に指示します。

```
<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
```

```
<Prefix>documents/</Prefix>
<Status>Enabled</Status>
<Transition>
  <Days>30</Days>
  <StorageClass>GLACIER</StorageClass>
</Transition>
</Rule>
<Rule>
  <ID>id2</ID>
  <Prefix>logs/</Prefix>
  <Status>Enabled</Status>
  <Expiration>
    <Days>365</Days>
  </Expiration>
</Rule>
</LifecycleConfiguration>
```

以下は、前述のライフサイクル設定を examplebucket バケットに追加する PUT `/?lifecycle` リクエストの例です。

```
PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:11:21 GMT
Content-MD5: q6yJDlIkBaGGfb3QLY69A==
Authorization: authorization string
Content-Length: 415

<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>id2</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
```

```
</Expiration>
</Rule>
</LifecycleConfiguration>
```

レスポンスの例を次に示します。

```
HTTP/1.1 200 OK
x-amz-id-2: r+qR7+nhXtJDDIJ0JJYcd+1j5nM/rUFiiiZ/fNbD0sd3JUE8NWMLNHXmvPfwMpdC
x-amz-request-id: 9E26D08072A8EF9E
Date: Wed, 14 May 2014 02:11:22 GMT
Content-Length: 0
Server: AmazonS3
```

## 例 2: バージョニングが有効になっているバケットにライフサイクル設定を追加する

以下のライフサイクル設定では、それぞれ Amazon S3 が実行するアクションを 1 つずつ含む 2 つのルールが指定されています。これらのアクションは、バケットのバージョニングが有効になっている場合、またはバージョニングが停止されている場合に指定します。

- NoncurrentVersionExpiration アクションは、オブジェクトが非現行になってから 100 日後に、「logs/」プレフィックスが付いたオブジェクトの非現行バージョンを期限切れにするように Amazon S3 に指示します。
- NoncurrentVersionTransition アクションは、オブジェクトが非現行になってから 30 日後に、「documents/」プレフィックスが付いたオブジェクトの非現行バージョンをストレージクラスに移行するように Amazon S3 に指示します。

```
<LifecycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>100</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
```

```
<NoncurrentVersionTransition>
  <NoncurrentDays>30</NoncurrentDays>
  <StorageClass>GLACIER</StorageClass>
</NoncurrentVersionTransition>
</Rule>
</LifecycleConfiguration>
```

以下は、前述のライフサイクル設定を examplebucket バケットに追加する PUT `/?lifecycle` リクエストの例です。

```
PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:21:48 GMT
Content-MD5: 96rxH9mDqVnKkaZDddgnw==
Authorization: authorization string
Content-Length: 598

<LifecycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>1</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionSoonAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>0</NoncurrentDays>
      <StorageClass>GLACIER</StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
</LifecycleConfiguration>
```

レスポンスの例を次に示します。

```
HTTP/1.1 200 OK
x-amz-id-2: aXQ+KbIrmMmo0//3bMdDTw/CnjArwje+J49Hf+j44yRb/VmbIkgI05A+PT98Cp/6k07hf
+LD2mY=
x-amz-request-id: 02D7EC4C10381EB1
```

```
Date: Wed, 14 May 2014 02:21:50 GMT
Content-Length: 0
Server: AmazonS3
```

## その他の例

STANDARD\_IA や ONEZONE\_IA などのストレージクラスにオブジェクトを移行するその他の例については、「[ライフサイクル設定の例](#)」を参照してください。

## 関連リソース

- [GetBucketLifecycleConfiguration](#)
- [POST Object restore](#)
- デフォルトでは、リソース所有者 (この場合、バケットを作成した AWS アカウント であるバケット所有者) は、すべてのオペレーションを実行できます。リソース所有者は、オペレーションを実行するアクセス許可を他のユーザーに付与することもできます。詳細については、Amazon Simple Storage Service コンソールユーザーガイドの以下のトピックを参照してください。
  - [ポリシーでのアクセス許可の指定](#)
  - [Managing Access Permissions to Your Amazon S3 Resources \(Amazon S3 リソースへのアクセス許可の管理\)](#)



# GET バケットライフサイクル (廃止)

## 説明

### Important

この API の更新バージョンについては、「[GetBucketLifecycleConfiguration](#)」を参照してください。<filter> 要素を使用してバケットライフサイクルを設定した場合は、このトピックの更新バージョンが表示されます。このトピックは、下位互換性のために用意されています。

バケットに設定されている lifecycle 設定情報を返します。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[オブジェクトのライフサイクルの管理](#)」を参照してください。

このオペレーションを使用するには、s3:GetLifecycleConfiguration アクションを実行するためのアクセス許可が必要です。デフォルトでは、バケット所有者がこのアクセス許可を持っています。バケット所有者は、他のユーザーにこのアクセス許可を付与することもできます。アクセス許可の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「[Amazon S3 リソースへのアクセス許可の管理](#)」を参照してください。

## リクエスト

### 構文

```
GET /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see ##### \(AWS ##### 4\))
```

### リクエストパラメータ

オペレーションの実装では、リクエストパラメータを使用しません。

### リクエストヘッダー

このオペレーションの実装では、すべてのオペレーションに共通のリクエストヘッダーのみを使用します。詳細については、「[共通のリクエストヘッダー](#)」を参照してください。

## リクエストの要素

このオペレーションの実装では、リクエストパラメータは使用しません。

## レスポンス

### レスポンスヘッダー

このオペレーションの実装はほとんどのレスポンスに共通のレスポンスヘッダーのみを使用します。詳細については、「[共通のレスポンスヘッダー](#)」を参照してください。

### レスポンス要素

この GET の実装は、次のレスポンス要素を返します。

| 名前                             | 説明                                                                                                                                                                                                                                                   | 必須                                          |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| AbortIncompleteMultipartUpload | <p>不完全なマルチパートアップロードが中止オペレーションの対象となるタイミングを指定するためのコンテナ。</p> <p>子: DaysAfterInitiation</p> <p>型: コンテナ</p> <p>祖先: Rule</p>                                                                                                                               | はい (ルールに他のアクションが指定されていない場合)                 |
| Date                           | <p>Amazon S3 がアクションを実行する日付。詳細については、「Amazon Simple Storage Service ユーザーガイド」中の「<a href="#">ライフサイクルルール: 特定の日付に基づく</a>」を参照してください。</p> <p>日付の値は、ISO 8601 形式に従う必要があります。この時刻は常に午前 0 時 (UTC) となります。</p> <p>タイプ: 文字列</p> <p>祖先: Expiration または Transition</p> | はい (Days と ExpiredObjectDeleteMarker がない場合) |

| 名前                  | 説明                                                                                                                                                                                                               | 必須                                          |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| Days                | <p>オブジェクトの作成後、特定のルールアクションが有効になるまでの日数を指定します。オブジェクトの適格時間は、作成時刻に日数を加算し、その計算結果の時刻を翌日の UTC 午前 0 時に切り上げた値として計算されます。</p> <p>型: Transition で使用する場合は非負整数、Expiration で使用する場合は正整数。</p> <p>祖先: Transition または Expiration</p> | はい (Date と ExpiredObjectDeleteMarker がない場合) |
| DaysAfterInitiation | <p>マルチパートアップロードを開始してから、そのマルチパートアップロードを完了する必要がある日数を指定します。指定した日数で完了しない場合、中止オペレーションの対象となり、Amazon S3 は不完全なマルチパートアップロードをキャンセルします。</p> <p>型: 正の整数</p> <p>祖先: AbortIncompleteMultipartUpload</p>                        | はい (Date がない場合)                             |

| 名前         | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 必須                 |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| Expiration | <p>このアクションは、オブジェクトの有効期間内で、Amazon S3 が適切な失効アクションを実行する期間を指定します。失効アクションは、子 Date または Days 要素で指定された期間に従って対象となるオブジェクトに対してのみ発生します。Amazon S3 が実行するアクションは、バケットでバージョニングが有効になっているかどうかによって異なります。</p> <ul style="list-style-type: none"><li>• バケットでバージョニングが有効になっていない場合、Amazon S3 はオブジェクトの唯一のコピーを完全に削除します。</li><li>• それ以外の場合、バージョニングが有効になっているバケット (またはバージョニングが停止されている) の場合、アクションは現在のバージョンのオブジェクトにのみ適用されます。バージョニングが有効またはバージョニングが停止しているバケットの場合、同じオブジェクトの複数のバージョン (現行バージョン 1 つと 非現行バージョン 0 個以上) を保持できます。</li></ul> <p>Amazon S3 は、現在のバージョンを削除する代わりに、削除マーカーを新しい現在のバージョンとして追加することで、そのバージョンを非現行バージョンとします。</p> <div> <b>Important</b><br/>バケットがバージョニング一時停止状態の場合、Amazon S3 はバージョン ID として null を使用する削除</div> | はい (親タグが指定されている場合) |

| 名前 | 説明                                                                                                                                                                                                                                                                                                                                          | 必須  |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|    | <div>マーカーを作成します。バージョン ID が null のバージョンがある場合、Amazon S3 はそのバージョンを上書きします。</div> <div><div> Note</div><div>非現行オブジェクトの有効期限を設定するには、NoncurrentVersionExpiration アクションを使用する必要があります。</div></div> <div>型: コンテナ</div> <div>子: Days または Date</div> <div>祖先: Rule</div> |     |
| ID | <div>ルール用の一意の識別子。値は 255 文字以下にする必要があります。</div> <div>タイプ: 文字列</div> <div>祖先: Rule</div>                                                                                                                                                                                                                                                       | いいえ |

| 名前                        | 説明                                                                                                                                                                                                                                                                                                                                                                         | 必須                     |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| LifecycleConfiguration    | <p>ライフサイクルルールのコンテナ。最大 1,000 個のルールを追加できます。</p> <p>型: コンテナ</p> <p>子: Rule</p> <p>祖先: なし</p>                                                                                                                                                                                                                                                                                  | はい                     |
| ExpiredObjectDeleteMarker | <p>バージョン管理されたバケット (バージョニングが有効なバケットまたはバージョニングが停止されたバケット) では、この要素は、Amazon S3 がバケット内の期限切れのオブジェクト削除マーカを削除するかどうかを示します。例については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">例 8: 期限切れのオブジェクト削除マーカを削除するための失効アクションを指定する</a>」を参照してください。</p> <p>タイプ: 文字列</p> <p>有効な値: true   false (値 false は許可されますが、no-op です。値が false の場合、Amazon S3 はアクションを実行しません)</p> <p>祖先: Expiration</p> | はい (Date と Days がない場合) |

| 名前                          | 説明                                                                                                                                                                                                                                                                                                                                        | 必須                         |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| NoncurrentDays              | <p>Amazon S3 が関連付けられたアクションを実行するまでにオブジェクトが非現行状態である日数を指定します。非現行日数の計算については、「Amazon Simple Storage Service ユーザーガイド」の「<a href="#">日数に基づくライフサイクルルール</a>」を参照してください。</p> <p>型: NoncurrentVersionTransition</p> <p>で使用する場合は非負整数、NoncurrentVersionExpiration で使用する場合は正整数</p> <p>祖先: NoncurrentVersionExpiration または NoncurrentVersionTransition</p> | はい (祖先が存在する場合のみ)           |
| NoncurrentVersionExpiration | <p>最新ではないオブジェクトのバージョンが期限切れになる時期を指定します。期限切れになると、Amazon S3 により、最新ではないオブジェクトのバージョンが完全に削除されます。</p> <p>バージョンングが有効 (または停止) になっているバケットにこのライフサイクル設定アクションを設定し、オブジェクトのライフサイクル内の特定の期間にオブジェクトの非現行バージョンを Amazon S3 が削除するようにリクエストします。</p> <p>型: コンテナ</p> <p>子: NoncurrentDays</p> <p>祖先: Rule</p>                                                       | はい (Rule に他のアクションが存在しない場合) |

| 名前                          | 説明                                                                                                                                                                                                                                                                                     | 必須                         |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| NoncurrentVersionTransition | <p>非現行オブジェクトが、STANDARD_IA、ONEZONE_IA、またはストレージクラスに移行する時期を記述する移行ルールのコンテナ。</p> <p>バケットでバージョニングが有効な場合 (またはバージョニングが停止している場合)、このアクションを設定して、オブジェクトの有効期間内の特定の期間に Amazon S3 がオブジェクトの非現行バージョンをストレージクラスに移行するように要求できます。</p> <p>型: コンテナ</p> <p>子: NoncurrentDays と StorageClass</p> <p>祖先: Rule</p> | はい (Rule に他のアクションが存在しない場合) |
| Prefix                      | <p>ルールの適用対象となるオブジェクトを 1 つ以上特定するオブジェクトキープレフィックス。</p> <p>タイプ: 文字列</p> <p>祖先: Rule</p>                                                                                                                                                                                                    | はい                         |
| Rule                        | <p>ライフサイクルルールのコンテナ。</p> <p>型: コンテナ</p> <p>祖先: LifecycleConfiguration</p>                                                                                                                                                                                                               | はい                         |



| 名前           | 説明                                                                                                                                                         | 必須 |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Status       | <p>Enabled にすると、Amazon S3 はスケジュールどおりにルールを実行します。Disabled にすると、Amazon S3 はルールを無視します。</p> <p>タイプ: 文字列</p> <p>祖先: Rule</p> <p>有効な値: Enabled または Disabled</p>   | はい |
| StorageClass | <p>オブジェクトの移行先となる Amazon S3 ストレージクラスを指定します。</p> <p>タイプ: 文字列</p> <p>祖先: Transition と NoncurrentVersionTransition</p> <p>有効な値: STANDARD_IA   ONEZONE_IA  </p> | はい |

| 名前         | 説明                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 必須                         |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| Transition | <p>このアクションは、オブジェクトの有効期間内で Amazon S3 がオブジェクトを STANDARD_IA 、ONEZONE_IA 、またはストレージクラスに移行する期間を指定します。このアクションが有効な場合、Amazon S3 の動作は、バケットのバージョニングが有効になっているかどうかによって異なります。</p> <ul style="list-style-type: none"><li>• バケットでバージョニングが有効になっていない場合、Amazon S3 はオブジェクトの唯一のコピーを指定されたストレージクラスに移行します。</li><li>• バケットのバージョニングが有効になっている (またはバージョニングが停止されている) 場合、Amazon S3 はルールで識別されたオブジェクトの現在のバージョンのみを移行します。</li></ul> <div><p> <b>Note</b></p><p>バージョニングが有効なバケットまたはバージョニングが停止されたバケットには、オブジェクトの多くのバージョンを含めることができます。このアクションは、オブジェクトの非現行バージョンには影響しません。非現行オブジェクトを移行するには、NoncurrentVersionTransition アクションを使用する必要があります。</p></div> | はい (Rule に他のアクションが存在しない場合) |

| 名前 | 説明                                                       | 必須 |
|----|----------------------------------------------------------|----|
|    | 型: コンテナ<br>子: Days または Date、および StorageClass<br>祖先: Rule |    |

## 特殊なエラー

| エラーコード                       | 説明                | HTTP ステータスコード | SOAP フォルトコードプレフィックス |
|------------------------------|-------------------|---------------|---------------------|
| NoSuchLifecycleConfiguration | ライフサイクル設定が存在しません。 | 404 Not Found | クライアント              |

Amazon S3 のエラーとエラーコードのリストの一般的な情報については、「[エラーレスポンス](#)」を参照してください。

## 例

### 例 1: ライフサイクルサブリソースを取得する

この例では、指定されたバケットから lifecycle サブリソースを取得する GET リクエストと、返されたライフサイクル設定を使用したレスポンスの例を示します。

### リクエスト例

```
GET /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Thu, 15 Nov 2012 00:17:21 GMT
Authorization: signatureValue
```

### レスポンス例

```
HTTP/1.1 200 OK
```

```
x-amz-id-2: ITnGT1y4RyTmXa3rPi4hk1TXouTf0hccUjo0iCPjz6FnfIutBj3M7fPG1W02SEWp
x-amz-request-id: 51991C342C575321
Date: Thu, 15 Nov 2012 00:17:23 GMT
Server: AmazonS3
Content-Length: 358
```

```
<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Rule>
    <ID>Archive and then delete rule</ID>
    <Prefix>projectdocs</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

## 関連リソース

- [PutBucketLifecycleConfiguration](#)
- [DeleteBucketLifecycle](#)

# ドキュメント履歴