デベロッパーガイド

Amazon S3 Glacier

API バージョン 2012-06-01

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon S3 Glacier: デベロッパーガイド

Copyright © 2025 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 が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

..... x
Amazon S3 Glacier とは 1
現在 S3 Glacier を使用していますか? 1
データモデル 3
ボールト 3
アーカイブ 4
ジョブ 5
通知設定 6
サポートされているオペレーション 6
ボールトオペレーション 7
アーカイブオペレーション 7
ジョブ 7
S3 Glacier へのアクセス 7
のリージョンとエンドポイント 8
開始方法 9
ステップ 1: 開始する前に 10
のセットアップ AWS アカウント 10
適切な AWS SDK をダウンロードする 12
ステップ 2: ボールトを作成する 13
ステップ 3: ボールトにアーカイブをアップロードする 14
Java を使用してアーカイブをアップロードする 15
.NET を使用してアーカイブをアップロードする 21
ステップ 4: ボールトからアーカイブをダウンロードする 22
Java を使用してアーカイブをダウンロードする 24
.NET を使用してアーカイブをダウンロードする 25
ステップ 5: ボールトからアーカイブを削除する 27
関連するセクション 28
Java を使用してアーカイブを削除する 28
.NET を用いてアーカイブを削除する 30
を使用したアーカイブの削除 AWS CLI 31
ステップ 6: ボールトを削除する 34
ここからどこへ進むべきですか? 35
ボールトに関する各種操作 36
S3 Glacier でのボールトオペレーション 37

ボールトの作成と削除 ... 37
ボールトメタデータの取得 ... 37
ボールトインベントリのダウンロード ... 37
ボールト通知の設定 ... 38
ボールトの作成 ... 39
Java を使用してボールトを作成する ... 39
.NET を使用したボールトの作成 ... 43
REST を使用したボールトの作成 ... 47
コンソールを用いたボールトの作成 ... 47
を使用したボールトの作成 AWS CLI ... 48
ボールトメタデータの取得 ... 49
Java によるボールトメタデータの取得 ... 50
.NET を使用してボールトメタデータを取得する ... 52
REST を使用したボールトメタデータの取得 ... 55
を使用したボールトメタデータの取得 AWS CLI ... 55
ボールトインベントリのダウンロード ... 56
インベントリについて ... 58
Java を使用してボールトインベントリをダウンロードする ... 59
.NET を使用してボールトインベントリをダウンロードする ... 66
REST を使用してボールトインベントリをダウンロードする ... 74
を使用したボールトインベントリのダウンロード AWS CLI ... 74
ボールト通知の設定 ... 77
一般的な概念 ... 78
Java を使用してボールト通知を設定する ... 80
.NET を使用してボールト通知を設定する ... 83
REST API を使用してボールト通知を設定する ... 86
コンソールを使用したボールト通知の設定 ... 86
CLI を使用したボールト通知の設定 ... 89
ボールトを削除する ... 90
Java を使用してボールトを削除する ... 91
.NET を使用してボールトを削除する ... 92
REST を使用してボールトを削除する ... 94
コンソールを使用した空のボールトの削除 ... 94
を使用したボールトの削除 AWS CLI ... 95
ボールトのタグ付け ... 98
Amazon S3 Glacier コンソールを使用してボールトにタグを付けます。 ... 99

を使用したボールトのタグ付け AWS CLI 101
Amazon S3 Glacier API を使用したボールトへのタグ付け 101
関連するセクション 102
Vault Lock 102
ボールトロックの概要 102
API を使用したボールトのロック 104
CLI を使用したボールトのロック 105
コンソールを使用したボールトのロック 107
アーカイブを使用する 109
アーカイブオペレーション 110
アーカイブのアップロード 110
アーカイブの検索 110
アーカイブのダウンロード 110
アーカイブの削除 111
アーカイブの更新 111
クライアント側でのアーカイブのメタデータの管理 111
アーカイブのアップロード 112
アーカイブをアップロードするためのオプション 112
1 回のオペレーションでアーカイブをアップロードする 113
パート単位での大きなアーカイブのアップロード 124
アーカイブのダウンロード 142
アーカイブの取得 143
Java を使用してアーカイブをダウンロードする 147
.NET を使用してアーカイブをダウンロードする 164
Python を使用した大規模なアーカイブのダウンロード 181
REST を使用したアーカイブのダウンロード 189
を使用したアーカイブのダウンロード AWS CLI 189
アーカイブの削除 192
Java を使用してアーカイブを削除する 194
.NET を使用してアーカイブを削除する 196
REST を使用したアーカイブの削除 199
を使用したアーカイブの削除 AWS CLI 199
AWS SDKsの使用 203
AWS Java および .NET 用の SDK ライブラリ 203
低レベル API とは 203
高レベル API とは 204

高レベル API と低レベル API を使用する場合 ... 204
AWS SDKs の使用 ... 205
の使用 AWS SDK for Java ... 206
低レベル API の使用 ... 206
高レベル API の使用 ... 207
Eclipse を使用した Java 実行例 ... 208
エンドポイントの設定 ... 209
の使用 AWS SDK for .NET ... 209
低レベル API の使用 ... 210
高レベル API の使用 ... 211
.NET サンプルの実行 ... 212
エンドポイントの設定 ... 212
コードの例 ... 214
基本 ... 216
Hello Amazon S3 Glacier ... 217
アクション ... 218
シナリオ ... 285
ファイルのアーカイブ、通知の取得、ジョブの開始 ... 285
アーカイブコンテンツの取得とアーカイブの削除 ... 292
セキュリティ ... 297
データ保護 ... 298
データ暗号化 ... 298
キーの管理 ... 299
インターネットトラフィックのプライバシー ... 299
ID とアクセス管理 ... 300
対象者 ... 300
アイデンティティを使用した認証 ... 301
ポリシーを使用したアクセスの管理 ... 304
Amazon S3 Glacier と IAM が連携する方法 ... 307
アイデンティティベースのポリシーの例 ... 315
リソースベースのポリシーの例 ... 323
トラブルシューティング ... 328
Amazon S3 Glacier API アクセス許可のリファレンス ... 330
ログ記録とモニタリング ... 338
コンプライアンス検証 ... 339
耐障害性 ... 341

インフラストラクチャセキュリティ 342
VPC エンドポイント 342
データ取り出しポリシー 343
S3 Glacier データ取り出しポリシーの選択 343
[Free Tier Only] ポリシー 344
[Max Retrieval Rate] ポリシー 345
[No Retrieval Limit] ポリシー 345
S3 Glacier コンソールを使用したデータ取り出しポリシーの設定 345
Amazon S3 Glacier API を使用したデータ取り出しポリシーの設定 346
Amazon S3 Glacier REST API を使用したデータ取り出しポリシーの設定 346
AWS SDKs を使用したデータ取得ポリシーの設定 346
リソースのタグ付け 347
タグ付けの基本 347
タグの制限 348
タグ付けを使用したコストの追跡 348
タグ付けによるアクセス制御の管理 348
関連するセクション 349
を使用した監査ログ記録 AWS CloudTrail 350
CloudTrail 内の Amazon S3 Glacier 情報 350
Amazon S3 Glacier ログファイルエントリの概要 351
API リファレンス 355
一般的なリクエストヘッダー 356
共通のレスポンスヘッダー 359
リクエストへの署名 360
署名の計算例 361
ストリーミングオペレーションの署名の計算 363
チェックサムの計算 365
木構造ハッシュの例 1: 単一のリクエストでのアーカイブのアップロード 367
木構造ハッシュの例 2: マルチパートアップロードを使用したアーカイブのアップロード 367
ファイルの木構造ハッシュの計算 368
データをダウンロードするときのチェックサムの受信 378
エラーレスポンス 380
例 1: 存在しないジョブ ID を使用したジョブリクエストに関する説明 383
例 2: リクエストパラメータに無効な値が設定されたジョブリクエストの一覧表示 385
ボールトオペレーション 385
ボールトロックの中止 386

ボールトにタグを追加する 389
ボールトの作成 393
ボールトロックの完了 396
ボールトの削除 399
ボールトアクセスポリシーの削除 402
ボールト通知の削除 404
ボールトの説明 407
ボールトアクセスポリシーの取得 411
ボールトロックの取得 414
ボールト通知の取得 419
ボールトロックの開始 422
ボールトのタグの一覧表示 427
ボールトのリスト 430
ボールトからタグを削除する 437
ボールトアクセスポリシーの設定 440
ボールトの通知設定の指定 443
アーカイブオペレーション 447
アーカイブの削除 448
アーカイブのアップロード 450
マルチパートアップロードオペレーション 456
マルチパートアップロードの中止 456
Complete Multipart Upload 459
Initiate Multipart Upload 464
パートのリスト 469
マルチパートアップロードのリスト 476
Upload Part 483
ジョブのオペレーション 490
ジョブの説明 490
ジョブの出力の取得 501
ジョブの開始 512
ジョブのリスト表示 523
ジョブオペレーションで使用されるデータ型 533
CSVInput 533
CSVOutput 535
Encryption 536
GlacierJobDescription 537

権限 541
被付与者 541
InputSerialization 542
InventoryRetrievalJobInput 543
jobParameters 544
OutputLocation 547
OutputSerialization 548
S3Location 548
SelectParameters 550
データ取り出しオペレーション 551
データ取り出しポリシーの取得 551
プロビジョニングされた容量の表示 555
プロビジョニングされた容量の購入 559
データ取り出しポリシーの設定 562
ドキュメント履歴 568
以前の更新 569
AWS 用語集 572

このページは、2012 年にリリースされた当初のボールトと REST API を使用する、S3 Glacier サービスの既存のお客様を対象としたものです。

アーカイブストレージソリューションをお探しの場合は、Amazon S3 の S3 Glacier ストレージクラス (S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive) を使用することをお勧めします。これらのストレージオプションの詳細については、「Amazon S3 ユーザーガイド」の「S3 Glacier ストレージクラス」および「長期データストレージとしての S3 Glacier ストレージクラスを理解する」を参照してください。これらのストレージクラスは Amazon S3 API を使用し、すべてのリージョンで利用可能で、Amazon S3 コンソール内で管理できます。提供される機能には、ストレージコスト分析、ストレージレンズ、高度なオプションの暗号化機能などがあります。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。

Amazon S3 Glacier とは

Amazon S3 Glacier (S3 Glacier) サービスを現在使用していて、さらに詳しく知りたい場合は、このガイドで必要な情報が見つかります。S3 Glacier は、ボールトを使用してデータのアーカイブと長期バックアップを行うための、安全性と耐久性に優れた低コストのサービスです。S3 Glacier のサービス料金の詳細については、「S3 Glacier の料金」を参照してください。

トピック

- 現在 S3 Glacier を使用していますか?
- Amazon S3 Glacier データモデル
- S3 Glacier でサポートされるオペレーション
- Amazon S3 Glacier へのアクセス

現在 S3 Glacier を使用していますか?

> **Note**
>
> このセクションでは、S3 Glacier のサービスについて説明します。現在 S3 Glacier ストレージクラス (S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive) を使用している場合は、「Amazon S3 ユーザーガイド」の「ほとんどアクセスすることのないオブジェクトのストレージクラス」を参照してください。

現在 S3 Glacier のサービスを使用していて、さらに詳しく知りたい場合は、初めに以下のセクションを読むことをお勧めします。

- Amazon S3 Glacier とは - このセクションの以降の部分では、基盤となるデータモデル、サポートしているオペレーション、サービスとの連携に利用できる AWS SDK について説明します。
- 開始方法 -「Amazon S3 Glacier の開始方法」セクションでは、ボールトの作成、アーカイブのアップロード、アーカイブのダウンロードジョブの作成、ジョブ出力の取得、アーカイブの削除のプロセスについて順を追って説明します。

> **Important**
>
> S3 Glacier は、コンソールも提供します。ただし、アップロード、ダウンロード、削除などのアーカイブオペレーションでは、 AWS Command Line Interface （ AWS CLI) を使用するか、コードを記述する必要があります。アーカイブオペレーションについては、コンソールによるサポートはありません。例えば、写真、動画、その他のドキュメントなどのデータをアップロードするには、 を使用するか、REST API を直接使用するか、 AWS SDKs を使用して、コードを AWS CLI 記述してリクエストを行う必要があります。をインストールするには AWS CLI、「」を参照してくださいAWS Command Line Interface。 AWS CLIでの S3 Glacier の使用の詳細については、「S3 GlacierのAWS CLI リファレンス」を参照してください。を使用してアーカイブを S3 Glacier AWS CLI にアップロードする例については、「 での S3 Glacier AWS Command Line Interface の使用」を参照してください。

「ご利用開始にあたって」セクションを読み終えたら、S3 Glacier のオペレーションについてさらに詳しく学習することをお勧めします。以下のセクションでは、REST API と AWS SDKs for Java および Microsoft .NET を使用した S3 Glacier の操作に関する詳細情報を提供します。

- Amazon S3 Glacier での AWS SDKs の使用 Amazon S3

  このセクションでは、このガイドのさまざまなコード例で使用される AWS SDKs の概要を説明します。このセクションの内容をよく確認しておくと、以降のセクションの理解に役立ちます。これらの SDK が提供する高レベルまたは低レベルの API の概要、使用する状況、このガイドで提供されるコード例を実行するための一般的な手順について記載します。

- Amazon S3 Glacier のボールトの操作

  このセクションでは、ボールトの作成、ボールトメタデータの取得、ボールトインベントリを取得するジョブの使用、ボールト通知の設定など、各種のボールトオペレーションの詳細を示します。S3 Glacier コンソールの使用に加えて、さまざまなボールトオペレーションに AWS SDKs を使用できます。このセクションでは、 API について説明し、 AWS SDK for Java と を使用して作業サンプルを提供します AWS SDK for .NET。

- Amazon S3 Glacier でのアーカイブの操作

  このセクションでは、単一のリクエストでアーカイブをアップロードしたり、マルチパートアップロードオペレーションを使用してパート単位で大きなアーカイブをアップロードしたりするなどのアーカイブオペレーションについて、その詳細を説明します。また、アーカイブを非同期的にダウ

ンロードするジョブの作成方法についても解説します。このセクションでは、 AWS SDK for Java と AWS SDK for .NETを使用した例を示します。

- Amazon S3 Glacier の API リファレンス

  S3 Glacier は RESTful サービスです。このセクションでは、REST オペレーションについて説明し、すべての REST オペレーションに関する構文と、リクエストおよびレスポンスの例を示します。 AWS SDK ライブラリはこの API をラップし、プログラミングタスクを簡素化します。

Amazon S3 Glacier データモデル

Amazon S3 Glacier データモデルの主要コンポーネントには、ボールトとアーカイブが含まれます。S3 Glacier は REST ベースのウェブサービスです。REST の観点からすると、ボールトとアーカイブはリソースです。さらに S3 Glacier データモデルには、ジョブおよび通知設定リソースも含まれます。これらのリソースは主要なリソースを補完します。

トピック

- ボールト
- アーカイブ
- ジョブ
- 通知設定

ボールト

S3 Glacier では、ボールトはアーカイブを格納するコンテナです。ボールトは Amazon S3 バケットに似ています。ボールトを作成するときは、名前を指定し、ボールト AWS リージョン を作成するを選択します。

各ボールトリソースは一意のアドレスを持ちます。一般的な形式は次のとおりです。

```
https://region-specific-endpoint/account-id/vaults/vault-name
```

例えば、米国西部 (オレゴン) リージョンで ID 111122223333 のアカウントにボールト (`examplevault`) を作成するとします。次の URI を使用してこのボールトに対応できます。

```
https://glacier.us-west-2.amazonaws.com/111122223333/vaults/examplevault
```

URI のさまざまなコンポーネントの意味は次のとおりです。

- `glacier.us-west-2.amazonaws.com`は、米国西部 (オレゴン) リージョンを識別します。
- `111122223333` はボールトを所有する AWS アカウント ID です。
- `vaults` は、 AWS アカウントによって所有されているボールトの集合のことです。
- `examplevault` は、ボールトの集合に含まれる特定のボールトを識別します。

AWS アカウント は、サポートされている任意の でボールトを作成できます AWS リージョン。サポートされている のリストについては AWS リージョン、「」を参照してくださいAmazon S3 Glacier へのアクセス。単一のリージョン内では、アカウントは一意のボールト名を使用する必要があります。 AWS アカウント は、異なるリージョンに同じ名前のボールトを作成できます。

ボールトに格納できるアーカイブの数に制限はありません。ビジネスまたはアプリケーションのニーズに応じて、単一のボールトまたは複数のボールトにそれらのアーカイブを格納できます。

S3 Glacier は多様なボールトオペレーションをサポートしています。ボールトオペレーションはリージョンに固有です。たとえば、ボールトを作成する際は、特定のリージョンで作成します。ボールトリストをリクエストするときは、特定の にリクエストします。結果のリストには AWS リージョン、その特定のリージョンで作成されたボールトのみが含まれます。

アーカイブ

アーカイブは、写真、動画、ドキュメントなどのデータです。アーカイブは Amazon S3 オブジェクトに類似しており、S3 Glacier のストレージの基本単位です。各アーカイブには一意の ID とオプションの説明が割り当てられます。アーカイブのアップロード中にのみ、オプションの説明を指定できます。S3 Glacier はアーカイブに ID を割り当てます。ID はアーカイブが保存されている AWS リージョン で一意です。

各アーカイブは一意のアドレスを持ちます。全体の形式は次のとおりです。

```
https://region-specific-endpoint/account-id/vaults/vault-name/archives/archive-id
```

次の例は、アカウント 111122223333 の米国西部 (オレゴン) リージョンの `examplevault` ボールトに格納されたアーカイブの URI です。

```
https://glacier.us-west-2.amazonaws.com/111122223333/vaults/
examplevault/archives/NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
```

ボールトに格納できるアーカイブの数に制限はありません。

ジョブ

S3 Glacier ジョブでは、アーカイブを取得したり、ボールトのインベントリを取得したりできます。

S3 Glacier では、アーカイブやボールトインベントリ (アーカイブのリスト) の取得は非同期オペレーションです。まずジョブが開始され、S3 Glacier がジョブを完了した後にジョブ出力のダウンロードが実行されます。

> **Note**
>
> S3 Glacier はコールドストレージデータのアーカイブを行うソリューションを提供します。リアルタイムでのデータ取得が欠かせないストレージソリューションを必要とする用途の場合は、Amazon S3 の使用もご検討ください。詳細は、Amazon Simple Storage Service (Amazon S3)を参照してください。

ボールトインベントリのジョブを開始するには、ボールト名を提供します。アーカイブの取得ジョブには、ボールト名とアーカイブ ID の両方が必要です。ジョブの説明を追加して、ジョブを識別することも可能です。

アーカイブの取得ジョブとボールトインベントリのジョブは、ボールトに関連付けられます。いつでも単一のボールトで複数のジョブを進行させることができます。ジョブのリクエスト (ジョブの開始) を送信すると、S3 Glacier はジョブを追跡するジョブ ID を返します。各ジョブは次の形式の URI で一意に識別されます。

```
https://region-specific-endpoint/account-id/vaults/vault-name/jobs/job-id
```

以下は、アカウント 111122223333 の米国西部 (オレゴン) リージョン内の `examplevault` ボールトに関連付けられたジョブの例です。

```
https://glacier.us-west-2.amazonaws.com/111122223333/vaults/examplevault/jobs/
HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

S3 Glacier はジョブのタイプ、説明、作成日、完了日、ジョブのステータスなどの情報をジョブごとに保持します。特定のジョブに関する情報、またはいずれかのボールトに関連するすべてのジョブの

リストを取得できます。S3 Glacier が返すジョブのリストには、進行中または最近終了したすべてのジョブが含まれます。

通知設定

ジョブは実行に時間がかかるため、S3 Glacier ではジョブの完了時に通知する通知メカニズムをサポートしています。ジョブの完了時に、Amazon Simple Notification Service (Amazon SNS) トピックで通知を受け取るように、ボールトを設定できます。通知設定で、ボールトごとに 1 つの Amazon SNS トピックを指定できます。

S3 Glacier は通知設定を JSON ドキュメントとして保存します。次の例は、ボールト通知設定を示しています。

```
{
   "Topic": "arn:aws:sns:us-west-2:111122223333:mytopic",
   "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

通知設定はボールトに関連付けられており、ボールトごとに 1 件設定できます。各通知設定リソースは次の形式の URI で一意に識別されます。

```
https://region-specific-endpoint/account-id/vaults/vault-name/notification-
configuration
```

S3 Glacier は、通知設定を指定、取得、削除するオペレーションをサポートしています。通知設定を削除すると、ボールトに対するデータ取得オペレーションが完了しても、通知は送信されません。

S3 Glacier でサポートされるオペレーション

ボールトとアーカイブを使用する (「Amazon S3 Glacier データモデル」を参照) ために、Amazon S3 Glacier は一連のオペレーションをサポートしています。サポート対象の全オペレーションのうち、次のオペレーションのみが非同期です。

- アーカイブの取得
- ボールトインベントリ (アーカイブの一覧) の取得

これらのオペレーションでは、最初にジョブを開始し、次にジョブの出力をダウンロードする必要があります。以下のセクションでは、S3 Glacier オペレーションの概要を示します。

ボールトオペレーション

S3 Glacier はボールトの作成と削除を行うオペレーションを提供します。特定のボールトの説明、または AWS リージョンに存在するすべてのボールトについての説明を取得できます。ボールトの説明には、作成日、ボールト内のアーカイブ数、ボールト内の全アーカイブが使用している合計サイズ (バイト単位)、S3 Glacier によるボールトインベントリの作成日などの情報が記されています。S3 Glacier は、ボールトで通知設定を指定、取得、削除するオペレーションも提供します。詳細については、「Amazon S3 Glacier のボールトの操作」を参照してください。

アーカイブオペレーション

S3 Glacier では、アーカイブをアップロードおよび削除するオペレーションを提供します。既存のアーカイブを更新することはできず、既存のアーカイブを削除し、新しいアーカイブをアップロードする必要があります。アーカイブをアップロードするたびに、S3 Glacier が新しいアーカイブ ID を生成します。詳細については、「Amazon S3 Glacier でのアーカイブの操作」を参照してください。

ジョブ

S3 Glacier ジョブを開始して、アーカイブの取得を実行したり、ボールトのインベントリを取得したりできます。

以下は、S3 Glacier ジョブのタイプです。

- `archive-retrieval` – アーカイブを取得します。

  詳細については、「S3 Glacier でのアーカイブのダウンロード」を参照してください。
- `inventory-retrieval` – ボールトのインベントリを作成します。

  詳細については、「Amazon S3 Glacier でボールトインベントリをダウンロードする」を参照してください。

Amazon S3 Glacier へのアクセス

Amazon S3 Glacier は、トランスポートプロトコルとして HTTP と HTTPS を使用し、メッセージシリアライズ形式として JavaScript Object Notation (JSON) を使用する RESTful なウェブサービスです。アプリケーションコードから直接、S3 Glacier ウェブサービス API へのリクエストを行うことができます。この REST API を直接使用するときは、リクエストの署名と認証のためのコードを書く必要があります。API の詳細については、「Amazon S3 Glacier の API リファレンス」を参照してください。

または、S3 Glacier REST API コールをラップする AWS SDKs を使用して、アプリケーション開発を簡素化することもできます。開発者が認証情報を指定すれば、ライブラリによって認証とリクエスト署名の処理が自動的に行われます。 AWS SDK の使用方法の詳細については、「Amazon S3 Glacier での AWS SDKs の使用 Amazon S3 」を参照してください。

S3 Glacier には、コンソールも提供します。ただし、すべてのアーカイブおよびジョブオペレーションでは、REST API を直接使用するか、 AWS SDK ラッパーライブラリを使用してコードを記述し、リクエストを行う必要があります。S3 Glacier コンソールにアクセスするには、を参照してください。S3 Glacier コンソール。

のリージョンとエンドポイント

特定の にボールトを作成します AWS リージョン。S3 Glacier リクエストは、常に AWS リージョンの固有のエンドポイントに送信します。S3 Glacier でサポートされる AWS リージョン のリストについては、「AWS 全般リファレンス」の「Amazon S3 Glacier エンドポイントとクォータ」を参照してください。

Amazon S3 Glacier の開始方法

ボールトとアーカイブを操作することにより Amazon S3 Glacier (S3 Glacier) の使用を開始できます。アーカイブを格納するコンテナをボールトと言います。アーカイブとは、ボールトに格納する写真、動画、ドキュメントなどのオブジェクトを指します。アーカイブは、S3 Glacier のストレージの基本単位です。この入門演習では、ボールトおよびアーカイブリソースに関する S3 Glacier の基本的なオペレーションを、手順に従って学習していきます。これらのリソースの詳細については、「Amazon S3 Glacier データモデル セクション」を参照してください。

この入門演習では、ボールトを作成し、アーカイブのアップロードやダウンロードを行った後、アーカイブとボールトを削除します。ここに挙げたオペレーションはすべて、プログラムで実行できます。ただし、今回の入門演習では、ボールトの作成と削除には S3 Glacier マネジメントコンソールを使用します。アーカイブのアップロードとダウンロードについては、この入門セクションでは、 AWS SDK for Java と の高レベル API を使用します AWS SDK for .NET。高レベル API は、S3 Glacier でプログラミングを簡単に行えるようにするものです。 AWS SDKs「」を参照してくださいAmazon S3 Glacier での AWS SDKs の使用 Amazon S3 。

> **Important**
>
> S3 Glacier は、コンソールも提供します。ただし、アップロード、ダウンロード、削除などのアーカイブオペレーションでは、 AWS Command Line Interface (CLI) を使用するか、コードを記述する必要があります。アーカイブオペレーションについては、コンソールによるサポートはありません。たとえば、写真、動画、その他のドキュメントなどのデータをアップロードするには、REST API を直接使用する AWS CLI か、 AWS SDKs を使用して、を使用するか、コードを記述してリクエストを行う必要があります。
> をインストールするには AWS CLI、「」を参照してくださいAWS Command Line Interface。で S3 Glacier を使用する方法の詳細については AWS CLI、AWS CLI S3 Glacier のリファレンス」を参照してください。を使用してアーカイブを S3 Glacier AWS CLI にアップロードする例については、「 での S3 Glacier AWS Command Line Interface の使用」を参照してください。

この入門演習では、アーカイブのアップロードおよびダウンロードのための Java および C# のコード例を用意しています。この入門演習の最後のセクションでは、S3 Glacier を使用した開発者のエクスペリエンスについて詳しく知ることができる手順を説明します。

トピック

- ステップ 1: S3 Glacier の使用を開始する前に
- ステップ 2: S3 Glacier でボールトを作成する
- ステップ 3: S3 Glacier でボールトにアーカイブをアップロードする
- ステップ 4: S3 Glacier でボールトからアーカイブをダウンロードする
- ステップ 5: S3 Glacier でボールトからアーカイブを削除する
- ステップ 6: S3 Glacier でボールトを削除する
- ここからどこへ進むべきですか?

ステップ 1: S3 Glacier の使用を開始する前に

この演習を開始する前に、 AWS アカウント (まだ持っていない場合は) にサインアップし、いずれかの AWS SDKs をダウンロードする必要があります。詳細については、以下のセクションを参照してください。

トピック

- AWS アカウント と管理者ユーザーのセットアップ
- 適切な AWS SDK をダウンロードする

AWS アカウント と管理者ユーザーのセットアップ

まだサインアップしていない場合は、 にサインアップ AWS アカウント し、アカウントに管理者ユーザーを作成する必要があります。

セットアップを完了するには、以下のトピックの指示に従ってください。

のセットアップ AWS アカウント と管理者ユーザーの作成

にサインアップする AWS

Amazon Web Services (AWS) にサインアップすると AWS、 AWS アカウント は S3 Glacier を含むのすべてのサービスに自動的にサインアップされます。料金は、使用するサービスの料金のみが請求されます。S3 Glacier の使用料の詳細については、Amazon S3 Glacier 料金ページを参照してください。

が既にある場合は AWS アカウント、「」に進みます適切な AWS SDK をダウンロードする。がない場合は AWS アカウント、次の手順を使用して作成します。

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

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

   サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

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

管理者ユーザーを作成するには、以下のいずれかのオプションを選択します。

管理者を管理する方法を 1 つ選択します	目的	方法	以下の操作も可能
IAM Identity Center 内 (推奨)	短期の認証情報を使用して AWSにアクセスします。 これはセキュリティのベストプラクティスと一致しています。ベストプラクティスの詳細については、IAM ユーザーガイドの「IAM でのセキュリティのベストプラク	AWS IAM Identity Center ユーザーガイドの「開始方法」の手順に従います。	AWS Command Line Interface ユーザーガイドの を使用する AWS CLI ように を設定 AWS IAM Identity Centerして、プログラムによるアクセスを設定します。

管理者を管理する方法を 1 つ選択します	目的	方法	以下の操作も可能
	ティス」を参照してください。		
IAM 内 (非推奨)	長期認証情報を使用して AWSにアクセスする。	IAM ユーザーガイドの「緊急アクセス用の IAM ユーザーを作成する」の手順に従います。	IAM ユーザーガイドの「IAM ユーザーのアクセスキーを管理する」の手順に従って、プログラムによるアクセスを設定します。

適切な AWS SDK をダウンロードする

開始方法の演習を試すには、使用するプログラミング言語を決定し、開発プラットフォームに適した AWS SDK をダウンロードする必要があります。

この入門演習では、Java と C# の例を用意しています。

AWS SDK for Javaのダウンロード

この開発者ガイドの Java の例をテストするには、 AWS SDK for Javaが必要です。次のいずれかの方法でダウンロードしてください。

- Eclipse を使用している場合は、更新サイト http://aws.amazon.com/eclipse/ AWS Toolkit for Eclipse を使用して をダウンロードしてインストールできます。詳細については、「AWS Toolkit for Eclipse」を参照してください。
- 他の IDE を使用してアプリケーションを作成する場合は、AWS SDK for Java をダウンロードしてください。

AWS SDK for .NETのダウンロード

この開発者ガイドの C# のコード例をテストするには、 AWS SDK for .NETが必要です。次のいずれかの方法でダウンロードしてください。

- Visual Studio を使用している場合は、 AWS SDK for .NET と の両方をインストールできます AWS Toolkit for Visual Studio。このツールキットには、 AWS 開発に使用できる Explorer for Visual Studio とプロジェクトテンプレートが用意されています。をダウンロードするには AWS SDK for .NET、http://aws.amazon.com/sdkfornet にアクセスしてください。デフォルトでは、インストールスクリプトは AWS SDK と の両方をインストールします AWS Toolkit for Visual Studio。ツールキットの詳細については、「AWS Toolkit for Visual Studio ユーザーガイド」を参照してください。
- 他の IDE を使用してアプリケーションを作成する場合は、前のステップに示したリンクを使用して AWS SDK for .NETのみをインストールしてください。

ステップ 2: S3 Glacier でボールトを作成する

ボールトは、アーカイブを格納するコンテナです。最初のステップは、サポートされている のいずれかにボールトを作成することです AWS リージョン。Amazon S3 Glacier で AWS リージョン サポートされている のリストについては、 AWS 全般のリファレンスのAmazon S3 Glacier エンドポイントとクォータ」を参照してください。

ボールトは、プログラムまたは S3 Glacier コンソールを使用して作成できます。このセクションでは、コンソールを使用してボールトを作成します。

ボールトを作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/glacier/home で S3 Glacier コンソールを開きます。
2. 左のナビゲーションペインで、[ボールト] を選択します。
3. [ボールトを作成] を選択します。

   [ボールトを作成] ページが開きます。
4. 「リージョンの選択」で、リージョンセレクター AWS リージョン から を選択します。ボールトは選択するリージョンにあります。
5. [ボールト名] にボールトの名前を入力します。

   以下に、ボールト命名の要件を示します。

   - ボールト名は、ボールトが作成される AWS アカウント と AWS リージョン 内で一意である必要があります。
   - ボールト名の長さは 1 ~ 255 文字である必要があります。

- ボールト名には、a–z、A–Z、0–9、_ (下線)、- (ハイフン)、. (ピリオド) のみを使用できます。

6. [イベント通知] で、ジョブが完了した際のボールトへの通知をオンまたはオフにするには、以下のいずれかの設定を選択します。

   - 通知をオフにします — 通知はオフになり、指定したジョブが完了すると Amazon Simple Notification Service (Amazon SNS) トピックに通知が送信されなくなります。

   - 通知をオンにします — 通知はオンになり、指定したジョブが完了すると Amazon SNS トピックに通知が送信されます。

     [通知をオンにします] を選択した場合は、「Amazon S3 Glacier コンソールを使用したボールト通知の設定」を参照してください。

7. AWS リージョン とボールト名が正しい場合は、ボールトの作成を選択します。

新しいボールトが S3 Glacier コンソールの [ボールト] ページに表示されます。

ステップ 3: S3 Glacier でボールトにアーカイブをアップロードする

このステップでは、前のステップ (「ステップ 2: S3 Glacier でボールトを作成する」を参照) で作成したボールトにサンプルアーカイブをアップロードします。使用する開発プラットフォームに応じて、このセクションの最後にあるリンクのいずれか 1 つをクリックしてください。

> **Important**
>
> いずれのアーカイブオペレーション (アップロード、ダウンロード、削除など) にも、 AWS Command Line Interface (CLI) の使用かコードの記述が必要になります。アーカイブオペレーションについては、コンソールによるサポートはありません。たとえば、写真、動画、その他のドキュメントなどのデータをアップロードするには、REST API を直接使用する AWS CLI か、 AWS SDKs を使用して、 を使用するか、コードを記述してリクエストを行う必要があります。
> をインストールするには AWS CLI、「」を参照してくださいAWS Command Line Interface。で S3 Glacier を使用する方法の詳細については AWS CLI、AWS CLI S3 Glacier のリファレンス」を参照してください。を使用してアーカイブを S3 Glacier AWS CLI に

アップロードする例については、「 での S3 Glacier AWS Command Line Interface の使用」を参照してください。

アーカイブとは、写真、動画、ドキュメントなど、ボールトに格納するオブジェクトを指します。アーカイブは、S3 Glacier のストレージの基本単位です。1 回のリクエストでは、アーカイブを 1 つアップロードできます。アーカイブが大きい場合、S3 Glacier ではアーカイブを分割してアップロードするためのマルチパートアップロード API オペレーションが用意されています。

入門演習のこのセクションでは、1 回のリクエストでサンプルアーカイブを 1 つアップロードします。この演習で指定するファイルのサイズは、比較的小さなものです。ファイルのサイズが大きい場合には、マルチパートアップロードが適しています。詳細については、「パート単位での大きなアーカイブのアップロード (マルチパートアップロード)」を参照してください。

トピック

- を使用して S3 Glacier のボールトにアーカイブをアップロードする AWS SDK for Java
- を使用して S3 Glacier のボールトにアーカイブをアップロードする AWS SDK for .NET

を使用して S3 Glacier のボールトにアーカイブをアップロードする AWS SDK for Java

次の Java コード例では、 の高レベル API AWS SDK for Java を使用してサンプルアーカイブをボールトにアップロードします。このコード例では、以下の点に注意してください。

- この例では、`AmazonGlacierClient` クラスのインスタンスを作成します。
- この例では、 AWS SDK for Javaの高レベル API の `ArchiveTransferManager` クラスの `upload` API オペレーションを使用します。
- この例では、米国西部 (オレゴン) リージョン (`us-west-2`) を使用します。

この例を実行するための詳しい手順については、「Eclipse を使用した Amazon S3 Glacier の Java 実行例」を参照してください。ここに示したコードは、アップロードするアーカイブファイルの名前で更新する必要があります。

> **Note**
>
> Amazon S3 Glacier では、ボールト内のすべてのアーカイブのインベントリが保持されます。以下の例でアーカイブをアップロードすると、ボールトインベントリが更新されるま

で、マネジメントコンソールでそのアーカイブはボールトに表示されません。この更新は通常、1 日 1 回実行されます。

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.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * 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 UploadArchive {

    static final int ONE_MB = 1024 * 1024;

    public static void main(String[] args) {
        final String usage = """

                Usage:   <strPath> <vaultName>\s
```

```
                Where:
                   strPath - The path to the archive to upload (for example, C:\\AWS
\\test.pdf).
                   vaultName - The name of the vault.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
        glacier.close();
    }

    public static String uploadContent(GlacierClient glacier, Path path, String
 vaultName, File myFile) {
        // Get an SHA-256 tree hash value.
        String checkVal = computeSHA256(myFile);
        try {
            UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
                    .vaultName(vaultName)
                    .checksum(checkVal)
                    .build();

            UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
            return res.archiveId();

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

```
    private static String computeSHA256(File inputFile) {
        try {
            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
            return toHex(treeHash);

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s", inputFile,
 ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
 %s", nsae.getMessage());
            System.exit(-1);
        }
        return "";
    }

    public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
            NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
```

```
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;

            while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                            ioe.getMessage());
                }
            }
        }
    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     */
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[][] prevLvlHashes = chunkSHA256Hashes;
        while (prevLvlHashes.length > 1) {
            int len = prevLvlHashes.length / 2;
            if (prevLvlHashes.length % 2 != 0) {
                len++;
            }
```

```
            byte[][] currLvlHashes = new byte[len][];
            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

                // If there are at least two elements remaining.
                if (prevLvlHashes.length - i > 1) {

                    // Calculate a digest of the concatenated nodes.
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();

                } else { // Take care of the remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }

            prevLvlHashes = currLvlHashes;
        }

        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);
        for (byte datum : data) {
            String hex = Integer.toHexString(datum & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
}
```

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

を使用して S3 Glacier のボールトにアーカイブをアップロードする AWS SDK for .NET

次の C# コード例では、 の高レベル API AWS SDK for .NET を使用してサンプルアーカイブをボールトにアップロードします。このコード例では、以下の点に注意してください。

- この例では、指定された Amazon S3 Glacier リージョンのエンドポイントに対して、`ArchiveTransferManager` クラスのインスタンスを作成します。
- このコード例では、米国西部 (オレゴン) リージョン (`us-west-2`) を使用します。
- この例では、`ArchiveTransferManager` クラスの Upload API オペレーションを使用してアーカイブをアップロードしています。小さいアーカイブでは、このオペレーションによりアーカイブが直接 S3 Glacier にアップロードされます。大きなアーカイブの場合、このオペレーションでは S3 Glacier のマルチパートアップロード API オペレーションを使用し、S3 Glacier へのデータのストリーミング中にエラーが発生したときにアップロードを複数のパートに分割してエラー回復を向上させます。

以下の例を実行するための詳しい手順については、「コード例の実行」を参照してください。ここに示したコードは、ボールトの名前とアップロードするアーカイブファイルの名前で更新する必要があります。

> **Note**
>
> S3 Glacier では、ボールト内のすべてのアーカイブのインベントリが保持されます。以下の例でアーカイブをアップロードすると、ボールトインベントリが更新されるまで、マネジメントコンソールでそのアーカイブはボールトに表示されません。この更新は通常、1 日 1 回実行されます。

Example — の高レベル API を使用してアーカイブをアップロードする AWS SDK for .NET

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
```

```
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
    class ArchiveUploadHighLevel_GettingStarted
    {
        static string vaultName = "examplevault";
        static string archiveToUpload = "*** Provide file name (with full path) to
 upload ***";

        public static void Main(string[] args)
        {
            try
            {
                var manager = new
 ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
                // Upload an archive.
                string archiveId = manager.Upload(vaultName, "getting started archive
 test", archiveToUpload).ArchiveId;
                Console.WriteLine("Copy and save the following Archive ID for the next
 step.");
                Console.WriteLine("Archive ID: {0}", archiveId);
                Console.WriteLine("To continue, press Enter");
                Console.ReadKey();
            }
            catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
            catch (Exception e) { Console.WriteLine(e.Message); }
            Console.WriteLine("To continue, press Enter");
            Console.ReadKey();
        }
    }
}
```

ステップ 4: S3 Glacier でボールトからアーカイブをダウンロードする

このステップでは、「ステップ 3: S3 Glacier でボールトにアーカイブをアップロードする」でアップロードしたサンプルアーカイブをダウンロードします。

> **Important**
>
> Amazon S3 Glacier は、コンソールも提供します。ただし、アップロード、ダウンロード、削除などのアーカイブオペレーションでは、 AWS Command Line Interface (CLI) を使用するか、コードを記述する必要があります。アーカイブオペレーションについては、コンソールによるサポートはありません。たとえば、写真、動画、その他のドキュメントなどのデータをアップロードするには、REST API を直接使用する AWS CLI か、 AWS SDKs を使用して、 を使用するか、コードを記述してリクエストを行う必要があります。
> をインストールするには AWS CLI、「」を参照してくださいAWS Command Line Interface。で S3 Glacier を使用する方法の詳細については AWS CLI、AWS CLI S3 Glacier のリファレンス」を参照してください。を使用して S3 Glacier にアーカイブ AWS CLI をアップロードする例については、「 での S3 Glacier AWS Command Line Interface の使用」を参照してください。

一般に、S3 Glacier からのデータの取得は、2 段階のプロセスです。

1. 取得ジョブを開始します。
2. ジョブが完了したら、データのバイトをダウンロードします。

S3 Glacier からアーカイブを取得するには、まずジョブを開始します。ジョブが完了したら、データをダウンロードします。アーカイブの取得に関する詳細については、「S3 Glacier アーカイブの取得」を参照してください。

リクエストのアクセス時間は、迅速、標準、大容量のどの取り出しオプションを選択したかによって決まります。最大規模のアーカイブ (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アーカイブは通常 1〜5 分以内で使用可能になります。標準取り出しを使用して取り出したアーカイブは、通常 3〜5 時間で使用可能になりす。通常、大容量取り出しは 5〜12 時間で使用可能になります。さまざまな取り出しオプションの詳細については、「S3 Glacier のよくある質問」を参照してください。データ取り出し料金については、「S3 Glacier の料金」ページを参照してください。

以下のトピックで示すコード例では、ジョブを開始し、その完了まで待機したうえで、アーカイブのデータをダウンロードしています。

トピック

- を使用して S3 Glacier のボールトからアーカイブをダウンロードする AWS SDK for Java

- を使用して S3 Glacier のボールトからアーカイブをダウンロードする AWS SDK for .NET

を使用して S3 Glacier のボールトからアーカイブをダウンロードする AWS SDK for Java

次の Java コード例では、 の高レベル API を使用して AWS SDK for Java 、前のステップでアップロードしたアーカイブをダウンロードします。このコード例では、以下の点に注意してください。

- この例では、`AmazonGlacierClient` クラスのインスタンスを作成します。
- このコードでは、「ステップ 2: S3 Glacier でボールトを作成する」でボールトを作成した場所に合わせて、`us-west-2` リージョン 米国西部(オレゴン) を使用します。
- この例では、 AWS SDK for Javaの高レベル API の `ArchiveTransferManager` クラスの `download` API オペレーションを使用します。この例は、Amazon Simple Notification Service (Amazon SNS) トピックと、そのトピックにサブスクライブされている Amazon Simple Queue Service (Amazon SQS) キューを作成します。「」の指示に従って AWS Identity and Access Management (IAM) 管理者ユーザーを作成した場合ステップ 1: S3 Glacier の使用を開始する前に、ユーザーには Amazon SNS トピックと Amazon SQS キューの作成と使用に必要な IAM アクセス許可があります。

この例を実行するための詳しい手順については、「Eclipse を使用した Amazon S3 Glacier の Java 実行例」を参照してください。ここに示したコードは、「ステップ 3: S3 Glacier でボールトにアーカイブをアップロードする」でアップロードしたファイルのアーカイブ ID で更新する必要があります。

Example - AWS SDK for Javaを使用してアーカイブをダウンロードする

```
import java.io.File;
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.transfer.ArchiveTransferManager;
import com.amazonaws.services.sns.AmazonSNSClient;
import com.amazonaws.services.sqs.AmazonSQSClient;

public class AmazonGlacierDownloadArchive_GettingStarted {
    public static String vaultName = "examplevault";
    public static String archiveId = "*** provide archive ID ***";
```

```
    public static String downloadFilePath  = "*** provide location to download archive
 ***";

    public static AmazonGlacierClient glacierClient;
    public static AmazonSQSClient sqsClient;
    public static AmazonSNSClient snsClient;

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


     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        glacierClient = new AmazonGlacierClient(credentials);
        sqsClient = new AmazonSQSClient(credentials);
        snsClient = new AmazonSNSClient(credentials);

        glacierClient.setEndpoint("glacier.us-west-2.amazonaws.com");
        sqsClient.setEndpoint("sqs.us-west-2.amazonaws.com");
        snsClient.setEndpoint("sns.us-west-2.amazonaws.com");

        try {
            ArchiveTransferManager atm = new ArchiveTransferManager(glacierClient,
 sqsClient, snsClient);

            atm.download(vaultName, archiveId, new File(downloadFilePath));

        } catch (Exception e)
        {
            System.err.println(e);
        }
    }
}
```

を使用して S3 Glacier のボールトからアーカイブをダウンロードする AWS SDK for .NET

次の C# コード例では、 の高レベル API を使用して AWS SDK for .NET 、 で以前にアップロードしたアーカイブをダウンロードします[を使用して S3 Glacier のボールトにアーカイブをアップロードする AWS SDK for .NET](#)。このコード例では、以下の点に注意してください。

- この例では、指定された Amazon S3 Glacier リージョンのエンドポイントに対して、`ArchiveTransferManager` クラスのインスタンスを作成します。
- このコード例では、以前に「`us-west-2`」でボールトを作成した場所に合わせて、 リージョン 米国西部(オレゴン) を使用します。
- この例では、`ArchiveTransferManager` クラスの `Download` API オペレーションを使用してアーカイブをダウンロードしています。この例は、Amazon Simple Notification Service (Amazon SNS) トピックと、そのトピックにサブスクライブされている Amazon Simple Queue Service (Amazon SQS) キューを作成します。「」の指示に従って AWS Identity and Access Management (IAM) 管理者ユーザーを作成した場合ステップ 1: S3 Glacier の使用を開始する前に、ユーザーには Amazon SNS トピックと Amazon SQS キューの作成と使用に必要な IAM アクセス許可があります。
- その後、この例ではアーカイブの取り出しジョブを開始し、使用可能にするアーカイブを探してキューをポーリングします。アーカイブが使用可能になると、ダウンロードが開始されます。取得時間に関する詳細については、「アーカイブの取り出しオプション」を参照してください。

この例を実行するための詳しい手順については、「コード例の実行」を参照してください。ここに示したコードは、「ステップ 3: S3 Glacier でボールトにアーカイブをアップロードする」でアップロードしたファイルのアーカイブ ID で更新する必要があります。

Example — の高レベル API を使用してアーカイブをダウンロードする AWS SDK for .NET

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
    class ArchiveDownloadHighLevel_GettingStarted
    {
        static string vaultName = "examplevault";
        static string archiveId = "*** Provide archive ID ***";
        static string downloadFilePath = "*** Provide the file name and path to where
 to store the download ***";

        public static void Main(string[] args)
        {
            try
            {
```

```
                var manager = new
ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);

                var options = new DownloadOptions();
                options.StreamTransferProgress +=
ArchiveDownloadHighLevel_GettingStarted.progress;
                // Download an archive.
                Console.WriteLine("Intiating the archive retrieval job and then polling
SQS queue for the archive to be available.");
                Console.WriteLine("Once the archive is available, downloading will
begin.");
                manager.Download(vaultName, archiveId, downloadFilePath, options);
                Console.WriteLine("To continue, press Enter");
                Console.ReadKey();
            }
            catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
            catch (Exception e) { Console.WriteLine(e.Message); }
            Console.WriteLine("To continue, press Enter");
            Console.ReadKey();
        }

        static int currentPercentage = -1;
        static void progress(object sender, StreamTransferProgressArgs args)
        {
            if (args.PercentDone != currentPercentage)
            {
                currentPercentage = args.PercentDone;
                Console.WriteLine("Downloaded {0}%", args.PercentDone);
            }
        }
    }
}
```

ステップ 5: S3 Glacier でボールトからアーカイブを削除する

このステップでは、「ステップ 3: S3 Glacier でボールトにアーカイブをアップロードする」でアップロードしたサンプルアーカイブを削除します。

> **Important**
>
> Amazon S3 Glacier コンソールを使用してアーカイブを削除することはできません。アップロード、ダウンロード、削除などのアーカイブオペレーションでは、 AWS Command Line Interface (CLI) を使用するか、コードを記述する必要があります。写真、動画、その他のドキュメントなどのデータをアップロードするには、 を使用するか、REST API を直接使用するか、 AWS SDKs を使用してリクエストを行うコードを AWS CLI 記述する必要があります。
> をインストールするには AWS CLI、「」を参照してくださいAWS Command Line Interface。で S3 Glacier を使用する方法の詳細については AWS CLI、AWS CLI S3 Glacier のリファレンス」を参照してください。を使用して S3 Glacier にアーカイブ AWS CLI をアップロードする例については、「 での S3 Glacier AWS Command Line Interface の使用」を参照してください。

以下のいずれかの SDK または AWS CLIを使用して、サンプルアーカイブを削除します。

- を使用して S3 Glacier のボールトからアーカイブを削除する AWS SDK for Java
- を使用して S3 Glacier のボールトからアーカイブを削除する AWS SDK for .NET
- AWS CLIを使用した S3 Glacier でのアーカイブの削除

関連するセクション

- ステップ 3: S3 Glacier でボールトにアーカイブをアップロードする
- Amazon S3 Glacier でアーカイブを削除する

を使用して S3 Glacier のボールトからアーカイブを削除する AWS SDK for Java

次のコード例では、 AWS SDK for Java を使用してアーカイブを削除します。このコードでは、以下の点に注意してください。

- `DeleteArchiveRequest` オブジェクトには、アーカイブが存在するボールトの名前やアーカイブ ID など、削除のリクエストを説明する情報が含まれています。

- `deleteArchive` API オペレーションは、アーカイブを削除するリクエストを Amazon S3 Glacier に送信します。
- この例では、米国西部 (オレゴン) リージョン (`us-west-2`) を使用します。

この例を実行するための詳しい手順については、「Eclipse を使用した Amazon S3 Glacier の Java 実行例」を参照してください。ここに示したコードは、「ステップ 3: S3 Glacier でボールトにアーカイブをアップロードする」でアップロードしたファイルのアーカイブ ID で更新する必要があります。

Example — AWS SDK for Javaを使用したアーカイブの削除

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.DeleteArchiveRequest;

public class AmazonGlacierDeleteArchive_GettingStarted {

    public static String vaultName = "examplevault";
    public static String archiveId = "*** provide archive ID***";
    public static AmazonGlacierClient client;

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


     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-west-2.amazonaws.com/");

        try {

            // Delete the archive.
            client.deleteArchive(new DeleteArchiveRequest()
                .withVaultName(vaultName)
                .withArchiveId(archiveId));

            System.out.println("Deleted archive successfully.");

        } catch (Exception e) {
            System.err.println("Archive not deleted.");
```

```
            System.err.println(e);
        }
    }
}
```

を使用して S3 Glacier のボールトからアーカイブを削除する AWS SDK for .NET

次の C# コード例では、 の高レベル API AWS SDK for .NET を使用して、前のステップでアップロードしたアーカイブを削除します。このコード例では、以下の点に注意してください。

- この例では、指定された Amazon S3 Glacier リージョンのエンドポイントに対して、`ArchiveTransferManager` クラスのインスタンスを作成します。
- このコード例では、米国西部 (オレゴン) リージョン (`us-west-2`) を使用します。
- この例では、 AWS SDK for .NETの高レベル API の一部として提供される `ArchiveTransferManager` クラスの `Delete` API オペレーションを使用します。

この例を実行するための詳しい手順については、「コード例の実行」を参照してください。ここに示したコードは、「ステップ 3: S3 Glacier でボールトにアーカイブをアップロードする」でアップロードしたファイルのアーカイブ ID で更新する必要があります。

Example — の高レベル API を使用してアーカイブを削除する AWS SDK for .NET

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveDeleteHighLevel_GettingStarted
  {
    static string vaultName = "examplevault";
    static string archiveId = "*** Provide archive ID ***";

    public static void Main(string[] args)
    {
      try
      {
```

```
        var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
        manager.DeleteArchive(vaultName, archiveId);
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }
  }
}
```

AWS CLIを使用した S3 Glacier でのアーカイブの削除

AWS Command Line Interface () を使用して、Amazon S3 Glacier のアーカイブを削除できますAWS CLI。

トピック

- (前提条件) のセットアップ AWS CLI
- 例: を使用してアーカイブを削除する AWS CLI

(前提条件) のセットアップ AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

   のインストール AWS Command Line Interface

   の設定 AWS Command Line Interface

2. コマンドプロンプトで次のコマンドを入力して、 AWS CLI セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを使用してください。

     ```
     aws help
     ```

   - 設定したアカウントの S3 Glacier ボールトのリストを取得するには、`list-vaults` コマンドを使用します。*123456789012* 「」を AWS アカウント ID に置き換えます。

```
aws glacier list-vaults --account-id 123456789012
```

- の現在の設定データを表示するには AWS CLI、 `aws configure list` コマンドを使用します。

```
aws configure list
```

例: を使用してアーカイブを削除する AWS CLI

1. インベントリ取得ジョブを開始するには、`initiate-job` コマンドを使用します。`initiate-job` コマンドの詳細については、「ジョブの開始」を参照してください。

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
job-parameters "{\"Type\": \"inventory-retrieval\"}"
```

正常な出力:

```
{
    "location": "/111122223333/vaults/awsexamplevault/jobs/*** jobid ***",
    "jobId": "*** jobid ***"
}
```

2. 以前の取り出しジョブのステータスをチェックするには、`describe-job` コマンドを使用します。`describe-job` コマンドの詳細については、「ジョブの説明」を参照してください。

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

正常な出力:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "*** vault arn ***",
    "Completed": false,
    "JobId": "*** jobid ***",
    "Action": "InventoryRetrieval",
```

```
    "CreationDate": "*** job creation date ***",
    "StatusCode": "InProgress"
}
```

3. ジョブが完了するまで待ちます。

   ジョブの出力をダウンロードする準備が整うまで待つ必要があります。ボールトに通知設定を指定している場合、またはジョブを開始したときに Amazon Simple Notification Service (Amazon SNS) トピックを指定している場合は、ジョブの完了後に S3 Glacier からそのトピックにメッセージが送信されます。

   ボールトに特定のイベントに対する通知設定を指定できます。詳細については、「Amazon S3 Glacier でのボールト通知の設定」を参照してください。S3 Glacier は、特定のイベントが発生するたびに、指定された Amazon SNS トピックにメッセージを送信します。

4. ジョブが完了したら、`get-job-output` コマンドを使用して、取り出しジョブをファイル `output.json` にダウンロードします。`get-job-output` コマンドの詳細については、「ジョブの出力の取得」を参照してください。

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output.json
```

   このコマンドは、次のフィールドを含むファイルを生成します。

```
{
"VaultARN":"arn:aws:glacier:region:111122223333:vaults/awsexamplevault",
"InventoryDate":""*** job completion date ***"",
"ArchiveList":[{
{"ArchiveId":""*** archiveid ***"",
"ArchiveDescription":"*** archive description (if set) ***",
"CreationDate":""*** archive creation date ***"",
"Size":""*** archive size (in bytes) ***"",
"SHA256TreeHash":"*** archive hash ***"
}],
"ArchiveId": 123456789

}
```

5. `delete-archive` コマンドを使用して、ボールトから各アーカイブを削除します。

```
aws glacier delete-archive --vault-name awsexamplevault --account-id 111122223333
 --archive-id="*** archiveid ***"
```

delete-archive コマンドの詳細については、「アーカイブの削除」を参照してください。

ステップ 6: S3 Glacier でボールトを削除する

ボールトは、アーカイブを格納するコンテナです。Amazon S3 Glacier ボールトを削除するには、まず、S3 Glacier によって計算された最後のインベントリの時点でボールトにある、既存のアーカイブをすべて削除する必要があります。

ボールトは、プログラムまたは S3 Glacier コンソールを使用して削除できます。プログラムを使用してボールトを削除する方法の詳細については、「Amazon S3 Glacier でボールトを削除する」を参照してください。

> **Important**
>
> 過去 24 時間以内にボールトにアーカイブをアップロードしたり、ボールトからアーカイブを削除したりした場合は、最新の情報を反映するためにボールトインベントリが更新されるまで待機する必要があります。S3 Glacier では、各ボールトについて 24 時間ごとにインベントリを作成します。

空のボールトを削除するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/glacier/home で S3 Glacier コンソールを開きます。
2. リージョンの選択メニューから、削除するボールト AWS リージョン の を選択します。

   この入門演習では、米国西部 (オレゴン) リージョンにサンプルのボールトがあります。
3. 削除する空のボールトの名前の横にあるオプションボタンを選択します。ボールトが空でない場合、ボールトを削除する前にすべてのアーカイブを削除する必要があります。詳細については、「Amazon S3 Glacier でアーカイブを削除する」を参照してください。

> **Important**
>
> ボールトの削除は元に戻せません。

4. [削除] を選択します。
5. [ボールトの削除] ダイアログボックスが表示されます。[削除] を選択します。

空でないボールトを削除するには

1. 空でないボールトを削除する場合は、ボールトを削除する前に、まず既存のアーカイブをすべて削除する必要があります。これを行うには、コードを記述して、REST API、、 AWS SDK for Java、 AWS SDK for .NET または のいずれかを使用してアーカイブの削除リクエストを行います AWS CLI。アーカイブの削除の詳細については、「ステップ 5: S3 Glacier でボールトからアーカイブを削除する」を参照してください。
2. ボールトが空になったら、前に説明した空のボールトを削除する手順に従います。

ここからどこへ進むべきですか?

これで入門演習が完了しました。次の各セクションでは Amazon S3 Glacier の詳細について説明します。

- Amazon S3 Glacier のボールトの操作
- Amazon S3 Glacier でのアーカイブの操作

Amazon S3 Glacier のボールトの操作

ボールトは、アーカイブを格納するコンテナです。ボールトを作成するときは、ボールト名とボールト AWS リージョン を作成する を指定します。S3 Glacier でサポートされる AWS リージョン のリストについては、「AWS 全般リファレンス」の「Amazon S3 Glacier エンドポイントとクォータ」を参照してください。

ボールトに格納できるアーカイブの数に制限はありません。

> **Important**
>
> S3 Glacier は、コンソールも提供します。ただし、アップロード、ダウンロード、削除などのアーカイブオペレーションでは、 AWS Command Line Interface （ AWS CLI) を使用するか、コードを記述する必要があります。アーカイブオペレーションについては、コンソールによるサポートはありません。例えば、写真、動画、その他のドキュメントなどのデータをアップロードするには、 を使用するか、REST API を直接使用するか、 AWS SDKs を使用して、コードを AWS CLI 記述してリクエストを行う必要があります。
> をインストールするには AWS CLI、「」を参照してくださいAWS Command Line Interface。で S3 Glacier を使用する方法の詳細については AWS CLI、AWS CLI S3 Glacier のリファレンス」を参照してください。を使用して S3 Glacier にアーカイブ AWS CLI をアップロードする例については、「 での S3 Glacier AWS Command Line Interface の使用」を参照してください。

トピック

- S3 Glacier でのボールトオペレーション
- Amazon S3 Glacier でボールトを作成する
- Amazon S3 Glacier でボールトメタデータを取得する
- Amazon S3 Glacier でボールトインベントリをダウンロードする
- Amazon S3 Glacier でのボールト通知の設定
- Amazon S3 Glacier でボールトを削除する
- S3 Glacier ボールトにタグを付ける
- S3 Glacier ボールトロック

S3 Glacier でのボールトオペレーション

S3 Glacier は多様なボールトオペレーションをサポートしています。ボールトオペレーションは、特定の AWS リージョンに固有です。つまり、ボールトを作成するときは、特定の AWS リージョンにボールトを作成します。ボールトを一覧表示すると、S3 Glacier によって、リクエストで指定した AWS リージョン のボールトのリストが返されます。

ボールトの作成と削除

AWS アカウント では、 あたり最大 1,000 個のボールトを作成できます AWS リージョン。S3 Glacier でサポートされる AWS リージョン のリストについては、「AWS 全般リファレンス」の「Amazon S3 Glacier エンドポイントとクォータ」を参照してください。

S3 Glacier が最後にインベントリを計算した時点でボールト内にアーカイブがなく、また、最後のインベントリからボールトへの書き込みがない場合にのみ、ボールトを削除できます。

> **Note**
>
> S3 Glacier では、各ボールトについて 24 時間ごとにインベントリを作成します。インベントリには最新の情報が反映されていないことがあるため、S3 Glacier では、ボールトが実際に空であるかどうかを確認するために、最後にボールトインベントリが作成されてから書き込み オペレーションがあったかどうかを確認します。

詳細については、Amazon S3 Glacier でボールトを作成するおよびAmazon S3 Glacier でボールトを削除するを参照してください。

ボールトメタデータの取得

ボールトの 作成日、ボールトの中のアーカイブの数、すべてのアーカイブの合計サイズなど、ボールトに関する情報を返します。S3 Glacier は、特定のボールトまたはアカウント内の特定の にあるすべてのボールトのこの情報を取得するための API コール AWS リージョン を提供します。詳細については、「Amazon S3 Glacier でボールトメタデータを取得する」を参照してください。

ボールトインベントリのダウンロード

ボールトインベントリとは、ボールト内のアーカイブのリストを指します。インベントリではリスト内の各アーカイブに、アーカイブ ID、作成日、サイズなど、アーカイブに関する情報が記載されて

います。S3 Glacier はボールトインベントリを 1 日 1 回のペースで更新します。この更新は、アーカイブがボールトに最初にアップロードされた日から開始されます。ボールトインベントリをダウンロードするには、ボールトインベントリが存在している必要があります。

ボールトインベントリのダウンロードは、非同期オペレーションです。まず、インベントリをダウンロードするジョブを開始する必要があります。S3 Glacier はジョブのリクエストを受け取った後で、ダウンロードするインベントリを準備します。ジョブが完了したら、インベントリデータをダウンロードできます。

このジョブの非同期であるという性質を考慮し、ジョブの完了時に Amazon Simple Notification Service (Amazon SNS) 通知を使用して通知を送信することができます。個々のジョブリクエストごとに Amazon SNS のトピックを指定することも、特定のボールトイベントが発生したときに通知を送信するようにボールトを設定することもできます。

S3 Glacier では、各ボールトについて 24 時間ごとにインベントリを作成します。最後のインベントリ以降に、ボールトに対してアーカイブの追加や削除が行われていない場合、インベントリの日付は更新されません。

ボールトインベントリに対してジョブを開始すると、S3 Glacier により最後に生成されたインベントリが返されます。そのインベントリはポイントインタイムのスナップショットであり、リアルタイムのデータではありません。アーカイブをアップロードするごとにボールトインベントリを取得することは、あまり便利には感じられないかもしれません。しかし、S3 Glacier にアップロードしたアーカイブに関するメタデータに含まれるデータベースをクライアント側で管理する場合を考えてみてください。そのような場合には、実際のボールトインベントリとデータベース内の情報とを照合できるため、ボールトインベントリの利便性が実感できるものと思われます。

ボールトインベントリの取得の詳細については、「Amazon S3 Glacier でボールトインベントリをダウンロードする」を参照してください。

ボールト通知の設定

S3 Glacier からのボールトのアーカイブやボールトインベントリなどの取得は 2 ステップのプロセスです。まず、ジョブを開始します。ジョブが完了したら、ジョブの出力をダウンロードします。ジョブがいつ完了したかを確認するには、S3 Glacier の通知を使用できます。S3 Glacier は、指定した Amazon Simple Notification Service (Amazon SNS) トピックに通知メッセージを送信します。

ボールトに通知を設定できるほか、ボールトイベントや、そのイベントが発生したときに通知を送信する Amazon SNS トピックを指定できます。S3 Glacier は、ボールトイベントが発生した時点で、指定された Amazon SNS トピックに通知を送信します。詳細については、「Amazon S3 Glacier でのボールト通知の設定」を参照してください。

Amazon S3 Glacier でボールトを作成する

ボールトを作成すると、アカウントのボールトのセットにボールトが追加されます。 AWS アカウント は、 AWS リージョンごとに最大 1,000 個のボールトを作成できます。Amazon S3 Glacier (S3 Glacier) でサポートされている 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「リージョンとエンドポイント」を参照してください。

ボールトの作成時に、ボールト名を指定する必要があります。以下に、ボールト名の要件を示します。

- 名前は 1 ~ 255 文字の長さです。
- 使用できる文字は、a-z、A-Z、0-9、'_' (アンダースコア)、'-' (ハイフン)、'.' (ピリオド) です。

ボールト名は、ボールトが作成されるアカウントと AWS リージョン内で一意である必要があります。つまり、アカウントは異なる AWS リージョンで同じ名前のボールトを作成できますが、同じ AWS リージョンでは作成できません。

トピック

- を使用した Amazon S3 Glacier でのボールトの作成 AWS SDK for Java
- を使用した Amazon S3 Glacier でのボールトの作成 AWS SDK for .NET
- Amazon S3 Glacier での REST API を使用したボールトの作成
- Amazon S3 Glacier のコンソールを使用してボールトを作成します。
- を使用した Amazon S3 Glacier でのボールトの作成 AWS Command Line Interface

を使用した Amazon S3 Glacier でのボールトの作成 AWS SDK for Java

低レベル API は、ボールトの作成と削除、ボールトの説明の取得、特定の で作成されたボールトのリストの取得など、すべてのボールトオペレーションのメソッドを提供します AWS リージョン。以下に、 AWS SDK for Javaを使用してボールトを作成する手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   ボールト AWS リージョン を作成する を指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `CreateVaultRequest` クラスのインスタンスを作成することにより、リクエスト情報を指定します。

   Amazon S3 Glacier (S3 Glacier) には、ボールト名とアカウント ID を指定する必要があります アカウント ID を指定しなかった場合には、リクエストに署名する際に指定した認証情報に関連付けられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3 」を参照してください。

3. リクエストオブジェクトをパラメータとして指定して、`createVault` メソッドを実行します。

   S3 Glacier が返すレスポンスは、`CreateVaultResult` オブジェクトで確認できます。

以下の Java コードスニペットは、前述の手順を示しています。このスニペットでは、`us-west-2` リージョンにボールトを作成します。`Location` 出力される は、アカウント ID、、ボールト名を含むボールトの相対 URI AWS リージョンです。

```
AmazonGlacierClient client = new AmazonGlacierClient(credentials);
client.setEndpoint("https://glacier.us-west-2.amazonaws.com");

CreateVaultRequest request = new CreateVaultRequest()
    .withVaultName("*** provide vault name ***");
CreateVaultResult result = client.createVault(request);

System.out.println("Created vault successfully: " + result.getLocation());
```

> **Note**
>
> 基本となる REST API については、「ボールトの作成 (PUT vault)」を参照してください。

例: を使用したボールトの作成 AWS SDK for Java

次の Java コード例では、 `us-west-2`リージョンにボールトを作成します (詳細については AWS リージョン、「」を参照してくださいAmazon S3 Glacier へのアクセス) 。さらに、コード例ではボールト情報を取得し、同じ 内のすべてのボールトを一覧表示してから AWS リージョン、作成されたボールトを削除します。

以下の例を実行するための詳しい手順については、「Eclipse を使用した Amazon S3 Glacier の Java 実行例」を参照してください。

Example

```
import java.io.IOException;
import java.util.List;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.CreateVaultRequest;
import com.amazonaws.services.glacier.model.CreateVaultResult;
import com.amazonaws.services.glacier.model.DeleteVaultRequest;
import com.amazonaws.services.glacier.model.DescribeVaultOutput;
import com.amazonaws.services.glacier.model.DescribeVaultRequest;
import com.amazonaws.services.glacier.model.DescribeVaultResult;
import com.amazonaws.services.glacier.model.ListVaultsRequest;
import com.amazonaws.services.glacier.model.ListVaultsResult;


public class AmazonGlacierVaultOperations {

    public static AmazonGlacierClient client;

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

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-east-1.amazonaws.com/");

        String vaultName = "examplevaultfordelete";

        try {
            createVault(client, vaultName);
            describeVault(client, vaultName);
            listVaults(client);
            deleteVault(client, vaultName);

        } catch (Exception e) {
            System.err.println("Vault operation failed." + e.getMessage());
        }
    }

    private static void createVault(AmazonGlacierClient client, String vaultName) {
        CreateVaultRequest createVaultRequest = new CreateVaultRequest()
            .withVaultName(vaultName);
```

```
        CreateVaultResult createVaultResult = client.createVault(createVaultRequest);

        System.out.println("Created vault successfully: " +
 createVaultResult.getLocation());
    }

    private static void describeVault(AmazonGlacierClient client, String vaultName) {
        DescribeVaultRequest describeVaultRequest = new DescribeVaultRequest()
            .withVaultName(vaultName);
        DescribeVaultResult describeVaultResult  =
 client.describeVault(describeVaultRequest);

        System.out.println("Describing the vault: " + vaultName);
        System.out.print(
                "CreationDate: " + describeVaultResult.getCreationDate() +
                "\nLastInventoryDate: " + describeVaultResult.getLastInventoryDate() +
                "\nNumberOfArchives: " + describeVaultResult.getNumberOfArchives() +
                "\nSizeInBytes: " + describeVaultResult.getSizeInBytes() +
                "\nVaultARN: " + describeVaultResult.getVaultARN() +
                "\nVaultName: " + describeVaultResult.getVaultName());
    }

    private static void listVaults(AmazonGlacierClient client) {
        ListVaultsRequest listVaultsRequest = new ListVaultsRequest();
        ListVaultsResult listVaultsResult = client.listVaults(listVaultsRequest);

        List<DescribeVaultOutput> vaultList = listVaultsResult.getVaultList();
        System.out.println("\nDescribing all vaults (vault list):");
        for (DescribeVaultOutput vault : vaultList) {
            System.out.println(
                    "\nCreationDate: " + vault.getCreationDate() +
                    "\nLastInventoryDate: " + vault.getLastInventoryDate() +
                    "\nNumberOfArchives: " + vault.getNumberOfArchives() +
                    "\nSizeInBytes: " + vault.getSizeInBytes() +
                    "\nVaultARN: " + vault.getVaultARN() +
                    "\nVaultName: " + vault.getVaultName());
        }
    }

    private static void deleteVault(AmazonGlacierClient client, String vaultName) {
        DeleteVaultRequest request = new DeleteVaultRequest()
            .withVaultName(vaultName);
        client.deleteVault(request);
        System.out.println("Deleted vault: " + vaultName);
```

```
    }

}
```

を使用した Amazon S3 Glacier でのボールトの作成 AWS SDK for .NET

どちらも高レベル API と低レベル APIAmazon SDK で.NET で提供されているボールト作成のためのメソッドがあります。

トピック

- AWS SDK for .NETの高レベル API を使用したボールトの作成
- の低レベル API を使用したボールトの作成 AWS SDK for .NET

AWS SDK for .NETの高レベル API を使用したボールトの作成

高レベル API `ArchiveTransferManager` クラスは、 AWS リージョンでボールト作成に使用できる `CreateVault` メソッドを提供します。

例: の高レベル API を使用したボールトオペレーション AWS SDK for .NET

以下の C# コード例では、US West (Oregon)Region でボールトを作成および削除します。ボールトを作成 AWS リージョン できる のリストについては、「」を参照してくださいAmazon S3 Glacier へのアクセス。

以下の例を実行するための詳しい手順については、「コード例の実行」を参照してください。ボールト名に示すコードを更新する必要があります。

Example

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class VaultCreateDescribeListVaultsDeleteHighLevel
  {
    static string vaultName = "*** Provide vault name ***";

    public static void Main(string[] args)
```

```
    {
      try
      {
          var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
          manager.CreateVault(vaultName);
          Console.WriteLine("Vault created. To delete the vault, press Enter");
          Console.ReadKey();
          manager.DeleteVault(vaultName);
          Console.WriteLine("\nVault deleted. To continue, press Enter");
          Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }
  }
}
```

の低レベル API を使用したボールトの作成 AWS SDK for .NET

低レベル API は、ボールトの作成と削除、ボールトの説明の取得、特定の で作成されたボールトのリストの取得など、すべてのボールトオペレーションのメソッドを提供します AWS リージョン。以下に、 AWS SDK for .NETを使用してボールトを作成する手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   ボールト AWS リージョン を作成する を指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `CreateVaultRequest` クラスのインスタンスを作成することにより、リクエスト情報を指定します。

   Amazon S3 Glacier (S3 Glacier) には、ボールト名とアカウント ID を指定する必要があります アカウント ID を指定しなかった場合は、リクエストに署名する際に指定した認証情報に関連づけられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for .NET での の使用 Amazon S3 」を参照してください。

3. リクエストオブジェクトをパラメータとして指定して、`CreateVault` メソッドを実行します。

   S3 Glacier が返すレスポンスは、`CreateVaultResponse` オブジェクトで確認できます。

例: の低レベル API を使用したボールトオペレーション AWS SDK for .NET

以下の C# の例は、前述の手順を示しています。この例では、米国西部 (オレゴン) リージョンにボールトを作成します。さらに、コード例ではボールト情報を取得し、同じ 内のすべてのボールトを一覧表示してから AWS リージョン、作成されたボールトを削除します。`Location` 出力されるは、アカウント ID、、ボールト名を含むボールトの相対 URI AWS リージョンです。

> **Note**
>
> 基本となる REST API については、「ボールトの作成 (PUT vault)」を参照してください。

以下の例を実行するための詳しい手順については、「コード例の実行」を参照してください。ボールト名に示すコードを更新する必要があります。

Example

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class VaultCreateDescribeListVaultsDelete
  {
    static string vaultName = "*** Provide vault name ***";
    static AmazonGlacierClient client;

    public static void Main(string[] args)
    {
       try
      {
         using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
          Console.WriteLine("Creating a vault.");
          CreateAVault();
          DescribeVault();
          GetVaultsList();
          Console.WriteLine("\nVault created. Now press Enter to delete the vault...");
          Console.ReadKey();
          DeleteVault();
```

```
        }
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static void CreateAVault()
    {
      CreateVaultRequest request = new CreateVaultRequest()
      {
        VaultName = vaultName
      };
      CreateVaultResponse response = client.CreateVault(request);
      Console.WriteLine("Vault created: {0}\n", response.Location);
    }

    static void DescribeVault()
    {
      DescribeVaultRequest describeVaultRequest = new DescribeVaultRequest()
      {
        VaultName = vaultName
      };

      DescribeVaultResponse describeVaultResponse =
  client.DescribeVault(describeVaultRequest);
      Console.WriteLine("\nVault description...");
      Console.WriteLine(
        "\nVaultName: " + describeVaultResponse.VaultName +
        "\nVaultARN: " + describeVaultResponse.VaultARN +
        "\nVaultCreationDate: " + describeVaultResponse.CreationDate +
        "\nNumberOfArchives: " + describeVaultResponse.NumberOfArchives +
        "\nSizeInBytes: " + describeVaultResponse.SizeInBytes +
        "\nLastInventoryDate: " + describeVaultResponse.LastInventoryDate
        );
    }

    static void GetVaultsList()
    {
      string lastMarker = null;
      Console.WriteLine("\n List of vaults in your account in the specific
  region ...");
```

```
      do
      {
        ListVaultsRequest request = new ListVaultsRequest()
        {
          Marker = lastMarker
        };
        ListVaultsResponse response = client.ListVaults(request);

        foreach (DescribeVaultOutput output in response.VaultList)
        {
          Console.WriteLine("Vault Name: {0} \tCreation Date: {1} \t #of archives:
 {2}",
                            output.VaultName, output.CreationDate,
 output.NumberOfArchives);
        }
        lastMarker = response.Marker;
      } while (lastMarker != null);
    }

    static void DeleteVault()
    {
      DeleteVaultRequest request = new DeleteVaultRequest()
      {
        VaultName = vaultName
      };
      DeleteVaultResponse response = client.DeleteVault(request);
    }
  }
}
```

Amazon S3 Glacier での REST API を使用したボールトの作成

REST API を使用してボールトを作成するには、「ボールトの作成 (PUT vault)」を参照してください。

Amazon S3 Glacier のコンソールを使用してボールトを作成します。

Amazon S3 Glacier (S3 Glacier) コンソールを使用してボールトを作成するには、「」を参照してください。ステップ 2: S3 Glacier でボールトを作成するの開始方法チュートリアル。

を使用した Amazon S3 Glacier でのボールトの作成 AWS Command Line Interface

AWS Command Line Interface (AWS CLI)を使用して、Amazon S3 Glacier (S3 Glacier) でボールトを作成するには、次の手順に従います。

トピック

- (前提条件) のセットアップ AWS CLI
- 例: を使用してボールトを作成する AWS CLI

(前提条件) のセットアップ AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

   のインストール AWS Command Line Interface

   の設定 AWS Command Line Interface

2. コマンドプロンプトで次のコマンドを入力して、 AWS CLI セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを使用してください。

     ```
     aws help
     ```

   - 設定したアカウントの S3 Glacier ボールトのリストを取得するには、`list-vaults` コマンドを使用します。*123456789012* 「」を AWS アカウント ID に置き換えます。

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - の現在の設定データを表示するには AWS CLI、 `aws configure list` コマンドを使用します。

     ```
     aws configure list
     ```

例: を使用してボールトを作成する AWS CLI

1. を使用するcreate-vaultという名前のボールトを作成するコマンド*awsexampleVault*アカウントの下に*111122223333*。

```
aws glacier create-vault --vault-name awsexamplevault --account-id 111122223333
```

正常な出力:

```
{
    "location": "/111122223333/vaults/awsexamplevault"
}
```

2. describe-vault コマンドを使用して作成を確認します。

```
aws glacier describe-vault --vault-name awsexamplevault --account-id 111122223333
```

Amazon S3 Glacier でボールトメタデータを取得する

ボールトの 作成日、アーカイブの数、すべてのアーカイブの合計サイズなど、ボールトに関する情報を返します。Amazon S3 Glacier (S3 Glacier) は、アカウント内の特定の AWS リージョン内の特定のボールトまたはすべてのボールトについて、この情報を取得するための API コールを提供します。

ボールトリストを取得すると、S3 Glacier は、ボールト名の ASCII 値でソートされたリストを返します。リストには最大 1,000 のボールトが含まれます。リストを維持するマーカーに対するレスポンスを常に確認する必要があります。それ以上の項目が存在しない場合、マーカーフィールドは null です。レスポンスで返されるボールトの数を制限することもできます。レスポンスで多くのボールトが返されると、結果はページ分割されます。ボールトの次のセットを取得するには、追加リクエストを送信する必要があります。

トピック

- を使用した Amazon S3 Glacier でのボールトメタデータの取得 AWS SDK for Java
- を使用した Amazon S3 Glacier でのボールトメタデータの取得 AWS SDK for .NET
- REST API を使用したボールトメタデータの取得

- を使用した Amazon S3 Glacier でのボールトメタデータの取得 AWS Command Line Interface

を使用した Amazon S3 Glacier でのボールトメタデータの取得 AWS SDK for Java

トピック

- 特定のボールトのボールトメタデータの取得
- リージョン内のすべてのボールトのボールトメタデータの取得
- 例: Amazon SDK for Java によるボールトメタデータの取得

特定のボールトのボールトメタデータの取得

特定のボールトまたは特定の AWS リージョンのすべてのボールトのメタデータを取得できます。以下に、Amazon SDK for Java の低レベル API を使用して特定のボールトのボールトメタデータを取得する手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   ボールトが存在する AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `DescribeVaultRequest` クラスのインスタンスを作成することにより、リクエスト情報を指定します。

   Amazon S3 Glacier (S3 Glacier) には、ボールト名とアカウント ID を指定する必要があります アカウント ID を指定しなかった場合には、リクエストに署名する際に使用した認証情報に関連付けられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3 」を参照してください。

3. リクエストオブジェクトをパラメータとして指定して、`describeVault` メソッドを実行します。

   S3 Glacier が返すボールトメタデータ情報は、`DescribeVaultResult` オブジェクトで使用できます。

以下の Java コードスニペットは、前述の手順を示しています。

```
DescribeVaultRequest request = new DescribeVaultRequest()
 .withVaultName("*** provide vault name***");

DescribeVaultResult result = client.describeVault(request);

System.out.print(
        "\nCreationDate: " + result.getCreationDate() +
        "\nLastInventoryDate: " + result.getLastInventoryDate() +
        "\nNumberOfArchives: " + result.getNumberOfArchives() +
        "\nSizeInBytes: " + result.getSizeInBytes() +
        "\nVaultARN: " + result.getVaultARN() +
        "\nVaultName: " + result.getVaultName());
```

> **Note**
>
> 基本となる REST API については、「ボールトの説明 (GET vault)」を参照してください。

リージョン内のすべてのボールトのボールトメタデータの取得

`listVaults` メソッドを使用して、特定の AWS リージョン内のすべてのボールトのメタデータを取得することもできます。

以下の Java コードスニペットにより、`us-west-2` リージョン内のボールトのリストを取得します。リクエストでは、レスポンスで返されるボールトの数を 5 に制限します。次に、コードスニペットは、 AWS リージョンからボールトリスト全体を取得するために一連の`listVaults`呼び出しを行います。

```
AmazonGlacierClient client;
client.setEndpoint("https://glacier.us-west-2.amazonaws.com/");

String marker = null;
do {
    ListVaultsRequest request = new ListVaultsRequest()
        .withLimit("5")
        .withMarker(marker);
    ListVaultsResult listVaultsResult = client.listVaults(request);

    List<DescribeVaultOutput> vaultList = listVaultsResult.getVaultList();
```

```
    marker = listVaultsResult.getMarker();
    for (DescribeVaultOutput vault : vaultList) {
        System.out.println(
                "\nCreationDate: " + vault.getCreationDate() +
                "\nLastInventoryDate: " + vault.getLastInventoryDate() +
                "\nNumberOfArchives: " + vault.getNumberOfArchives() +
                "\nSizeInBytes: " + vault.getSizeInBytes() +
                "\nVaultARN: " + vault.getVaultARN() +
                "\nVaultName: " + vault.getVaultName());
    }
} while (marker != null);
```

このコードスニペットにおいて、リクエストの `Limit` の値を指定しなかった場合、S3 Glacier は、S3 Glacier API に定められているように、最大 10 個のボールトを返します。リストに表示するボールトがさらに存在する場合、レスポンスの `marker` フィールドにボールト Amazon リソースネーム (ARN) が含まれ、この中に新しいリクエストのリストの続きが含まれます。そうでない場合、`marker` フィールドは null になります。

リスト内の各ボールトに対して返される情報は、特定のボールトに対して `describeVault` メソッドを呼び出して取得する情報と同じであることに注意してください。

> **Note**
>
> `listVaults` メソッドは基本となる REST API を呼び出します (「ボールトのリスト (GET vaults)」を参照)。

例: Amazon SDK for Java によるボールトメタデータの取得

コード例については、「例: を使用したボールトの作成 AWS SDK for Java」を参照してください。この Java コード例では、ボールトを作成し、ボールトメタデータを取得します。

を使用した Amazon S3 Glacier でのボールトメタデータの取得 AWS SDK for .NET

トピック

- 特定のボールトのボールトメタデータの取得
- リージョン内のすべてのボールトのボールトメタデータの取得

- 例: の低レベル API を使用したボールトメタデータの取得 AWS SDK for .NET

特定のボールトのボールトメタデータの取得

特定のボールトまたは特定の AWS リージョンのすべてのボールトのメタデータを取得できます。以下に、 AWS SDK for .NETの低レベル API を使用して特定のボールトのボールトメタデータを取得する手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   ボールトが存在する AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。
2. `DescribeVaultRequest` クラスのインスタンスを作成することにより、リクエスト情報を指定します。

   Amazon S3 Glacier (S3 Glacier) には、ボールト名とアカウント ID を指定する必要があります アカウント ID を指定しなかった場合には、リクエストに署名する際に使用した認証情報に関連付けられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for .NET での の使用 Amazon S3 」を参照してください。
3. リクエストオブジェクトをパラメータとして指定して、`DescribeVault` メソッドを実行します。

   S3 Glacier が返すボールトメタデータ情報は、`DescribeVaultResult` オブジェクトで使用できます。

以下の C# コードスニペットは、前述の手順を示しています。このスニペットでは、米国西部 (オレゴン) リージョン 内の既存のボールトのメタデータ情報を取得します。

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);

DescribeVaultRequest describeVaultRequest = new DescribeVaultRequest()
{
  VaultName = "*** Provide vault name ***"
};
DescribeVaultResponse describeVaultResponse =
 client.DescribeVault(describeVaultRequest);
Console.WriteLine("\nVault description...");
```

```
Console.WriteLine(
   "\nVaultName: " + describeVaultResponse.VaultName +
   "\nVaultARN: " + describeVaultResponse.VaultARN +
   "\nVaultCreationDate: " + describeVaultResponse.CreationDate +
   "\nNumberOfArchives: " + describeVaultResponse.NumberOfArchives +
   "\nSizeInBytes: " + describeVaultResponse.SizeInBytes +
   "\nLastInventoryDate: " + describeVaultResponse.LastInventoryDate
   );
```

> **Note**
>
> 基本となる REST API については、「ボールトの説明 (GET vault)」を参照してください。

リージョン内のすべてのボールトのボールトメタデータの取得

`ListVaults` メソッドを使用して、特定の AWS リージョン内のすべてのボールトのメタデータを取得することもできます。

次の C# コードスニペットでは、米国西部 (オレゴン) リージョン のボールトのリストを取得します。リクエストでは、レスポンスで返されるボールトの数を 5 に制限します。次に、コードスニペットは、 AWS リージョンからボールトリスト全体を取得するために一連の`ListVaults`呼び出しを行います。

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);
string lastMarker = null;
Console.WriteLine("\n List of vaults in your account in the specific AWS Region ...");
do
{
  ListVaultsRequest request = new ListVaultsRequest()
  {
    Limit = 5,
    Marker = lastMarker
  };
  ListVaultsResponse response = client.ListVaults(request);

  foreach (DescribeVaultOutput output in response.VaultList)
  {
    Console.WriteLine("Vault Name: {0} \tCreation Date: {1} \t #of archives: {2}",
```

```
                        output.VaultName, output.CreationDate, output.NumberOfArchives);
  }
  lastMarker = response.Marker;
} while (lastMarker != null);
```

このコードスニペットにおいて、リクエストの `Limit` の値を指定しなかった場合、S3 Glacier は、S3 Glacier API に定められているように、最大 10 個のボールトを返します。

リスト内の各ボールトに対して返される情報は、特定のボールトに対して `DescribeVault` メソッドを呼び出して取得する情報と同じであることに注意してください。

> **Note**
>
> `ListVaults` メソッドは基本となる REST API を呼び出します (「ボールトのリスト (GET vaults)」を参照)。

例: の低レベル API を使用したボールトメタデータの取得 AWS SDK for .NET

コード例については、「例: の低レベル API を使用したボールトオペレーション AWS SDK for .NET」を参照してください。この C# コード例では、ボールトを作成し、ボールトメタデータを取得します。

REST API を使用したボールトメタデータの取得

REST API を使用してボールトをリストするには、「ボールトのリスト (GET vaults)」を参照してください。1 つのボールトを説明するには、「ボールトの説明 (GET vault)」を参照してください。

を使用した Amazon S3 Glacier でのボールトメタデータの取得 AWS Command Line Interface

この例では、 AWS Command Line Interface () を使用して Amazon S3 Glacier (S3 Glacier) でボールト情報とメタデータを取得する方法を示しますAWS CLI。

トピック

- (前提条件) のセットアップ AWS CLI
- 例: を使用したボールトメタデータの取得 AWS CLI

(前提条件) のセットアップ AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

   のインストール AWS Command Line Interface

   の設定 AWS Command Line Interface

2. コマンドプロンプトで次のコマンドを入力して、 AWS CLI セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを使用してください。

     ```
     aws help
     ```

   - 設定したアカウントの S3 Glacier ボールトのリストを取得するには、`list-vaults` コマンドを使用します。*123456789012* 「」を AWS アカウント ID に置き換えます。

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - の現在の設定データを表示するには AWS CLI、 `aws configure list` コマンドを使用します。

     ```
     aws configure list
     ```

例: を使用したボールトメタデータの取得 AWS CLI

- 使用する`describe-vault`という名前のボールトを記述するコマンド*awsexampleVault*アカウントの下に*111122223333*。

  ```
  aws glacier describe-vault --vault-name awsexamplevault --account-id 111122223333
  ```

Amazon S3 Glacier でボールトインベントリをダウンロードする

最初のアーカイブをボールトにアップロードすると、Amazon S3 Glacier (S3 Glacier) により、ボールトインベントリが自動的に作成され、インベントリが約 1 日 1 回のペースで更新されます。S3

Glacier によって最初に作成されるインベントリは、取得できるようになるまで、通常半日から最大 1 日かかります。次の 2 ステップのプロセスで、S3 Glacier からボールトインベントリを取得できます。

1. ジョブの開始 (ジョブの POST) オペレーションを使用して、インベントリの取得ジョブを開始します。

> **Important**
>
> データ取り出しポリシーにより、`PolicyEnforcedException` 例外が発生して、取り出しジョブの開始リクエストが失敗することがあります。データ取り出しポリシーの詳細については、「S3 Glacier データ取得ポリシー」を参照してください。`PolicyEnforcedException` 例外の詳細については、「エラーレスポンス」を参照してください。

2. ジョブが完了したら、ジョブの出力の取得 (GET output) オペレーションを使用してバイトをダウンロードします。

たとえば、アーカイブまたはボールトインベントリを取得するには、最初に取得ジョブを開始する必要があります。ジョブのリクエストは非同期的に実行されます。取得ジョブを開始すると、S3 Glacier はジョブを作成し、レスポンスでジョブ ID を返します。S3 Glacier がジョブを完了すると、ジョブの出力、アーカイブのバイト数、またはボールトインベントリデータを取得できます。

出力を取得する前にジョブが完了している必要があります。次のオプションを使用してジョブのステータスを確認できます。

- ジョブの完了の通知を待つ ジョブの完了後に S3 Glacier が通知を投稿する Amazon Simple Notification Service (Amazon SNS) トピックを指定できます。以下のメソッドを使用して、Amazon SNS トピックを指定できます。
  - ジョブごとに Amazon SNS トピックを指定する。

    ジョブを開始する際に、オプションで Amazon SNS トピックを指定できます。
  - ボールトに通知設定を指定する。

    ボールトに特定のイベントに対する通知 設定を指定できます。(Amazon S3 Glacier でのボールト通知の設定 を参照) 。S3 Glacier は、特定のイベントが発生するたびに、指定された SNS トピックにメッセージを送信します。

ボールトに通知設定を指定し、さらに、ジョブを開始する際に Amazon SNS トピックを指定した場合、S3 Glacier は両方のトピックにジョブの完了メッセージを送信します。

E メールで通知を受け取るか、アプリケーションがポーリングできる Amazon Simple Queue Service (Amazon SQS) にメッセージを格納するように SNS トピックを設定できます。メッセージがキューに表示されたら、ジョブが正常に完了したかどうかを確認し、ジョブの出力をダウンロードできます。

- 明示的にジョブ情報をリクエストする -S3 Glacier では、ジョブの説明オペレーション (ジョブの説明 (GET JobID)) も用意されており、ジョブの情報をポーリングできるようになっています。このリクエストを定期的に送信して、ジョブ情報を取得できます。ただし、Amazon SNS 通知を使用することをお勧めします。

> **Note**
>
> SNS 通知により取得する情報は、ジョブの説明を呼び出して取得する情報と同じです。

トピック

- インベントリについて
- を使用した Amazon S3 Glacier でのボールトインベントリのダウンロード AWS SDK for Java
- Amazon S3 Glacier で AWS SDK for .NETを使用してボールトインベントリをダウンロードする
- REST API を使用してボールトインベントリをダウンロードする
- を使用した Amazon S3 Glacier でのボールトインベントリのダウンロード AWS Command Line Interface

インベントリについて

S3 Glacier はボールトインベントリを少なくとも 1 日 1 回のペースで更新します。この更新は、アーカイブを最初にボールトにアップロードした日から開始されます。最後のインベントリ以降に、ボールトに対してアーカイブの追加や削除が行われていない場合、インベントリの日付は更新されません。ボールトインベントリに対してジョブを開始すると、S3 Glacier により最後に生成されたインベントリが返されます。そのインベントリはポイントインタイムのスナップショットであり、リアルタイムのデータではありません。S3 Glacier によりボールトに対して最初に作成されるインベントリは、取得できるようになるまで、通常半日から最大 1 日かかることに注意してください。

アーカイブをアップロードするごとにボールトインベントリを取得することは、あまり便利には感じられないかもしれません。しかし、S3 Glacier にアップロードしたアーカイブに関するメタデータに関連付けられたデータベースをクライアント側で管理する場合を考えてみてください。そのような場合には、実際のボールトインベントリとデータベース内の情報とを必要に応じて照合できるため、ボールトインベントリの利便性が実感できるものと思われます。アーカイブの作成日でフィルタするか、クォータを設定することによって、取得されるインベントリの項目数を制限できます。インベントリの取得の制限の詳細については、「インベントリの取得の範囲」を参照してください。

インベントリは、カンマ区切り値 (CSV) と JSON の 2 つの形式で返すことができます。インベントリジョブを開始する際に、オプションで形式を指定できます。デフォルト形式は JSON です。インベントリジョブの出力で返されるデータフィールドの詳細については、Get Job Output APIの「レスポンス本文」を参照してください。

を使用した Amazon S3 Glacier でのボールトインベントリのダウンロード AWS SDK for Java

以下に、 AWS SDK for Javaの低レベル API を使用してボールトインベントリを取得する手順を示します。高レベル API では、ボールトインベントリの取得はサポートされていません。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   ボールトが存在する AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `initiateJob` メソッドを実行してインベントリの取得ジョブを開始します。

   `InitiateJobRequest` オブジェクトにジョブ情報を指定して、`initiateJob`を実行します。

   > **Note**
   > ボールトのインベントリが完了していない場合は、エラーが返されることに注意してください。Amazon S3 Glacier (S3 Glacier)では、各ボールトについて 24 時間ごとにインベントリを作成します。

   S3 Glacier は、レスポンスとしてジョブ ID を返します。レスポンスは、`InitiateJobResult` クラスのインスタンスで使用できます。

```
InitiateJobRequest initJobRequest = new InitiateJobRequest()
    .withVaultName("*** provide vault name ***")
    .withJobParameters(
            new JobParameters()
                .withType("inventory-retrieval")
                .withSNSTopic("*** provide SNS topic ARN ****")
      );

InitiateJobResult initJobResult = client.initiateJob(initJobRequest);
String jobId = initJobResult.getJobId();
```

3. ジョブが完了するまで待ちます。

   ジョブの出力をダウンロードする準備が整うまで待つ必要があります。ボールトに通知設定を指定している場合、またはジョブを開始したときに Amazon Simple Notification Service (Amazon SNS) トピックを指定している場合は、ジョブの完了後に S3 Glacier からそのトピックにメッセージが送信されます。

   また、`describeJob` メソッドを呼び出して S3 Glacier にポーリングすることで、ジョブの完了ステータスを調べることもできます。ただし、通知のために Amazon SNS トピックを使用する方法が推奨されています。以下のセクションに示しているコード例では、Amazon SNS を使用して、 でメッセージを発行します。

4. `getJobOutput` メソッドを実行し、ジョブの出力 (ボールトインベントリデータ) をダウンロードします。

   `GetJobOutputRequest` クラスのインスタンスを作成して、アカウント ID、ジョブ ID、およびボールト名を指定します。アカウント ID を指定しなかった場合には、リクエストに署名する際に指定した認証情報に関連付けられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3 」を参照してください。

   S3 Glacier により返される出力は `GetJobOutputResult` オブジェクトで使用できます。

```
GetJobOutputRequest jobOutputRequest = new GetJobOutputRequest()
        .withVaultName("*** provide vault name ***")
        .withJobId("*** provide job ID ***");
GetJobOutputResult jobOutputResult = client.getJobOutput(jobOutputRequest);
// jobOutputResult.getBody(); provides the output stream.
```

> **Note**
>
> 基本となるジョブ関連の REST API の詳細については、「ジョブのオペレーション」を参照してください。

例: Amazon SDK for Java を使用してボールトインベントリを取得する

次の Java コード例では、指定されたボールトのボールトインベントリを取得します。

この例では次のタスクを実行しています。

- Amazon Simple Notification Service (Amazon SNS) のトピックの作成

  S3 Glacier は、ジョブの完了後、このトピックに通知を送信します。
- Amazon Simple Queue Service (Amazon SQS) キューの作成

  この例では、ポリシーをキューにアタッチして、Amazon SNS トピックでメッセージをキューに投稿できるようにします。
- 指定したアーカイブをダウンロードするジョブを開始します。

  ジョブのリクエストでは、ジョブの完了後に S3 Glacier がトピックへの通知を発行できるように、作成した Amazon SNS トピックを指定しています。
- Amazon SQS キューにジョブ ID を含むメッセージがあるかどうかを確認します。

  メッセージがある場合は、JSON を解析し、ジョブが正常に完了したかどうかを確認します。正常に完了している場合は、アーカイブをダウンロードします。
- Amazon SNS トピックおよび作成された Amazon SQS キューを削除して、クリーンアップします。

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParseException;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
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.SQSActions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.GetJobOutputRequest;
import com.amazonaws.services.glacier.model.GetJobOutputResult;
import com.amazonaws.services.glacier.model.InitiateJobRequest;
import com.amazonaws.services.glacier.model.InitiateJobResult;
import com.amazonaws.services.glacier.model.JobParameters;
import com.amazonaws.services.sns.AmazonSNSClient;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
import com.amazonaws.services.sns.model.DeleteTopicRequest;
import com.amazonaws.services.sns.model.SubscribeRequest;
import com.amazonaws.services.sns.model.SubscribeResult;
import com.amazonaws.services.sns.model.UnsubscribeRequest;
import com.amazonaws.services.sqs.AmazonSQSClient;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.CreateQueueResult;
import com.amazonaws.services.sqs.model.DeleteQueueRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;


public class AmazonGlacierDownloadInventoryWithSQSPolling {

    public static String vaultName = "*** provide vault name ***";
    public static String snsTopicName = "*** provide topic name ***";
    public static String sqsQueueName = "*** provide queue name ***";
```

```
    public static String sqsQueueARN;
    public static String sqsQueueURL;
    public static String snsTopicARN;
    public static String snsSubscriptionARN;
    public static String fileName = "*** provide file name ***";
    public static String region = "*** region ***";
    public static long sleepTime = 600;
    public static AmazonGlacierClient client;
    public static AmazonSQSClient sqsClient;
    public static AmazonSNSClient snsClient;

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

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier." + region + ".amazonaws.com");
        sqsClient = new AmazonSQSClient(credentials);
        sqsClient.setEndpoint("https://sqs." + region + ".amazonaws.com");
        snsClient = new AmazonSNSClient(credentials);
        snsClient.setEndpoint("https://sns." + region + ".amazonaws.com");

        try {
            setupSQS();

            setupSNS();

            String jobId = initiateJobRequest();
            System.out.println("Jobid = " + jobId);

            Boolean success = waitForJobToComplete(jobId, sqsQueueURL);
            if (!success) { throw new Exception("Job did not complete
successfully."); }

            downloadJobOutput(jobId);

            cleanUp();

        } catch (Exception e) {
            System.err.println("Inventory retrieval failed.");
            System.err.println(e);
        }
    }
```

```
    private static void setupSQS() {
        CreateQueueRequest request = new CreateQueueRequest()
            .withQueueName(sqsQueueName);
        CreateQueueResult result = sqsClient.createQueue(request);
        sqsQueueURL = result.getQueueUrl();

        GetQueueAttributesRequest qRequest = new GetQueueAttributesRequest()
            .withQueueUrl(sqsQueueURL)
            .withAttributeNames("QueueArn");

        GetQueueAttributesResult qResult = sqsClient.getQueueAttributes(qRequest);
        sqsQueueARN = qResult.getAttributes().get("QueueArn");

        Policy sqsPolicy =
            new Policy().withStatements(
                    new Statement(Effect.Allow)
                    .withPrincipals(Principal.AllUsers)
                    .withActions(SQSActions.SendMessage)
                    .withResources(new Resource(sqsQueueARN)));
        Map<String, String> queueAttributes = new HashMap<String, String>();
        queueAttributes.put("Policy", sqsPolicy.toJson());
        sqsClient.setQueueAttributes(new SetQueueAttributesRequest(sqsQueueURL,
 queueAttributes));

    }
    private static void setupSNS() {
        CreateTopicRequest request = new CreateTopicRequest()
            .withName(snsTopicName);
        CreateTopicResult result = snsClient.createTopic(request);
        snsTopicARN = result.getTopicArn();

        SubscribeRequest request2 = new SubscribeRequest()
            .withTopicArn(snsTopicARN)
            .withEndpoint(sqsQueueARN)
            .withProtocol("sqs");
        SubscribeResult result2 = snsClient.subscribe(request2);

        snsSubscriptionARN = result2.getSubscriptionArn();
    }
    private static String initiateJobRequest() {

        JobParameters jobParameters = new JobParameters()
            .withType("inventory-retrieval")
            .withSNSTopic(snsTopicARN);
```

```
        InitiateJobRequest request = new InitiateJobRequest()
            .withVaultName(vaultName)
            .withJobParameters(jobParameters);

        InitiateJobResult response = client.initiateJob(request);

        return response.getJobId();
    }

    private static Boolean waitForJobToComplete(String jobId, String sqsQueueUrl)
throws InterruptedException, JsonParseException, IOException {

        Boolean messageFound = false;
        Boolean jobSuccessful = false;
        ObjectMapper mapper = new ObjectMapper();
        JsonFactory factory = mapper.getFactory();

        while (!messageFound) {
            List<Message> msgs = sqsClient.receiveMessage(
               new
ReceiveMessageRequest(sqsQueueUrl).withMaxNumberOfMessages(10)).getMessages();

            if (msgs.size() > 0) {
                for (Message m : msgs) {
                    JsonParser jpMessage = factory.createJsonParser(m.getBody());
                    JsonNode jobMessageNode = mapper.readTree(jpMessage);
                    String jobMessage = jobMessageNode.get("Message").textValue();

                    JsonParser jpDesc = factory.createJsonParser(jobMessage);
                    JsonNode jobDescNode = mapper.readTree(jpDesc);
                    String retrievedJobId = jobDescNode.get("JobId").textValue();
                    String statusCode = jobDescNode.get("StatusCode").textValue();
                    if (retrievedJobId.equals(jobId)) {
                        messageFound = true;
                        if (statusCode.equals("Succeeded")) {
                            jobSuccessful = true;
                        }
                    }
                }

            } else {
              Thread.sleep(sleepTime * 1000);
            }
```

```
          }
        return (messageFound && jobSuccessful);
    }

    private static void downloadJobOutput(String jobId) throws IOException {

        GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
            .withVaultName(vaultName)
            .withJobId(jobId);
        GetJobOutputResult getJobOutputResult =
 client.getJobOutput(getJobOutputRequest);

        FileWriter fstream = new FileWriter(fileName);
        BufferedWriter out = new BufferedWriter(fstream);
        BufferedReader in = new BufferedReader(new
 InputStreamReader(getJobOutputResult.getBody()));
        String inputLine;
        try {
            while ((inputLine = in.readLine()) != null) {
                out.write(inputLine);
            }
        }catch(IOException e) {
            throw new AmazonClientException("Unable to save archive", e);
        }finally{
            try {in.close();}  catch (Exception e) {}
            try {out.close();}  catch (Exception e) {}
        }
        System.out.println("Retrieved inventory to " + fileName);
    }

    private static void cleanUp() {
        snsClient.unsubscribe(new UnsubscribeRequest(snsSubscriptionARN));
        snsClient.deleteTopic(new DeleteTopicRequest(snsTopicARN));
        sqsClient.deleteQueue(new DeleteQueueRequest(sqsQueueURL));
    }
}
```

Amazon S3 Glacier で AWS SDK for .NETを使用してボールトインベントリをダウンロードする

以下に、 AWS SDK for .NETの低レベル API を使用してボールトインベントリを取得する手順を示します。高レベル API では、ボールトインベントリの取得はサポートされていません。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   ボールトが存在する AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `InitiateJob` メソッドを実行してインベントリの取得ジョブを開始します。

   `InitiateJobRequest`オブジェクトでジョブ情報を指定します。Amazon S3 Glacier (S3 Glacier) は、レスポンスとしてジョブ ID を返します。レスポンスは、`InitiateJobResponse` クラスのインスタンスで使用できます。

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);

InitiateJobRequest initJobRequest = new InitiateJobRequest()
{
  VaultName = vaultName,
  JobParameters = new JobParameters()
  {
    Type = "inventory-retrieval",
    SNSTopic = "*** Provide Amazon SNS topic arn ***",
  }
};
InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
string jobId = initJobResponse.JobId;
```

3. ジョブが完了するまで待ちます。

   ジョブの出力をダウンロードする準備が整うまで待つ必要があります。ボールトの通知設定により Amazon Simple Notification Service (Amazon SNS) トピックを指定している場合、またはジョブを開始したときに Amazon SNS トピックを指定している場合は、ジョブの完了後に S3 Glacier によりそのトピックにメッセージが送信されます。以下のセクションに示しているコード例では、Amazon SNS を使用して、S3 Glacier でメッセージを発行します。

   また、`DescribeJob` メソッドを呼び出して S3 Glacier にポーリングすることで、ジョブの完了ステータスを調べることもできます。ただし、通知のために Amazon SNS トピックを使用することをお勧めします。

4. `GetJobOutput` メソッドを実行し、ジョブの出力 (ボールトインベントリデータ) をダウンロードします。

`GetJobOutputRequest` クラスのインスタンスを作成して、アカウント ID、ボールト名、およびジョブ ID 情報を指定します。アカウント ID を指定しなかった場合には、リクエストに署名する際に使用した認証情報に関連付けられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for .NET での の使用 Amazon S3 」を参照してください。

S3 Glacier により返される出力は `GetJobOutputResponse` オブジェクトで使用できます。

```
GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
{
  JobId = jobId,
  VaultName = vaultName
};

GetJobOutputResponse getJobOutputResponse =
 client.GetJobOutput(getJobOutputRequest);
using (Stream webStream = getJobOutputResponse.Body)
{
   using (Stream fileToSave = File.OpenWrite(fileName))
   {
     CopyStream(webStream, fileToSave);
   }
}
```

> **Note**
>
> 基本となるジョブ関連の REST API の詳細については、「ジョブのオペレーション」を参照してください。

例: の低レベル API を使用したボールトインベントリの取得 AWS SDK for .NET

次の C# コード例では、指定されたボールトのボールトインベントリを取得します。

この例では次のタスクを実行しています。

- Amazon SNS トピックを設定します。

  S3 Glacier は、ジョブの完了後、このトピックに通知を送信します。

- Amazon SQS キューを設定する。

  この例では、ポリシーをキューにアタッチして、Amazon SNS トピックでメッセージを投稿できるようにします。

- 指定したアーカイブをダウンロードするジョブを開始します。

  この例では、ジョブのリクエストとして、S3 Glacier によりジョブの完了後にメッセージが送信されるように Amazon SNS トピックを指定します。

- Amazon SQS キューにメッセージがあるかどうかを定期的に確認します。

  メッセージがある場合は、JSON を解析し、ジョブが正常に完了したかどうかを確認します。正常に完了している場合は、アーカイブをダウンロードします。コード例では、JSON.NET ライブラリ (JSON.NET 参照) を使用して JSON を解析しています。

- Amazon SNS トピックおよび作成された Amazon SQS キューを削除して、クリーンアップします。

Example

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;
using Amazon.SQS;
using Amazon.SQS.Model;
using Newtonsoft.Json;

namespace glacier.amazon.com.docsamples
{
  class VaultInventoryJobLowLevelUsingSNSSQS
  {
    static string topicArn;
    static string queueUrl;
    static string queueArn;
    static string vaultName = "*** Provide vault name ***";
```

```
    static string fileName  = "*** Provide file name and path where to store inventory
***";
    static AmazonSimpleNotificationServiceClient snsClient;
    static AmazonSQSClient sqsClient;
    const string SQS_POLICY =
        "{" +
        "    \"Version\" : \"2012-10-17\"," +
        "    \"Statement\" : [" +
        "        {" +
        "            \"Sid\" : \"sns-rule\"," +
        "            \"Effect\" : \"Allow\"," +
        "            \"Principal\" : {\"AWS\" : \"arn:aws:iam::123456789012:root\" },"
+
        "            \"Action\"    : \"sqs:SendMessage\"," +
        "            \"Resource\"  : \"{QuernArn}\"," +
        "            \"Condition\" : {" +
        "                \"ArnLike\" : {" +
        "                    \"aws:SourceArn\" : \"{TopicArn}\"" +
        "                }" +
        "            }" +
        "        }" +
        "    ]" +
        "}";

    public static void Main(string[] args)
    {
      AmazonGlacierClient client;
      try
      {
        using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
            Console.WriteLine("Setup SNS topic and SQS queue.");
            SetupTopicAndQueue();
            Console.WriteLine("To continue, press Enter"); Console.ReadKey();

            Console.WriteLine("Retrieve Inventory List");
            GetVaultInventory(client);
        }
        Console.WriteLine("Operations successful.");
        Console.WriteLine("To continue, press Enter"); Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
```

```
      finally
      {
       // Delete SNS topic and SQS queue.
       snsClient.DeleteTopic(new DeleteTopicRequest() { TopicArn = topicArn });
       sqsClient.DeleteQueue(new DeleteQueueRequest() { QueueUrl = queueUrl });
      }
    }

    static void SetupTopicAndQueue()
    {
      long ticks = DateTime.Now.Ticks;

      // Setup SNS topic.
      snsClient = new
 AmazonSimpleNotificationServiceClient(Amazon.RegionEndpoint.USWest2);
      sqsClient = new AmazonSQSClient(Amazon.RegionEndpoint.USWest2);

      topicArn = snsClient.CreateTopic(new CreateTopicRequest { Name =
 "GlacierDownload-" + ticks }).TopicArn;
      Console.Write("topicArn: "); Console.WriteLine(topicArn);

      CreateQueueRequest createQueueRequest =  new CreateQueueRequest();
      createQueueRequest.QueueName = "GlacierDownload-" + ticks;
      CreateQueueResponse createQueueResponse =
 sqsClient.CreateQueue(createQueueRequest);
      queueUrl = createQueueResponse.QueueUrl;
      Console.Write("QueueURL: "); Console.WriteLine(queueUrl);

      GetQueueAttributesRequest getQueueAttributesRequest = new
 GetQueueAttributesRequest();
      getQueueAttributesRequest.AttributeNames = new List<string> { "QueueArn" };
      getQueueAttributesRequest.QueueUrl = queueUrl;
      GetQueueAttributesResponse response =
 sqsClient.GetQueueAttributes(getQueueAttributesRequest);
      queueArn = response.QueueARN;
      Console.Write("QueueArn: ");Console.WriteLine(queueArn);

      // Setup the Amazon SNS topic to publish to the SQS queue.
      snsClient.Subscribe(new SubscribeRequest()
      {
        Protocol = "sqs",
        Endpoint = queueArn,
        TopicArn = topicArn
      });
```

```
      // Add the policy to the queue so SNS can send messages to the queue.
      var policy = SQS_POLICY.Replace("{TopicArn}", topicArn).Replace("{QuernArn}",
queueArn);

      sqsClient.SetQueueAttributes(new SetQueueAttributesRequest()
      {
          QueueUrl = queueUrl,
          Attributes = new Dictionary<string, string>
          {
              { QueueAttributeName.Policy, policy }
          }
      });

    }

    static void GetVaultInventory(AmazonGlacierClient client)
    {
      // Initiate job.
      InitiateJobRequest initJobRequest = new InitiateJobRequest()
      {
        VaultName = vaultName,
        JobParameters = new JobParameters()
        {
          Type = "inventory-retrieval",
          Description = "This job is to download a vault inventory.",
          SNSTopic = topicArn,
        }
      };

      InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
      string jobId = initJobResponse.JobId;

      // Check queue for a message and if job completed successfully, download
inventory.
      ProcessQueue(jobId, client);
    }

    private static void ProcessQueue(string jobId, AmazonGlacierClient client)
    {
      ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest()
{ QueueUrl = queueUrl, MaxNumberOfMessages = 1 };
      bool jobDone = false;
      while (!jobDone)
```

```
      {
        Console.WriteLine("Poll SQS queue");
        ReceiveMessageResponse receiveMessageResponse =
sqsClient.ReceiveMessage(receiveMessageRequest);
        if (receiveMessageResponse.Messages.Count == 0)
        {
          Thread.Sleep(10000 * 60);
          continue;
        }
        Console.WriteLine("Got message");
        Message message = receiveMessageResponse.Messages[0];
        Dictionary<string, string> outerLayer =
JsonConvert.DeserializeObject<Dictionary<string, string>>(message.Body);
        Dictionary<string, object> fields =
JsonConvert.DeserializeObject<Dictionary<string, object>>(outerLayer["Message"]);
        string statusCode = fields["StatusCode"] as string;

        if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_SUCCEEDED,
StringComparison.InvariantCultureIgnoreCase))
        {
          Console.WriteLine("Downloading job output");
          DownloadOutput(jobId, client); // Save job output to the specified file
location.
        }
        else if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_FAILED,
StringComparison.InvariantCultureIgnoreCase))
          Console.WriteLine("Job failed... cannot download the inventory.");

        jobDone = true;
        sqsClient.DeleteMessage(new DeleteMessageRequest() { QueueUrl = queueUrl,
ReceiptHandle = message.ReceiptHandle });
      }
    }

    private static void DownloadOutput(string jobId, AmazonGlacierClient client)
    {
      GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
      {
        JobId = jobId,
        VaultName = vaultName
      };

      GetJobOutputResponse getJobOutputResponse =
client.GetJobOutput(getJobOutputRequest);
```

```
      using (Stream webStream = getJobOutputResponse.Body)
      {
        using (Stream fileToSave = File.OpenWrite(fileName))
        {
          CopyStream(webStream, fileToSave);
        }
      }
    }

    public static void CopyStream(Stream input, Stream output)
    {
      byte[] buffer = new byte[65536];
      int length;
      while ((length = input.Read(buffer, 0, buffer.Length)) > 0)
      {
        output.Write(buffer, 0, length);
      }
    }
  }
}
```

REST API を使用してボールトインベントリをダウンロードする

REST API を使用してボールトインベントリをダウンロードするには

ボールトインベントリのダウンロードは、2 ステップのプロセスです。

1. `inventory-retrieval` タイプのジョブを開始します。詳細については、「ジョブの開始 (ジョブの POST)」を参照してください。
2. ジョブが完了したら、インベントリデータをダウンロードします。詳細については、「ジョブの出力の取得 (GET output)」を参照してください。

を使用した Amazon S3 Glacier でのボールトインベントリのダウンロード AWS Command Line Interface

AWS Command Line Interface (AWS CLI) を使用して Amazon S3 Glacier (S3 Glacier) でボールトインベントリをダウンロードする手順は、次のとおりです。

トピック

- (前提条件) のセットアップ AWS CLI

- 例: を使用したボールトインベントリのダウンロード AWS CLI

(前提条件) のセットアップ AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

   のインストール AWS Command Line Interface

   の設定 AWS Command Line Interface

2. コマンドプロンプトで次のコマンドを入力して、 AWS CLI セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを使用してください。

     ```
     aws help
     ```

   - 設定したアカウントの S3 Glacier ボールトのリストを取得するには、`list-vaults` コマンドを使用します。*123456789012* 「」を AWS アカウント ID に置き換えます。

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - の現在の設定データを表示するには AWS CLI、 `aws configure list` コマンドを使用します。

     ```
     aws configure list
     ```

例: を使用したボールトインベントリのダウンロード AWS CLI

1. インベントリ取得ジョブを開始するには、`initiate-job` コマンドを使用します。

   ```
   aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
   job-parameters='{"Type": "inventory-retrieval"}'
   ```

   正常な出力:

```
{
    "location": "/111122223333/vaults/awsexamplevault/jobs/*** jobid ***",
    "jobId": "*** jobid ***"
}
```

2. 以前の取得ジョブのステータスをチェックするには、describe-job コマンドを使用します。

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

正常な出力:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "*** vault arn ***",
    "Completed": false,
    "JobId": "*** jobid ***",
    "Action": "InventoryRetrieval",
    "CreationDate": "*** job creation date ***",
    "StatusCode": "InProgress"
}
```

3. ジョブが完了するまで待ちます。

   ジョブの出力をダウンロードする準備が整うまで待つ必要があります。ジョブ ID は、S3 Glacier がジョブを完了してから少なくとも 24 時間は有効です。ボールトに通知設定を指定している場合、またはジョブを開始したときに Amazon Simple Notification Service (Amazon SNS) トピックを指定している場合は、ジョブの完了後に S3 Glacier からそのトピックにメッセージが送信されます。

   ボールトに特定のイベントに対する通知設定を指定できます。詳細については、「Amazon S3 Glacier でのボールト通知の設定」を参照してください。S3 Glacier は、特定のイベントが発生するたびに、指定された SNS トピックにメッセージを送信します。

4. 完了したら、get-job-output コマンドを使用して、取得ジョブをファイル output.json にダウンロードします。

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output.json
```

このコマンドは、次のフィールドを含むファイルを生成します。

```
{
"VaultARN":"arn:aws:glacier:region:111122223333:vaults/awsexamplevault",
"InventoryDate":"*** job completion date ***",
"ArchiveList":[
{"ArchiveId":"*** archiveid ***",
"ArchiveDescription":"*** archive description (if set) ***",
"CreationDate":"*** archive creation date ***",
"Size":"*** archive size (in bytes) ***",
"SHA256TreeHash":"*** archive hash ***"
}
{"ArchiveId":
...
]}
```

Amazon S3 Glacier でのボールト通知の設定

Amazon S3 Glacier からのボールトのアーカイブやボールトインベントリなどの取得は 2 ステップのプロセスです。

1. 取得ジョブを開始します。
2. ジョブが完了したら、ジョブの出力をダウンロードします。

ボールトの通知 設定で、ジョブが完了したときに Amazon Simple Notification Service (Amazon SNS) トピックにメッセージが送信されるように設定できます。

トピック

- でのボールト通知の設定:S3 Glacier 一般的な概念
- Amazon S3 Glacier で AWS SDK for Javaを使用してボールト通知を設定する
- Amazon S3 Glacier で AWS SDK for .NETを使用してボールト通知を設定する
- S3 Glacier で REST API を使用してボールト通知を設定する

- S3 Glacier コンソールを使用したボールト通知の設定
- AWS Command Line Interfaceコンソールを使用したボールト通知の設定

でのボールト通知の設定:S3 Glacier 一般的な概念

S3 Glacier の取り出しジョブリクエストは非同期的に実行されます。出力を取得するには、S3 Glacier がジョブを完了するまで待機する必要があります。ジョブのステータスを確認するために定期的に S3 Glacier にポーリングできますが、これは最適な方法ではありません。 では通知もサポートされています。S3 Glacier は通知もサポートしています。ジョブの完了時に、そのジョブで Amazon Simple Notification Service (Amazon SNS) トピックにメッセージを投稿できます。この機能を使用するには、ボールトの通知設定を指定する必要があります。設定では、イベントが発生したときに S3 Glacier でメッセージを送信する 1 つ以上のイベントおよび Amazon SNS トピックを指定します。

S3 Glacier では、ボールトの通知設定に追加できるジョブの完了に関連するイベント (`ArchiveRetrievalCompleted`、`InventoryRetrievalCompleted`) を定義しています。特定のジョブが完了すると、S3 Glacier は SNS トピックに通知メッセージを発行します。

通知設定は以下の例に示すように JSON ドキュメントです。

```
{
   "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic",
   "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

1 つのボールトに対して設定できる Amazon SNS トピックは 1 つのみです。

> **Note**
>
> ボールトに通知設定を追加すると、S3 Glacier では、通知設定で指定されたイベントが発生するたびに通知を送信します。オプションで、各ジョブの開始リクエストで Amazon SNS トピックを指定することもできます。ボールトの通知設定を追加し、さらに、ジョブの開始リクエストで Amazon SNS トピックを指定した場合、S3 Glacier は両方の通知を送信します。

S3 Glacier が送信するジョブの完了メッセージには、ジョブのタイプ (`InventoryRetrieval`、`ArchiveRetrieval`)、ジョブの完了ステータス、SNS ト

ピック名、ジョブのステータスコード、ボールト ARN などの情報が含まれます。次に、`InventoryRetrieval` ジョブの完了後に S3 Glacier が SNS トピックに送信した通知の例を示します。

```
{
 "Action": "InventoryRetrieval",
 "ArchiveId": null,
 "ArchiveSizeInBytes": null,
 "Completed": true,
 "CompletionDate": "2012-06-12T22:20:40.790Z",
 "CreationDate": "2012-06-12T22:20:36.814Z",
 "InventorySizeInBytes":11693,
 "JobDescription": "my retrieval job",
 "JobId":"HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID",
 "SHA256TreeHash":null,
 "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic",
 "StatusCode":"Succeeded",
 "StatusMessage": "Succeeded",
 "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
}
```

`Completed` フィールドが true の場合は、`StatusCode` を確認して、ジョブが正常に完了したか、または失敗したかを確認する必要もあります。

> **Note**
>
> Amazon SNS トピックでは、ボールトで通知を発行できるようにする必要があります。デフォルトでは、Amazon SNS トピックの所有者のみがトピックにメッセージを発行できます。ただし、Amazon SNS トピックとボールトが異なる によって所有されている場合は AWS アカウント、ボールトからの発行を受け入れるように Amazon SNS トピックを設定する必要があります。Amazon SNS トピックのポリシーは、Amazon SNS コンソール で設定できます。

Amazon SNS の詳細については、「Amazon SNS の開始方法」を参照してください。

Amazon S3 Glacier で AWS SDK for Javaを使用してボールト通知を設定する

以下では、 AWS SDK for Javaの低レベル API を使用してボールトに通知を設定する手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   ボールトが存在する AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `SetVaultNotificationsRequest` クラスのインスタンスを作成することにより、通知設定の情報を指定します。

   ボールト名、通知設定の情報、およびアカウント ID を指定する必要があります。通知設定の指定で、既存の Amazon SNS トピックの Amazon リソースネーム (ARN) と、通知する 1 つ以上のイベントを指定します。サポートされているイベントのリストについては、「ボールトの通知設定の指定 (PUT notification-configuration)」を参照してください。

3. リクエストオブジェクトをパラメータとして指定して、`setVaultNotifications` メソッドを実行します。

以下の Java コードスニペットは、前述の手順を示しています。このスニペットでは、ボールトに通知設定を設定します。この設定では、`ArchiveRetrievalCompleted` イベントまたは `InventoryRetrievalCompleted` イベントが発生したときに、指定した Amazon SNS トピックに通知を送信するように Amazon S3 Glacier(S3 Glacier) にリクエストします。

```
SetVaultNotificationsRequest request = new SetVaultNotificationsRequest()
        .withAccountId("-")
        .withVaultName("*** provide vault name ***")
        .withVaultNotificationConfig(
                new VaultNotificationConfig()
                .withSNSTopic("*** provide SNS topic ARN ***")
                .withEvents("ArchiveRetrievalCompleted", "InventoryRetrievalCompleted")
         );
client.setVaultNotifications(request);
```

> **Note**
>
> 基本となる REST API については、「ボールトオペレーション」を参照してください。

例: を使用してボールトの通知設定を設定する AWS SDK for Java

以下の Java コード例は、ボールトの通知設定を指定したうえでその設定を削除し、その後で削除した設定を復元するものです。以下の例を実行するための詳しい手順については、「Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3 」を参照してください。

Example

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.DeleteVaultNotificationsRequest;
import com.amazonaws.services.glacier.model.GetVaultNotificationsRequest;
import com.amazonaws.services.glacier.model.GetVaultNotificationsResult;
import com.amazonaws.services.glacier.model.SetVaultNotificationsRequest;
import com.amazonaws.services.glacier.model.VaultNotificationConfig;


public class AmazonGlacierVaultNotifications {

    public static AmazonGlacierClient client;
    public static String vaultName = "*** provide vault name ****";
    public static String snsTopicARN = "*** provide sns topic ARN ***";

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

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-east-1.amazonaws.com/");

        try {

            System.out.println("Adding notification configuration to the vault.");
            setVaultNotifications();
            getVaultNotifications();
```

```
            deleteVaultNotifications();

        } catch (Exception e) {
            System.err.println("Vault operations failed." + e.getMessage());
        }
    }

    private static void setVaultNotifications() {
        VaultNotificationConfig config = new VaultNotificationConfig()
            .withSNSTopic(snsTopicARN)
            .withEvents("ArchiveRetrievalCompleted", "InventoryRetrievalCompleted");

        SetVaultNotificationsRequest request = new SetVaultNotificationsRequest()
                .withVaultName(vaultName)
                .withVaultNotificationConfig(config);

        client.setVaultNotifications(request);
        System.out.println("Notification configured for vault: " + vaultName);
    }

    private static void getVaultNotifications() {
        VaultNotificationConfig notificationConfig = null;
        GetVaultNotificationsRequest request = new GetVaultNotificationsRequest()
                .withVaultName(vaultName);
        GetVaultNotificationsResult result = client.getVaultNotifications(request);
        notificationConfig = result.getVaultNotificationConfig();

        System.out.println("Notifications configuration for vault: "
                + vaultName);
        System.out.println("Topic: " + notificationConfig.getSNSTopic());
        System.out.println("Events: " + notificationConfig.getEvents());
    }

    private static void deleteVaultNotifications() {
            DeleteVaultNotificationsRequest request = new
 DeleteVaultNotificationsRequest()
                .withVaultName(vaultName);
            client.deleteVaultNotifications(request);
            System.out.println("Notifications configuration deleted for vault: " +
 vaultName);
    }
}
```

Amazon S3 Glacier で AWS SDK for .NETを使用してボールト通知を設定する

以下では、 AWS SDK for .NETの低レベル API を使用してボールトに通知を設定する手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   ボールトが存在する AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `SetVaultNotificationsRequest` クラスのインスタンスを作成することにより、通知設定の情報を指定します。

   ボールト名、通知設定の情報、およびアカウント ID を指定する必要があります。アカウント ID を指定しなかった場合には、リクエストに署名する際に使用した認証情報に関連付けられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for .NET での使用 Amazon S3 」を参照してください。

   通知設定の指定で、既存の Amazon SNS トピックの Amazon リソースネーム (ARN) と、通知する 1 つ以上のイベントを指定します。サポートされているイベントのリストについては、「ボールトの通知設定の指定 (PUT notification-configuration)」を参照してください。

3. リクエストオブジェクトをパラメータとして指定して、`SetVaultNotifications` メソッドを実行します。

4. ボールトに通知設定を設定した後は、クライアントにより提供される、`GetVaultNotifications` メソッドを呼び出して設定情報を取得することや、`DeleteVaultNotifications` メソッドを呼び出して設定情報を削除することができます。

例: を使用してボールトの通知設定を設定する AWS SDK for .NET

以下の C# コードの例は、前述の手順を示しています。この例では、US West (Oregon) Region のボールト ("`examplevault`") の通知 設定の設定、取得、および削除を行います。この設定では、`ArchiveRetrievalCompleted` イベントまたは `InventoryRetrievalCompleted` イベントが発生したときに、指定した Amazon SNS トピックに通知を送信するように Amazon S3 Glacier (S3 Glacier) にリクエストします。

> **Note**
>
> 基本となる REST API については、「ボールトオペレーション」を参照してください。

以下の例を実行するための詳しい手順については、「コード例の実行」を参照してください。ここに示したコードを更新し、既存のボールト名および Amazon SNS トピックを指定する必要があります。

Example

```
using System;
using System.Collections.Generic;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class VaultNotificationSetGetDelete
  {
    static string vaultName   = "examplevault";
    static string snsTopicARN = "*** Provide Amazon SNS topic ARN ***";

    static IAmazonGlacier client;

    public static void Main(string[] args)
    {
      try
      {
        using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
          Console.WriteLine("Adding notification configuration to the vault.");
          SetVaultNotificationConfig();
          GetVaultNotificationConfig();
          Console.WriteLine("To delete vault notification configuration, press Enter");
          Console.ReadKey();
          DeleteVaultNotificationConfig();
        }
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
```

```
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static void SetVaultNotificationConfig()
    {

      SetVaultNotificationsRequest request = new SetVaultNotificationsRequest()
      {
        VaultName = vaultName,
        VaultNotificationConfig = new VaultNotificationConfig()
        {
          Events   = new List<string>() { "ArchiveRetrievalCompleted",
"InventoryRetrievalCompleted" },
          SNSTopic = snsTopicARN
        }
      };
      SetVaultNotificationsResponse response = client.SetVaultNotifications(request);
    }

    static void GetVaultNotificationConfig()
    {
      GetVaultNotificationsRequest request = new GetVaultNotificationsRequest()
      {
        VaultName = vaultName,
        AccountId = "-"
      };
      GetVaultNotificationsResponse response = client.GetVaultNotifications(request);
      Console.WriteLine("SNS Topic ARN: {0}",
response.VaultNotificationConfig.SNSTopic);
      foreach (string s in response.VaultNotificationConfig.Events)
        Console.WriteLine("Event : {0}", s);
    }

    static void DeleteVaultNotificationConfig()
    {
      DeleteVaultNotificationsRequest request = new DeleteVaultNotificationsRequest()
      {
        VaultName = vaultName
      };
      DeleteVaultNotificationsResponse response =
client.DeleteVaultNotifications(request);
    }
```

```
  }
}
```

S3 Glacier で REST API を使用してボールト通知を設定する

REST API を使用してボールト通知を設定するには、「ボールトの通知設定の指定 (PUT notification-configuration)」を参照してください。さらに、ボールト通知の取得 (ボールト通知の取得 (GET notification-configuration)) およびボールト通知の削除 (ボールト通知の削除 (通知設定の削除)) を行うこともできます。

S3 Glacier コンソールを使用したボールト通知の設定

このセクションでは、Amazon S3 Glacier コンソールを使用してボールト通知を設定する方法について説明します。通知を設定するときに、Amazon Simple Notification Service (Amazon SNS) トピックに通知を送信するジョブの完了イベントを指定します。ボールト通知の設定に加えて、ジョブを開始するときに、通知を発行するトピックを指定することもできます。特定のイベントの通知を送信するようにボールトを設定し、ジョブの開始リクエストでも通知を指定すると、2 つの通知が送信されます。

ボールト通知を設定するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/glacier/home://www.com」で S3 Glacier コンソールを開きます。
2. 左のナビゲーションペインで、[ボールト] を選択します。
3. [ボールト] リストで、ボールトを選択します。
4. [通知] セクションに移動して、[編集] を選択します。
5. [イベント通知] ページで、 [通知をオンにします] を選択します。
6. [通知] セクションで、以下の Amazon Simple Notification Service (Amazon SNS) オプションのいずれかを選択し、対応する手順に従います。

Amazon SNS オプション	アクション
新しい SNS トピックの作成	1. [新しい SNS トピックの作成] を選択します。 2. [トピック名] に新しいトピックの名前を入力します。

Amazon SNS オプション	アクション
	トピック名は最大 256 文字です。英数字、ハイフン (-)、および下線 (_) を使用できます。トピック名は、アカウント および 内で一意である必要があります AWS リージョン。 3. (オプション) SMS メッセージを使用してトピックをサブスクライブする場合は、[表示名] に名前を入力します。 表示名には最大 100 文字使用できます。

Amazon SNS オプション	アクション
既存の SNS トピックを選択	1. [既存の SNS トピックを選択] を選択します。 2. [SNS トピックを指定] で、次のいずれかのオプションを選択します。 • SNS トピックから選択 [SNS トピック] ドロップダウンリストが表示されます。 ドロップダウンリストから既存のトピックを選択します。 • SNS トピック ARN を入力 [Amazon SNS トピック ARN] テキストボックスが表示されます。 SNS トピックの Amazon リソースネーム (ARN) を入力します。SNS トピック ARN の形式は次のとおりです。 `arn:aws:sns: region:account-id :topic-name` SNS トピック ARN は、Amazon SNS コンソールで確認できます。

7. [イベント] で、通知を送信するイベントを 1 つまたは両方選択します。
   - アーカイブの取得ジョブが完了したときのみに通知を送信するには、[アーカイブの取得ジョブの完了] を選択します。
   - ボールトインベントリジョブが完了したときのみに通知を送信するには、[ボールトインベントリの取得ジョブの完了] を選択します。

AWS Command Line Interfaceコンソールを使用したボールト通知の設定

このセクションでは、 AWS Command Line Interfaceを使用してボールト通知を設定する方法について説明します。通知を設定するときに、Amazon Simple Notification Service(Amazon (Amazon SNS) トピックへの通知をトリガーするジョブの完了イベントを指定します。ボールト通知の設定に加えて、ジョブを開始するときに、通知を発行するトピックを指定することもできます。特定のイベントに対して通知を行うようにボールトを設定し、ジョブの開始リクエストにも通知を指定すると、2 つの通知が送信されます。

次の手順に従って、 AWS CLIを使用してボールト通知を設定します。

トピック

- (前提条件) のセットアップ AWS CLI
- 例: を使用してボールト通知を設定する AWS CLI

(前提条件) のセットアップ AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

   のインストール AWS Command Line Interface

   の設定 AWS Command Line Interface

2. コマンドプロンプトで次のコマンドを入力して、 AWS CLI セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを使用してください。

     ```
     aws help
     ```

   - 設定したアカウントの S3 Glacier ボールトのリストを取得するには、`list-vaults` コマンドを使用します。*123456789012* 「」を AWS アカウント ID に置き換えます。

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - の現在の設定データを表示するには AWS CLI、 `aws configure list` コマンドを使用します。

```
aws configure list
```

例: を使用してボールト通知を設定する AWS CLI

1. `set-vault-notifications` コマンドを使用して、特定のイベントがボールトに発生したときに送信される通知を設定します。デフォルトでは、通知は受信できません。

```
aws glacier set-vault-notifications --vault-name examplevault --account-
id 111122223333 --vault-notification-config file://notificationconfig.json
```

2. 通知設定は以下の例に示すように JSON ドキュメントです。

```
{
   "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic",
   "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

S3 Glacierでの Amazon SNS トピックの使用の詳細については、「S3 Glacier でのボールト通知の設定:一般的な概念」を参照してください。

Amazon SNS の詳細については、「Amazon SNS の開始方法」を参照してください。

Amazon S3 Glacier でボールトを削除する

Amazon S3 Glacier (S3 Glacier) では、最後にインベントリを計算した時点でボールト内にアーカイブがなく、また、最後にインベントリを計算してからボールトへの書き込みがない場合にのみ、ボールトを削除できます。アーカイブの削除の詳細については、「Amazon S3 Glacier でアーカイブを削除する」を参照してください。ボールトインベントリのダウンロードの詳細については、「Amazon S3 Glacier でボールトインベントリをダウンロードする」を参照してください。

> **Note**
>
> S3 Glacier では、各ボールトについて 24 時間ごとにインベントリを作成します。インベントリには最新の情報が反映されていないことがあるため、S3 Glacier では、ボールトが実際

に空であるかどうかを確認するために、最後にボールトインベントリが作成されてから書き込み オペレーションがあったかどうかを確認します。

> **Note**
>
> ボールトアーカイブの自動削除については、Amazon S3 Glacier でのボールトアーカイブの自動削除」を参照してください。

トピック

- を使用した Amazon S3 Glacier でのボールトの削除 AWS SDK for Java
- を使用した Amazon S3 Glacier でのボールトの削除 AWS SDK for .NET
- S3 Glacier で REST API を使用してボールトを削除する
- S3 Glacier コンソールを使用した空のボールトの削除
- を使用した Amazon S3 Glacier でのボールトの削除 AWS Command Line Interface

を使用した Amazon S3 Glacier でのボールトの削除 AWS SDK for Java

以下では、 AWS SDK for Javaの低レベル API を使用してボールトを削除する手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   ボールトを削除する AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。
2. `DeleteVaultRequest` クラスのインスタンスを作成することにより、リクエスト情報を指定します。

   ボールト名およびアカウント ID を指定する必要があります。アカウント ID を指定しなかった場合は、リクエストに署名する際に指定した認証情報に関連づけられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3 」を参照してください。
3. リクエストオブジェクトをパラメータとして指定して、`deleteVault` メソッドを実行します。

   Amazon S3 Glacier (S3 Glacier) は、ボールトが空の場合にのみボールトを削除します。詳細については、「ボールトの削除 (DELETE vault)」を参照してください。

以下の Java コードスニペットは、前述の手順を示しています。

```
try {
    DeleteVaultRequest request = new DeleteVaultRequest()
        .withVaultName("*** provide vault name ***");

    client.deleteVault(request);
    System.out.println("Deleted vault: " + vaultName);
} catch (Exception e) {
    System.err.println(e.getMessage());
}
```

> **Note**
>
> 基本となる REST API については、「ボールトの削除 (DELETE vault)」を参照してください。

例: を使用してボールトを削除する AWS SDK for Java

コード例については、「例: を使用したボールトの作成 AWS SDK for Java」を参照してください。この Java コード例では、ボールトの作成、削除など、基本的なボールトオペレーションを示しています。

を使用した Amazon S3 Glacier でのボールトの削除 AWS SDK for .NET

両方高レベル API と低レベル API.NET 用の Amazon SDK で提供されており、ボールトを削除する方法を提供します。

トピック

- の高レベル API を使用したボールトの削除 AWS SDK for .NET
- の低レベル API を使用したボールトの削除 AWS SDK for .NET

の高レベル API を使用したボールトの削除 AWS SDK for .NET

高レベル API の `ArchiveTransferManager` クラスには、ボールトの削除に使用できる `DeleteVault` メソッドが用意されています。

例: の高レベル API を使用してボールトを削除する AWS SDK for .NET

コード例については、「例: の高レベル API を使用したボールトオペレーション AWS SDK for .NET」を参照してください。この C# コード例では、ボールトの作成、削除など、基本的なボールトオペレーションを示しています。

の低レベル API を使用したボールトの削除 AWS SDK for .NET

以下に、 AWS SDK for .NETを使用してボールトを削除する手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   ボールトを削除する AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `DeleteVaultRequest` クラスのインスタンスを作成することにより、リクエスト情報を指定します。

   ボールト名およびアカウント ID を指定する必要があります。アカウント ID を指定しなかった場合は、リクエストに署名する際に指定した認証情報に関連づけられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for .NET での の使用 Amazon S3 」を参照してください。

3. リクエストオブジェクトをパラメータとして指定して、`DeleteVault` メソッドを実行します。

   Amazon S3 Glacier (S3 Glacier) は、ボールトが空の場合にのみボールトを削除します。詳細については、「ボールトの削除 (DELETE vault)」を参照してください。

以下の C# コードスニペットは、前述の手順を示しています。スニペットは、デフォルトの AWS リージョンに存在するボールトのメタデータ情報を取得します。

```
AmazonGlacier client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1);

DeleteVaultRequest request = new DeleteVaultRequest()
{
  VaultName = "*** provide vault name ***"
};

DeleteVaultResponse response = client.DeleteVault(request);
```

> **Note**
>
> 基本となる REST API については、「ボールトの削除 (DELETE vault)」を参照してください。

例: の低レベル API を使用してボールトを削除する AWS SDK for .NET

コード例については、「例: の低レベル API を使用したボールトオペレーション AWS SDK for .NET」を参照してください。この C# コード例では、ボールトの作成、削除など、基本的なボールトオペレーションを示しています。

S3 Glacier で REST API を使用してボールトを削除する

REST API を使用してボールトを削除する方法については、「ボールトの削除 (DELETE vault)」を参照してください。

S3 Glacier コンソールを使用した空のボールトの削除

> **Note**
>
> ボールトを削除する前に、ボールト内の既存のアーカイブをすべて削除する必要があります。これを行うには、REST API、、、または AWS Command Line Interface () を使用して AWS SDK for Java AWS SDK for .NETアーカイブの削除リクエストを行うコードを記述しますAWS CLI。アーカイブの削除の詳細については、「ステップ 5: S3 Glacier でボールトからアーカイブを削除する」を参照してください。

ボールトが空になったら、次の手順を使用してボールトを削除できます。

Amazon S3 Glacier コンソールを使用して空のボールトを削除する方法

1. にサインイン AWS Management Console し、S3 Glacier コンソールで S3 Glacier コンソールを開きます。
2. 「リージョンの選択」で、ボールトが存在する AWS リージョン を選択します。
3. 左のナビゲーションペインで、[ボールト] を選択します。
4. [ボールト] リストで、削除するボールトの名前の横にあるオプションボタンを選択してから、ページの上部にある [削除] を選択します。

5. [ボールトの削除] ダイアログボックスで、[削除] を選択してボールトを削除することを確認します。

> **Important**
> ボールトの削除は元に戻せません。

6. ボールトを削除したことを確認するには、[ボールト] リストを開き、削除したボールトの名前を入力します。ボールトが見つからなければ、削除は成功しています。

を使用した Amazon S3 Glacier でのボールトの削除 AWS Command Line Interface

AWS Command Line Interface (AWS CLI) を使用して、Amazon S3 Glacier(S3 Glacier) 内の空のボールトと空でないボールトを削除できます。

トピック

- (前提条件) のセットアップ AWS CLI
- 例: を使用して空のボールトを削除する AWS CLI
- 例: を使用して空でないボールトを削除する AWS CLI

(前提条件) のセットアップ AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

   のインストール AWS Command Line Interface

   の設定 AWS Command Line Interface

2. コマンドプロンプトで次のコマンドを入力して、 AWS CLI セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを使用してください。

```
aws help
```

- 設定したアカウントの S3 Glacier ボールトのリストを取得するには、`list-vaults` コマンドを使用します。*123456789012* 「」を AWS アカウント ID に置き換えます。

```
aws glacier list-vaults --account-id 123456789012
```

- の現在の設定データを表示するには AWS CLI、 `aws configure list` コマンドを使用します。

```
aws configure list
```

例: を使用して空のボールトを削除する AWS CLI

- アーカイブを含まないボールトを削除するには、`delete-vault` コマンドを使用します。
  -

```
aws glacier delete-vault --vault-name awsexamplevault --account-id 111122223333
```

例: を使用して空でないボールトを削除する AWS CLI

S3 Glacier では、最後にインベントリを計算した時点でボールト内にアーカイブがなく、また、最後にインベントリを計算してからボールトへの書き込みがない場合にのみ、ボールトを削除できます。空でないボールトの削除は 3 つのステップからなります。ボールトのインベントリレポートからアーカイブ ID を取得し、各アーカイブを削除してから、ボールトを削除します。

1. インベントリ取得ジョブを開始するには、`initiate-job` コマンドを使用します。

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
job-parameters '{"Type": "inventory-retrieval"}'
```

正常な出力:

```
{
    "location": "/111122223333/vaults/awsexamplevault/jobs/*** jobid ***",
    "jobId": "*** jobid ***"
}
```

2. 以前の取得ジョブのステータスをチェックするには、`describe-job` コマンドを使用します。

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

正常な出力:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "*** vault arn ***",
    "Completed": false,
    "JobId": "*** jobid ***",
    "Action": "InventoryRetrieval",
    "CreationDate": "*** job creation date ***",
    "StatusCode": "InProgress"
}
```

3. ジョブが完了するまで待ちます。

   ジョブの出力をダウンロードする準備が整うまで待つ必要があります。ボールトに通知設定を指定している場合、またはジョブを開始したときに Amazon Simple Notification Service (Amazon SNS) トピックを指定している場合は、ジョブの完了後に S3 Glacier からそのトピックにメッセージが送信されます。

   ボールトに特定のイベントに対する通知設定を指定できます。詳細については、「Amazon S3 Glacier でのボールト通知の設定」を参照してください。S3 Glacier は、特定のイベントが発生するたびに、指定された SNS トピックにメッセージを送信します。

4. 完了したら、`get-job-output` コマンドを使用して、取得ジョブをファイル `output.json` にダウンロードします。

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output.json
```

   このコマンドは、次のフィールドを含むファイルを生成します。

```
{
```

```
"VaultARN":"arn:aws:glacier:region:111122223333:vaults/awsexamplevault",
"InventoryDate":"*** job completion date ***",
"ArchiveList":[
{"ArchiveId":"*** archiveid ***",
"ArchiveDescription":*** archive description (if set) ***,
"CreationDate":"*** archive creation date ***",
"Size":"*** archive size (in bytes) ***",
"SHA256TreeHash":"*** archive hash ***"
}
{"ArchiveId":
...
]}
```

5. `delete-archive` コマンドを使用して、ボールトから各アーカイブを削除します。

```
aws glacier delete-archive --vault-name awsexamplevault --account-id 111122223333
 --archive-id "*** archiveid ***"
```

> **Note**
>
> アーカイブ ID がハイフンまたは他の特殊文字で始まる場合は、このコマンドを実行するためにアーカイブ ID を引用符で囲む必要があります。

6. `initiate-job` コマンドを使用して、新しいインベントリ取得ジョブを開始します。

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
job-parameters '{"Type": "inventory-retrieval"}'
```

7. 完了したら、`delete-vault` コマンドを使用して、アーカイブのないボールトを削除します。

```
aws glacier delete-vault --vault-name awsexamplevault --account-id 111122223333
```

S3 Glacier ボールトにタグを付ける

タグ形式で Amazon S3 Glacier ボールトに独自のメタデータを割り当てることができます。タグは、ボールトに対して定義するキーと値のペアです。タグの制限を含むタグ付けの基本情報については、「Amazon S3 Glacier リソースのタグ付け」を参照してください。

以下のトピックでは、ボールトのタグの追加、一覧表示、および削除の方法について説明します。

トピック

- Amazon S3 Glacier コンソールを使用してボールトにタグを付けます。
- を使用したボールトのタグ付け AWS CLI
- Amazon S3 Glacier API を使用したボールトへのタグ付け
- 関連するセクション

Amazon S3 Glacier コンソールを使用してボールトにタグを付けます。

次の手順で説明するように、S3 Glacier コンソールを使用してタグの追加、一覧表示、削除を行えます。

ボールトのタグを表示するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/glacier/home:// www.com」で S3 Glacier コンソールを開きます。
2. 「リージョンの選択」で、リージョンセレクター AWS リージョン から を選択します。
3. 左のナビゲーションペインで、[ボールト] を選択します。
4. [ボールト] リストで、ボールトを選択します。
5. [ボールトのプロパティ] タブを選択します。[タグ] セクションまでスクロールすると、ボールトに関連するタグが表示されます。

ボールトにタグを追加するには

1 つのボールトに最大 50 個のタグを関連付けることができます。ボールトに関連付けるタグには一意のタグキーが必要です。

タグ付けの制約の詳細については、「Amazon S3 Glacier リソースのタグ付け」を参照してください。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/glacier/home:// www.com」で S3 Glacier コンソールを開きます。
2. 「リージョンの選択」で、リージョンセレクター AWS リージョン から を選択します。
3. 左のナビゲーションペインで、[ボールト] を選択します。
4. [名前] リストで、タグを追加するボールトの名前を選択します。
5. [ボールトのプロパティ] タブを選択します。

6. [Tタグ] セクションで [追加] を選択します。タグの追加 ページが表示されます。
7. [タグを追加] ページの [キー] フィールドでタグキーを指定し、オプションとして [値] フィールドでタグ値を指定します。
8. [S変更の保存] をクリックします。

タグを編集するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/glacier/home:// www.com」で S3 Glacier コンソールを開きます。
2. 「リージョンの選択」で、リージョンセレクター AWS リージョン から を選択します。
3. 左のナビゲーションペインで、[ボールト] を選択します。
4. [ボールト] リストで、ボールト名を選択します。
5. [ボールトのプロパティ] タブを選択し、[タグ] セクションまで下にスクロールします。
6. [タグ] で、変更するタグの横にあるチェックボックスを選択し、[編集] を選択します。[タグを編集] ページが表示されます。
7. [キー] フィールドでタグキーを更新し、任意で [値] フィールドでタグ値を更新します。
8. [Save changes] (変更の保存) をクリックします。

ボールトからタグを削除するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/glacier/home:// www.com」で S3 Glacier コンソールを開きます。
2. 「リージョンの選択」で、リージョンセレクター AWS リージョン から を選択します。
3. 左のナビゲーションペインで、[ボールト] を選択します。
4. [ボールト] リストで、タグを削除するボールトの名前を選択します。
5. [ボールトのプロパティ] タブを選択します。[Tags] (タグ) セクションまでスクロールダウンします。
6. [タグ] で、削除するタグの横にあるチェックボックスを選択し、[削除] を選択します。
7. [タグを削除] ダイアログボックスが開きます。選択したタグを削除することを確認するには、[削除] を選択します。

を使用したボールトのタグ付け AWS CLI

AWS Command Line Interface () を使用してタグを追加、一覧表示、または削除するには、次の手順に従いますAWS CLI。

各タグはキーと値で構成されます。各ボールトは、最大 50 個のタグを持つことができます。

1. ボールトにタグを追加するには、`add-tags-to-vault` コマンドを使用します。

```
aws glacier add-tags-to-vault --vault-name examplevault --account-id 111122223333
 --tags id=1234,date=2020
```

   このボールトオペレーションの詳細については、「ボールトにタグを追加する」を参照してください。

2. ボールトにアタッチされるタグをすべて一覧表示するには、`list-tags-for-vault` コマンドを使用します。

```
aws glacier list-tags-for-vault --vault-name examplevault --account-id 111122223333
```

   このボールトオペレーションの詳細については、「ボールトのタグの一覧表示」を参照してください。

3. ボールトにアタッチされた一連のタグから 1 つ以上のタグを削除するには、`remove-tags-from-vault` コマンドを使用します。

```
aws glacier remove-tags-from-vault --vault-name examplevault --account-
id 111122223333 --tag-keys date
```

   このボールトオペレーションの詳細については、「ボールトからタグを削除する」を参照してください。

Amazon S3 Glacier API を使用したボールトへのタグ付け

S3 Glacier API を使用してタグの追加、一覧表示、削除を行うことができます。例については、次のドキュメントを参照してください。

ボールトにタグを追加する (POST タグの追加)

指定したボールトのタグを追加または更新します。

ボールトのタグの一覧表示 (GET タグ)

指定したボールトのタグを一覧表示します。

ボールトからタグを削除する (POST タグの削除)

指定したボールトからタグを削除します。

関連するセクション

- Amazon S3 Glacier リソースのタグ付け

S3 Glacier ボールトロック

以下のトピックでは、Amazon S3 Glacier でボールトをロックする方法とボールトロックポリシーの使用方法について説明します。

トピック

- ボールトロックの概要
- S3 Glacier API を使用したボールトのロック
- を使用したボールトのロック AWS Command Line Interface
- S3 Glacier コンソールを使用したボールトのロック

ボールトロックの概要

S3 Glacier のボールトロックでは、ボールトロックポリシーを使用して、S3 Glacier の各ボールトに対するコンプライアンス管理を簡単にデプロイして適用することができます。ボールトロックポリシーで「write once read many」(WORM) などの管理を指定してポリシーをロックし、今後編集できないようにします。

> **Important**
>
> ボールトロックポリシーがロックされると、そのポリシーを変更したり削除したりできなくなります。

S3 Glacier では、ボールトロックポリシーによって設定された管理を実施することにより、コンプライアンス目標の達成に役立てることができます。例えば、ボールトロックポリシーを使用してデータ保持を適用することができます。 AWS Identity and Access Management (IAM) ポリシー言語を使用して、ボールトロックポリシーにさまざまなコンプライアンスコントロールをデプロイできます。ボールトロックポリシーの詳細については、「ボールトロックポリシー」を参照してください。

ボールトロックポリシーは、ボールトアクセスポリシーとは異なります。どちらのポリシーも、ボールトへのアクセスを制御します。ただし、ボールトロックポリシーでは、ロックによって以後の変更を禁止することにより、コンプライアンス管理を強力に実施することができます。ボールトロックポリシーは、データアクセスに対する厳密な管理が求められることの多い、規制やコンプライアンス管理のデプロイに使用できます。

> **Important**
>
> 最初にボールトを作成し、ボールトロックポリシーを完成させた後、アーカイブをボールトにアップロードしてポリシーを適用することをお勧めします。

対照的に、ボールトアクセスポリシーでは、コンプライアンスと関係がなく、一時的で、頻繁に変更が発生しやすいアクセス制御を実装します。ボールトロックポリシーとボールトアクセスポリシーは同時に使用できます。例えば、ボールトロックポリシーで時間ベースのデータ保持ルールを実装し (削除の拒否)、ボールトアクセスポリシーで、指定したサードパーティやビジネスパートナーに対して読み取り許可を付与することができます (読み取りの許可)。

ボールトのロックには、2 つのステップがあります。

1. ボールトロックポリシーをボールトに関連付けることによってロックを開始します。これにより、ロックが進行中状態になり、ロック ID が返されます。ポリシーが進行中状態にある間は、ロック ID の有効期限が切れるまでの 24 時間のうちにボールトロックポリシーを検証することができます。ボールトの進行中状態が終了しないようにするには、24 時間以内にボールドロック処理を完了する必要があります。完了しないと、ボールトロックポリシーは削除されます。
2. ロック ID を使用してロック処理を完了します。ボールトロックポリシーが想定どおりに機能しない場合は、ボールトロック処理を中止して最初からやり直すことができます。S3 Glacier API をボールトのロックに使用する方法については、「S3 Glacier API を使用したボールトのロック」を参照してください。

S3 Glacier API を使用したボールトのロック

Amazon S3 Glacier API を使用してボールトをロックするには、まず、デプロイする管理を指定するボールトロックポリシーを使用して ボールトロックの開始 (ロックポリシーの POST) を呼び出します。`Initiate Vault Lock` オペレーションにより、ポリシーがボールトにアタッチされ、ボールトロックが進行中状態になり、一意のロック ID が返されます。ボールトロックが進行中状態になった後は、`Initiate Vault Lock` コールで返されたロック ID を使用して ボールトロックの完了 (ロック ID の POST) を呼び出すことによりロックを完了する時間が 24 時間あります。

> **Important**
>
> - 最初にボールトを作成し、ボールトロックポリシーを完成させた後、アーカイブをボールトにアップロードしてポリシーを適用することをお勧めします。
> - ボールトロックポリシーは、一度ロックされると変更したり削除したりできません。

進行中状態になってから 24 時間以内にボールトロック処理を完了しないと、ボールトの進行中状態が自動的に終了し、ボールトロックポリシーが削除されます。`Initiate Vault Lock` を再度呼び出して新しいボールトロックポリシーをインストールすると、進行中状態にすることができます。

進行中状態では、ロックする前にボールトロックポリシーをテストすることができます。進行中状態のボールトロックポリシーは、ボールトロックの中止 (ロックポリシーの DELETE) を呼び出してポリシーを削除できる点を除き、ボールトをロックした場合のように完全に有効になります。ポリシーを微調整するには、必要に応じて `Abort Vault Lock` と `Initiate Vault Lock` の組み合わせを繰り返して、ボールトロックポリシーの変更を検証することができます。

ボールトロックポリシーを検証したら、最後に返されたロック ID を指定して ボールトロックの完了 (ロック ID の POST) を呼び出すことによって、ボールトロック処理を完了することができます。ボールトがロック状態になると、ボールトロックポリシーは変更できなくなり、`Abort Vault Lock` を呼び出して削除できなくなります。

関連するセクション

- ボールトロックポリシー
- ボールトロックの中止 (ロックポリシーの DELETE)
- ボールトロックの完了 (ロック ID の POST)

- ボールトロックの取得 (ロックポリシーの GET)
- ボールトロックの開始 (ロックポリシーの POST)

を使用したボールトのロック AWS Command Line Interface

を使用してボールトをロックできます AWS Command Line Interface。これにより、指定したボールトにボールトロックポリシーがインストールされ、ロック ID が返されます。ボールトロック処理は 24 時間以内に完了する必要があります。完了しない場合、ボールトロックポリシーはボールトから削除されます。

(前提条件) のセットアップ AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

   のインストール AWS Command Line Interface

   の設定 AWS Command Line Interface
2. コマンドプロンプトで次のコマンドを入力して、 AWS CLI セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを使用してください。

     ```
     aws help
     ```

   - 設定したアカウントの S3 Glacier ボールトのリストを取得するには、`list-vaults` コマンドを使用します。*123456789012* 「」を AWS アカウント ID に置き換えます。

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - の現在の設定データを表示するには AWS CLI、 `aws configure list` コマンドを使用します。

     ```
     aws configure list
     ```

1. `initiate-vault-lock` を使用してボールトロックポリシーをインストールし、ボールトロックのロック状態を `InProgress` に設定します。

```
aws glacier initiate-vault-lock --vault-name examplevault --account-id 111122223333
 --policy file://lockconfig.json
```

2. ロック設定は以下の例に示すように JSON ドキュメントです。このコマンドを使用する前に、*VAULT_ARN* と *######* をユースケースに適した値に置き換えてください。

   ロックするボールトの ARN を見つけるには、`list-vaults` コマンドを使用できます。

```
{"Policy":"{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":\"Define-vault-lock
\",\"Effect\":\"Deny\",\"Principal\":{\"AWS\":\"arn:aws:iam::111122223333:root\"},
\"Action\":\"glacier:DeleteArchive\",\"Resource\":\"VAULT_ARN\",\"Condition\":
{\"NumericLessThanEquals\":{\"glacier:ArchiveAgeinDays\":\"365\"}}}]}"}
```

3. ボールトロックを開始すると、`lockId` が返されるはずです。

```
{
    "lockId": "LOCK_ID"
}
```

ボールトロックを完了するには 24 時間以内に `complete-vault-lock` を実行する必要があります。完了しない場合、ボールトロックポリシーはボールトから削除されます。

```
aws glacier complete-vault-lock --vault-name examplevault --account-id 111122223333 --
lock-id LOCK_ID
```

関連するセクション

- 「AWS CLI コマンドリファレンス」の「initiate-vault-lock」
- 「AWS CLI コマンドリファレンス」の「list-vaults」
- 「AWS CLI コマンドリファレンス」の「complete-vault-lock」
- ボールトロックポリシー
- ボールトロックの中止 (ロックポリシーの DELETE)
- ボールトロックの完了 (ロック ID の POST)
- ボールトロックの取得 (ロックポリシーの GET)
- ボールトロックの開始 (ロックポリシーの POST)

S3 Glacier コンソールを使用したボールトのロック

Amazon S3 Glacier のボールトロックでは、ボールトロックポリシーを使用して、S3 Glacier の各ボールトに対するコンプライアンス管理を簡単にデプロイして適用することができます。S3 Glacier ボールトロックの詳細については、「ボールトロックポリシーによる Amazon S3 Glacier のアクセス制御」を参照してください。

> **Important**
>
> - 最初にボールトを作成し、ボールトロックポリシーを完成させた後、アーカイブをボールトにアップロードしてポリシーを適用することをお勧めします。
> - ボールトロックポリシーは、一度ロックされると変更したり削除したりできません。

S3 Glacier コンソールを使用してボールトでボールトロックポリシーを開始する方法

ボールトロックポリシーをボールトにアタッチすることによってロックを開始します。これにより、ロックが進行中状態になり、ロック ID が返されます。ポリシーが進行中状態にある間は、ロック ID の有効期限が切れるまでの 24 時間のうちにボールトロックポリシーを検証することができます。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/glacier/home://www.com」で S3 Glacier コンソールを開きます。
2. 「リージョンの選択」で、リージョンセレクター AWS リージョン から を選択します。
3. 左のナビゲーションペインで、[ボールト] を選択します。
4. [ボールト] ページで、[ボールトを作成] を選択します。
5. 新しいボールトを作成します。

   > **Important**
   >
   > 最初にボールトを作成し、ボールトロックポリシーを完成させた後、アーカイブをボールトにアップロードしてポリシーを適用することをお勧めします。

6. ボールト リストから新しいボールトを選択します。
7. [ボールトポリシー] タブを選択します。
8. [ボールトロックポリシー] セクションで、[ボールトロックポリシーを開始] を選択します。

9. [ボールトロックポリシーを開始] ページで、標準テキストボックスにテキスト形式でボールトロックポリシーのレコード保持の管理を指定します。

> **Note**
>
> ボールトロックポリシーでレコード保持管理をテキスト形式で指定し、`Initiate Vault Lock` API オペレーションを呼び出すか、S3 Glacier コンソールのインタラクティブ UI を使用してボールトロックを開始できます。ボールトロックポリシーの形式については、「Amazon S3 Glacier ボールトロックポリシーの例」を参照してください。

10. [変更の保存] をクリックします。
11. [ボールトロック ID を記録] ダイアログボックスで、ロック ID をコピーして安全な場所に保存します。

> **Important**
>
> ボールトロックポリシーが開始されたら、24 時間以内にポリシーを検証し、ロック処理を完了する必要があります。ロック処理を完了するには、ロック ID を指定する必要があります。24 時間以内に指定しない場合、ロック ID の有効期限が切れて、進行中のポリシーは削除されます。

12. ロック ID を安全な場所に保存したら、[閉じる] を選択します。
13. 24 時間以内にボールトロックポリシーをテストします。ポリシーが意図したとおりに機能している場合は、[ボールトロックポリシーを完了] を選択します。
14. [ボールトロックを完了] ダイアログボックスで、ボールトロックポリシー処理を完了すると元に戻せないことを確認するチェックボックスを選択します。
15. テキストボックスに指定した [ロック ID] を入力します。
16. [ボールトロックを完了] を選択します。

Amazon S3 Glacier でのアーカイブの操作

アーカイブとは、写真、動画、ドキュメントなど、ボールトに格納するオブジェクトを指します。これは、Amazon S3 Glacier (S3 Glacier) のストレージの基本単位です。各アーカイブには一意の ID とオプションの説明が割り当てられます。アーカイブをアップロードすると、S3 Glacier はアーカイブ ID を含むレスポンスを返します。このアーカイブ ID は、アーカイブが保存されている AWS リージョンで一意です。アーカイブ ID の例を次に示します。

```
TJgHcrOSfAkV6hdPqOATYfp_0ZaxL1pIBOc02iZ0gDPMr2ig-
nhwd_PafstsdIf6HSrjHnP-3p6LCJClYytFT_CBhT9CwNxbRaM5MetS3I-
GqwxI3Y8QtgbJbhEQPs0mJ3KExample
```

アーカイブ ID は 138 バイトです。アーカイブをアップロードする際に、オプションの説明を指定できます。ID を使用してアーカイブを取得することはできますが、説明を使用してアーカイブを取得することはできません。

> **Important**
>
> S3 Glacier は管理コンソールを提供します。コンソールを使用して、ボールトの作成および削除を行うことができます。ただし、それ以外の S3 Glacier の操作には、 AWS Command Line Interface (CLI) の使用またはコードの記述が必要になります。例えば、写真、動画、その他のドキュメントなどのデータをアップロードするには、 を使用するか、REST API を直接使用する AWS CLI か、Amazon SDKs を使用してリクエストを行うコードを記述する必要があります。で S3 Glacier を使用する方法の詳細については AWS CLI、AWS CLI S3 Glacier のリファレンス」を参照してください。をインストールするには AWS CLI、「」を参照してくださいAWS Command Line Interface。

トピック

- Amazon S3 Glacier でのアーカイブオペレーション
- クライアント側でのアーカイブのメタデータの管理
- Amazon S3 Glacier へのアーカイブのアップロード
- S3 Glacier でのアーカイブのダウンロード
- Amazon S3 Glacier でアーカイブを削除する

Amazon S3 Glacier でのアーカイブオペレーション

S3 Glacier では、基本的なアーカイブオペレーション (アップロード、ダウンロード、および削除) をサポートしています。アーカイブのダウンロードは、非同期オペレーションです。

Amazon S3 Glacier へのアーカイブのアップロード

アーカイブは、1 回のオペレーションまたはパート単位でアップロードできます。アーカイブをパート単位でアップロードするために使用する API 呼び出しはマルチパートアップロードと呼ばれます。詳細については、「Amazon S3 Glacier へのアーカイブのアップロード」を参照してください。

> **Important**
>
> S3 Glacier は管理コンソールを提供します。コンソールを使用して、ボールトの作成および削除を行うことができます。ただし、S3 Glacier との他のすべてのやり取りでは、 AWS Command Line Interface (CLI) を使用するか、コードを記述する必要があります。例えば、写真、動画、その他のドキュメントなどのデータをアップロードするには、 を使用するか、REST API を直接使用する AWS CLI か、Amazon SDKs を使用してリクエストを行うコードを記述する必要があります。で S3 Glacier を使用する方法の詳細については AWS CLI、AWS CLI S3 Glacier のリファレンス」を参照してください。をインストールするには AWS CLI、「」を参照してくださいAWS Command Line Interface。

Amazon S3 Glacier でのアーカイブ ID の検索

アーカイブ ID は、そのアーカイブを含むボールトのボールトインベントリをダウンロードすることで取得できます。ボールトインベントリのダウンロードの詳細については、「Amazon S3 Glacier でボールトインベントリをダウンロードする」を参照してください。

Amazon S3 Glacier でのアーカイブのダウンロード

アーカイブのダウンロードは、非同期オペレーションです。まず、特定のアーカイブをダウンロードするためのジョブを開始する必要があります。S3 Glacier はジョブのリクエストを受け取った後で、ダウンロードするアーカイブを準備します。ジョブが完了すると、アーカイブデータをダウンロードできます。このジョブは非同期であるため、ジョブの完了時に Amazon Simple Notification Service (Amazon SNS) トピックに通知を送信するように にリクエストすることができます。個々のジョブ

リクエストごとに SNS トピックを指定することや、特定のイベントが発生したときに通知を送信するようにボールトを設定することができます。アーカイブのダウンロードの詳細については、「S3 Glacier でのアーカイブのダウンロード」を参照してください。

Amazon S3 Glacier でのアーカイブの削除

S3 Glacier に用意されている API コールを使用して削除できるアーカイブは一度に 1 つです。詳細については、「Amazon S3 Glacier でアーカイブを削除する」を参照してください。

S3 Glacier でのアーカイブの更新

アーカイブをアップロードした後で、アーカイブのコンテンツや説明を更新することはできません。アーカイブのコンテンツや説明を更新する唯一の方法は、アーカイブを削除し、新しくアーカイブをアップロードすることです。アーカイブをアップロードするたびに、S3 Glacier により一意のアーカイブ ID が返されることに注意してください。

クライアント側でのアーカイブのメタデータの管理

オプションのアーカイブの説明以外に、S3 Glacier ではアーカイブに対してどのような追加のメタデータもサポートしていません。アーカイブのアップロード時に、S3 Glacier により ID が割り当てられます。ID はアーカイブに関するどのような情報も推察することができないように、意味のない文字列になっています。クライアント側でアーカイブに関するメタデータを管理することもできます。メタデータには、アーカイブ名と、アーカイブに関するその他の有益な情報を含めることができます。

> **Note**
>
> Amazon Simple Storage Service (Amazon S3) のユーザーなら、バケットにオブジェクトをアップロードすると、そのオブジェクトに `MyDocument.txt` や `SomePhoto.jpg` などのオブジェクトキーを割り当てることができることをご存知だと思います。S3 Glacier では、アップロードしたアーカイブにオブジェクトキーを割り当てることはできません。

クライアント側でアーカイブのメタデータを管理する場合、アーカイブのアップロード時に指定したアーカイブの説明とアーカイブ ID を含むボールトインベントリを S3 Glacier が管理していることに注意してください。アーカイブのメタデータを管理するクライアント側のデータベースの問題を調整するために、ボールトインベントリをダウンロードすることもできます。ただし、S3 Glacier はほぼ

毎日、ボールトインベントリを更新します。ボールトインベントリをリクエストすると、S3 Glacier は用意した最新のインベントリ (ポイントインタイムのスナップショット) を返します。

Amazon S3 Glacier へのアーカイブのアップロード

Amazon S3 Glacier (S3 Glacier) が備えている管理コンソールを使用して、ボールトの作成と削除を実行できます。ただし、管理コンソールを使用して S3 Glacier にアーカイブをアップロードすることはできません。写真、動画、その他のドキュメントなどのデータをアップロードするには、REST API を直接使用する AWS CLI か、Amazon SDKs を使用して、 を使用するか、コードを記述してリクエストを行う必要があります。

で S3 Glacier を使用する方法については AWS CLI、AWS CLI S3 Glacier のリファレンス」を参照してください。をインストールするには AWS CLI、「」を参照してくださいAWS Command Line Interface。以下の「アップロード」トピックでは、Amazon SDK for Java、Amazon SDK for .NET、REST API を使用して、アーカイブを S3 Glacier にアップロードする方法を説明します。

トピック

- Amazon S3 Glacier にアーカイブをアップロードするためのオプション
- 1 回のオペレーションでアーカイブをアップロードする
- パート単位での大きなアーカイブのアップロード (マルチパートアップロード)

Amazon S3 Glacier にアーカイブをアップロードするためのオプション

S3 Glacier には、アップロードするデータのサイズに応じた以下のオプションが用意されています。

- 単一操作によりアーカイブをアップロードする - 1 回のアップロードオペレーションでは、1 バイトから最大 4 GB までのサイズのアーカイブをアップロードできます。ただし、S3 Glacier のユーザーには 100 MB を超えるサイズのアーカイブをアップロードする際には、マルチパートアップロードを使用することをお勧めします。詳細については、「1 回のオペレーションでアーカイブをアップロードする」を参照してください。
- 複数のパートに分けてアーカイブをアップロードする - マルチパートアップロード API を使用すると、最大約 40,000 GB (10,000 x 4 GB) の大きなアーカイブをアップロードすることができます。

  マルチパートアップロード API 呼び出しは、大容量のアーカイブのアップロードを効率良く行えるように設計されています。アーカイブをいくつかのパートに分けてアップロードできます。パー

トは、任意の順序で独立かつ並列にアップロードされます。パートのアップロードが失敗した場合、アーカイブ全体ではなく、失敗したパートのみを再度アップロードするだけで済みます。マルチパートアップロードは、1 バイトから約 40,000 GB までのサイズのアーカイブに対して使用できます。詳細については、「パート単位での大きなアーカイブのアップロード (マルチパートアップロード)」を参照してください。

> **Important**
>
> S3 Glacier ボールトインベントリは 1 日 1 回のみ更新されます。アーカイブをアップロードしても、ボールトに追加された新しいアーカイブはすぐには、コンソールやダウンロード済みボールトインベントリのリストに表示されません。表示されるのは、ボールトインベントリが更新されてからになります。

AWS Snowball Edge のサービスの使用

AWS Snowball Edge は、インターネットをバイパスして、Amazon 所有のデバイス AWS の使用との間で大量のデータの移動を高速化します。詳細については、AWS Snowball Edge の詳細ページを参照してください。

既存のデータを Amazon S3 Glacier (S3 Glacier) にアップロードするには、いずれかの AWS Snowball Edge デバイスタイプを使用して Amazon S3 にデータをインポートしてから、ライフサイクルルールを使用してアーカイブのために S3 Glacier ストレージクラスに移行することを検討できます。Amazon S3 オブジェクトを S3 Glacier ストレージクラスに移行する際に、Amazon S3 は内部的に S3 Glacier を使用して、堅牢なストレージをより低コストで実現します。オブジェクトは S3 Glacier に保存されますが、引き続き Amazon S3 で管理する Amazon S3 オブジェクトであり、S3 Glacier を介して直接アクセスすることはできません。

Amazon S3 ライフサイクル設定と S3 Glacier ストレージクラスへのオブジェクト移行の詳細については、Amazon Simple Storage Service User Guide の「オブジェクトのライフサイクル 管理」と「オブジェクトの移行」を参照してください。

1 回のオペレーションでアーカイブをアップロードする

「Amazon S3 Glacier へのアーカイブのアップロード」で説明しているように、1 回のオペレーションで小さいサイズのアーカイブをアップロードできます。ただし、Amazon S3 Glacier (S3 Glacier) のユーザーには 100 MB を超えるサイズのアーカイブをアップロードする際には、マルチパートアップロードを使用することをお勧めします。

トピック

- を使用した 1 回のオペレーションでのアーカイブのアップロード AWS Command Line Interface
- を使用した 1 回のオペレーションでのアーカイブのアップロード AWS SDK for Java
- Amazon S3 Glacier の を使用して 1 回のオペレーション AWS SDK for .NET でアーカイブをアップロードする
- REST API を使用して 1 回のオペレーションでアーカイブをアップロードする

を使用した 1 回のオペレーションでのアーカイブのアップロード AWS Command Line Interface

AWS Command Line Interface () を使用して Amazon S3 Glacier (S3 Glacier) にアーカイブをアップロードできますAWS CLI。

トピック

- (前提条件) のセットアップ AWS CLI
- 例: を使用してアーカイブをアップロードする AWS CLI

(前提条件) のセットアップ AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

   のインストール AWS Command Line Interface

   の設定 AWS Command Line Interface

2. コマンドプロンプトで次のコマンドを入力して、 AWS CLI セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを使用してください。

     ```
     aws help
     ```

   - 設定したアカウントの S3 Glacier ボールトのリストを取得するには、`list-vaults` コマンドを使用します。*123456789012* 「」を AWS アカウント ID に置き換えます。

```
aws glacier list-vaults --account-id 123456789012
```

- の現在の設定データを表示するには AWS CLI、 `aws configure list` コマンドを使用します。

```
aws configure list
```

例: を使用してアーカイブをアップロードする AWS CLI

アーカイブをアップロードするには、ボールトを作成している必要があります。ボールトの作成方法の詳細については、「Amazon S3 Glacier でボールトを作成する」を参照してください。

1. `upload-archive` コマンドを使用して、既存のボールトにアーカイブを追加します。以下の例では、`vault name` を `account ID` と置き換えます。body パラメータには、アップロードするファイルへのパスを指定します。

```
aws glacier upload-archive --vault-name awsexamplevault --account-id 123456789012
 --body archive.zip
```

2. 正常な出力:

```
{
    "archiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
    "checksum": "969fb39823836d81f0cc028195fcdbcbbe76cdde932d4646fa7de5f21e18aa67",
    "location": "/123456789012/vaults/awsexamplevault/archives/
kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw"
}
```

終了すると、コマンドはアーカイブ ID、チェックサム、S3 Glacier 内の場所を出力します。upload-archive コマンドの詳細については、「AWS CLI コマンドリファレンス」の「upload-archive」を参照してください。

を使用した 1 回のオペレーションでのアーカイブのアップロード AWS SDK for Java

両方高レベル API と低レベル APIAmazon SDK for Java で提供されており、アーカイブをアップロードする方法を提供します。

トピック

- の高レベル API を使用したアーカイブのアップロード AWS SDK for Java
- の低レベル API を使用した 1 回のオペレーションでのアーカイブのアップロード AWS SDK for Java

の高レベル API を使用したアーカイブのアップロード AWS SDK for Java

高レベル API の `ArchiveTransferManager` クラスには、ボールトへのアーカイブのアップロードに使用できる `upload` メソッドが用意されています。

> **Note**
>
> `upload` メソッドを使用して、小さなアーカイブや大きなアーカイブをアップロードできます。このメソッドでは、アップロードするアーカイブのサイズに応じて、1 回のオペレーションでアップロードするか、マルチパートアップロード API を使用してアーカイブをパート単位でアップロードするかを決定します。

例: の高レベル API を使用したアーカイブのアップロード AWS SDK for Java

次の Java コード例では、米国西部 (オレゴン リージョン (`us-west-2`)のボールト (`examplevault`) にアーカイブをアップロードします。サポートされている AWS リージョンとエンドポイントのリストについては、「」を参照してくださいAmazon S3 Glacier へのアクセス。

この例を実行するための詳しい手順については、「Eclipse を使用した Amazon S3 Glacier の Java 実行例」を参照してください。ここに示したコードは、アップロードするボールトの名前とファイルの名前で更新する必要があります。

Example

```
import java.io.File;
import java.io.IOException;
```

```
import java.util.Date;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.transfer.ArchiveTransferManager;
import com.amazonaws.services.glacier.transfer.UploadResult;


public class ArchiveUploadHighLevel {
    public static String vaultName = "*** provide vault name ***";
    public static String archiveToUpload = "*** provide name of file to upload ***";

    public static AmazonGlacierClient client;

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


     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-west-2.amazonaws.com/");

        try {
            ArchiveTransferManager atm = new ArchiveTransferManager(client,
 credentials);

            UploadResult result = atm.upload(vaultName, "my archive " + (new Date()),
 new File(archiveToUpload));
            System.out.println("Archive ID: " + result.getArchiveId());

        } catch (Exception e)
        {
            System.err.println(e);
        }
    }
}
```

の低レベル API を使用した 1 回のオペレーションでのアーカイブのアップロード AWS SDK for Java

低レベル API には、アーカイブに関するあらゆるオペレーションのメソッドが用意されています。以下では、 AWS SDK for Javaを使用してアーカイブをアップロードするステップを説明します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

アーカイブをアップロードする AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `UploadArchiveRequest` クラスのインスタンスを作成することにより、リクエスト情報を指定します。

   アップロードするデータのほかにも、ペイロードのチェックサム (SHA-256 木構造ハッシュ)、ボールト名、データのコンテンツの長さ、およびアカウント ID を指定する必要があります。

   アカウント ID を指定しなかった場合には、リクエストに署名する際に使用した認証情報に関連付けられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3 」を参照してください。

3. リクエストオブジェクトをパラメータとして指定して、`uploadArchive` メソッドを実行します。

   レスポンスでは、Amazon S3 Glacier (S3 Glacier) によって新しくアップロードされたアーカイブのアーカイブ ID が返されます。

以下の Java コードスニペットは、前述の手順を示しています。

```
AmazonGlacierClient client;

UploadArchiveRequest request = new UploadArchiveRequest()
    .withVaultName("*** provide vault name ***")
    .withChecksum(checksum)
    .withBody(new ByteArrayInputStream(body))
    .withContentLength((long)body.length);

UploadArchiveResult uploadArchiveResult = client.uploadArchive(request);

System.out.println("Location (includes ArchiveID): " +
 uploadArchiveResult.getLocation());
```

例: の低レベル API を使用して 1 回のオペレーションでアーカイブをアップロードする AWS SDK for Java

次の Java コード例では AWS SDK for Java 、 を使用してアーカイブをボールト () にアップロードします`examplevault`。この例を実行するための詳しい手順については、「Eclipse を使用した Amazon S3 Glacier の Java 実行例」を参照してください。ここに示したコードは、アップロードするボールトの名前とファイルの名前で更新する必要があります。

```
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.TreeHashGenerator;
import com.amazonaws.services.glacier.model.UploadArchiveRequest;
import com.amazonaws.services.glacier.model.UploadArchiveResult;
public class ArchiveUploadLowLevel {

    public static String vaultName = "*** provide vault name ****";
    public static String archiveFilePath = "*** provide to file upload ****";
    public static AmazonGlacierClient client;

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

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-east-1.amazonaws.com/");

        try {
            // First open file and read.
            File file = new File(archiveFilePath);
            InputStream is = new FileInputStream(file);
            byte[] body = new byte[(int) file.length()];
            is.read(body);

            // Send request.
            UploadArchiveRequest request = new UploadArchiveRequest()
                .withVaultName(vaultName)
                .withChecksum(TreeHashGenerator.calculateTreeHash(new
 File(archiveFilePath)))
                .withBody(new ByteArrayInputStream(body))
                .withContentLength((long)body.length);

            UploadArchiveResult uploadArchiveResult = client.uploadArchive(request);

            System.out.println("ArchiveID: " + uploadArchiveResult.getArchiveId());
```

```
        } catch (Exception e)
        {
            System.err.println("Archive not uploaded.");
            System.err.println(e);
        }
    }
}
```

Amazon S3 Glacier の を使用して 1 回のオペレーション AWS SDK for .NET でアーカイブをアップロードする

両方高レベル API と低レベル API.NET 用の Amazon SDK で提供されているには、1 回のオペレーションでアーカイブをアップロードすることができます。

トピック

- の高レベル API を使用したアーカイブのアップロード AWS SDK for .NET
- の低レベル API を使用した 1 回のオペレーションでのアーカイブのアップロード AWS SDK for .NET

の高レベル API を使用したアーカイブのアップロード AWS SDK for .NET

高レベル API の `ArchiveTransferManager` クラスには、ボールトへのアーカイブのアップロードに使用できる `Upload` メソッドが用意されています。

> **Note**
>
> `Upload` メソッドを使用して、小さなファイルや大きなファイルをアップロードできます。このメソッドでは、アップロードするファイルのサイズに応じて、1 回のオペレーションでアップロードするか、マルチパートアップロード API を使用してファイルをパート単位でアップロードするかを決定します。

例: の高レベル API を使用したアーカイブのアップロード AWS SDK for .NET

次の C# コード例では、米国西部 (オレゴン リージョン のボールト (`examplevault`) にアーカイブをアップロードします。

この例を実行するための詳しい手順については、「コード例の実行」を参照してください。ここに示したコードは、アップロードするファイルの名前で更新する必要があります。

Example

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveUploadHighLevel
  {
    static string vaultName = "examplevault";
    static string archiveToUpload = "*** Provide file name (with full path) to upload
 ***";

    public static void Main(string[] args)
    {
       try
      {
         var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
         // Upload an archive.
         string archiveId = manager.Upload(vaultName, "upload archive test",
 archiveToUpload).ArchiveId;
         Console.WriteLine("Archive ID: (Copy and save this ID for use in other
 examples.) : {0}", archiveId);
         Console.WriteLine("To continue, press Enter");
         Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }
  }
}
```

の低レベル API を使用した 1 回のオペレーションでのアーカイブのアップロード AWS SDK for .NET

低レベル API には、アーカイブに関するあらゆるオペレーションのメソッドが用意されています。以下では、 AWS SDK for .NETを使用してアーカイブをアップロードするステップを説明します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   アーカイブをアップロードする AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。
2. `UploadArchiveRequest` クラスのインスタンスを作成することにより、リクエスト情報を指定します。

   アップロードするデータのほかにも、ペイロードのチェックサム (SHA-256 木構造ハッシュ)、ボールト名、およびアカウント ID を指定する必要があります。

   アカウント ID を指定しなかった場合には、リクエストに署名する際に使用した認証情報に関連付けられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for .NET での の使用 Amazon S3 」を参照してください。
3. リクエストオブジェクトをパラメータとして指定して、`UploadArchive` メソッドを実行します。

   レスポンスでは、S3 Glacier によって新しくアップロードされたアーカイブのアーカイブ ID が返されます。

例: の低レベル API を使用して 1 回のオペレーションでアーカイブをアップロードする AWS SDK for .NET

以下の C# コードの例は、前述の手順を示しています。この例では、 AWS SDK for .NET を使用してアーカイブをボールト () にアップロードします`examplevault`。

> **Note**
>
> 1 回のリクエストでアーカイブをアップロードする際に基盤となる REST API については、「アーカイブのアップロード (POST archive)」を参照してください。

この例を実行するための詳しい手順については、「コード例の実行」を参照してください。ここに示したコードは、アップロードするファイルの名前で更新する必要があります。

Example

```
using System;
using System.IO;
using Amazon.Glacier;
```

```
using Amazon.Glacier.Model;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveUploadSingleOpLowLevel
  {
    static string vaultName       = "examplevault";
    static string archiveToUpload = "*** Provide file name (with full path) to upload
 ***";

    public static void Main(string[] args)
    {
      AmazonGlacierClient client;
      try
      {
         using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
          Console.WriteLine("Uploading an archive.");
          string archiveId = UploadAnArchive(client);
          Console.WriteLine("Archive ID: {0}", archiveId);
        }
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static string UploadAnArchive(AmazonGlacierClient client)
    {
      using (FileStream fileStream = new FileStream(archiveToUpload, FileMode.Open,
 FileAccess.Read))
      {
        string treeHash = TreeHashGenerator.CalculateTreeHash(fileStream);
        UploadArchiveRequest request = new UploadArchiveRequest()
        {
          VaultName = vaultName,
          Body = fileStream,
          Checksum = treeHash
        };
        UploadArchiveResponse response = client.UploadArchive(request);
        string archiveID = response.ArchiveId;
```

```
        return archiveID;
      }
    }
  }
}
```

REST API を使用して 1 回のオペレーションでアーカイブをアップロードする

アーカイブのアップロード API コールを使用して、1 回のオペレーションでアーカイブをアップロードすることができます。詳細については、「アーカイブのアップロード (POST archive)」を参照してください。

パート単位での大きなアーカイブのアップロード (マルチパートアップロード)

トピック

- マルチパートアップロードのプロセス
- 基本情報
- AWS CLIを使用して大きなアーカイブをアップロードする
- Amazon SDK for Java を使用してパート単位で大きなアーカイブをアップロードする
- を使用した大きなアーカイブのアップロード AWS SDK for .NET
- REST API を使用してパート単位で大きなアーカイブをアップロードする

マルチパートアップロードのプロセス

「Amazon S3 Glacier へのアーカイブのアップロード」で説明しているように、Amazon S3 Glacier (S3 Glacier) をご利用のお客様には、100 メビバイト (MiB) を超えるサイズのアーカイブをアップロードする際にはマルチパートアップロードを使用することをお勧めします。

1. Initiate Multipart Upload

   マルチパートアップロードを開始するリクエストを送信すると、マルチパートアップロード ID が S3 Glacier から返されます。アップロード ID はマルチパートアップロードの一意の識別子です。後続のマルチパートアップロードオペレーションには、この ID が必要です。この ID は、S3 Glacier がジョブを完了してから少なくとも 24 時間は有効です。

   マルチパートアップロードの開始リクエストで、パートサイズをバイト数で指定します。アップロードするパートは、最後のパートを除き、すべてこのサイズになります。

> **Note**
>
> マルチパートアップロードを使用する際にアーカイブ全体のサイズを把握している必要はありません。つまり、アーカイブのアップロードを開始するときにアーカイブのサイズがわからない場合でも、マルチパートアップロードを使用できます。パートサイズを決定する必要があるのは、マルチパートアップロードの開始時のみです。

マルチパートアップロードの開始リクエストでは、オプションでアーカイブの説明を指定することもできます。

2. パートのアップロード

   パートのアップロードの各リクエストに、ステップ 1 で取得したマルチパートアップロード ID を含める必要があります。リクエストには、最終的なアーカイブ内でのパートの位置を特定するコンテンツ範囲をバイト単位で指定する必要もあります。S3 Glacier は後でコンテンツ範囲情報を使用して、アーカイブを適切な順序で組み立てます。アップロードする各パートのコンテンツ範囲を指定するため、&GL; は最終的にアセンブルされたアーカイブ内でのパートの位置を特定できます。そのため、任意の順序でパートをアップロードできます。このほか、複数のパートを並行してアップロードすることもできます。以前にアップロードしたパートと同じコンテンツ範囲を使って新しいパートをアップロードした場合、以前のパートは上書きされます。

3. マルチパートアップロードの完了 (または中止)

   アーカイブのパートをすべてアップロードしたら、完了オペレーションを使用します。ここでも、リクエストでアップロード ID を指定する必要があります。S3 Glacier は、指定したコンテンツ範囲に基づいて昇順に連結されたアーカイブを作成します。マルチパートアップロードの完了リクエストに対する S3 Glacier レスポンスには、新しく作成されたアーカイブのアーカイブ ID が含まれます。マルチパートアップロードの開始リクエストでオプションのアーカイブの説明を指定した場合は、S3 Glacier により、アセンブルされたアーカイブにそのアーカイブの説明が関連付けられます。マルチパートアップロードが正常に完了すると、マルチパートアップロード ID を参照できなくなります。つまり、マルチパートアップロード ID に関連付けられているパートにアクセスできなくなります。

   マルチパートアップロードを中止すると、そのマルチパートアップロード ID を使用してパートをアップロードできなくなります。中止されたマルチパートアップロードに関連付けられているパートによって消費されているストレージはすべて解放されます。パートのアップロードが進行

しているときにマルチパートアップロードを停止した場合は、停止後もそのパートのアップロードは成功または失敗する可能性があります。

その他のマルチパートアップロードオペレーション

Amazon S3 Glacier (S3 Glacier) には、次のマルチパートアップロード API コールが追加で用意されています。

- パートのリスト - このオペレーションを使用すると、特定のマルチパートアップロードのパートのリストを表示できます。マルチパートアップロードでアップロードしたパートに関する情報が返されます。パートのリストのリクエストごとに、S3 Glacier により最大 1,000 個のパートの情報が返されます。表示するマルチパートアップロードのパートがさらにある場合、結果がページ分割され、レスポンスにはリストの続きを表示するためのマーカーが返されます。後続のパートを取得するには、追加のリクエストを送信する必要があります。返されるパートのリストには、アップロードが完了していないパートは含まれていないことにご留意ください。
- マルチパートアップロードのリスト - このオペレーションを使用すると、進行中のマルチパートアップロードのリストを取得できます。進行中のマルチパートアップロードとは、開始されているものの、まだ完了または停止されていないアップロードを意味します。マルチパートアップロードのリストのリクエストごとに、S3 Glacier は最大 1,000 個のマルチパートアップロードが返します。表示するマルチパートアップロードがさらにある場合、結果がページ分割され、レスポンスにはリストの続きを表示するためのマーカーが返されます。残りのマルチパートアップロードを取得するには、追加のリクエストを送信する必要があります。

基本情報

次の表は、マルチパートアップロードの主な仕様をまとめたものです。

項目	仕様
アーカイブの最大サイズ	10,000 x 4 ギビバイト (GiB)
アップロードあたりの最大パート数	10,000
パートサイズ	1 MiB ~ 4 GiB、最後の部分は 1 MiB 未満にすることができます。サイズの値をバイト単位で指定します。

項目	仕様
	パートサイズは、メビバイト (1024 キビバイト [KiB]) に 2 の累乗を掛けた値でなければなりません。たとえば、1048576 (1 MiB)、2097152 (2 MiB)、4194304 (4 MiB)、8388608 (8 MiB) です。
パートのリストリクエストで返されるパートの最大数	1,000
マルチパートアップロードのリストリクエストで返されるマルチパートアップロードの最大数	1,000

AWS CLIを使用して大きなアーカイブをアップロードする

AWS Command Line Interface () を使用して、Amazon S3 Glacier (S3 Glacier) にアーカイブをアップロードできますAWS CLI。大きなアーカイブのアップロードエクスペリエンスを向上させるために、S3 Glacier にはマルチパートアップロードをサポートするいくつかの API オペレーションが用意されています。これらの API オペレーションを使用すると、アーカイブを分割してアップロードできます。パートは、任意の順序で独立かつ並列にアップロードされます。パートのアップロードが失敗した場合、アーカイブ全体ではなく、失敗したパートのみを再度アップロードするだけで済みます。マルチパートアップロードは、1 バイトから約 40,000 ギビバイト (GiB) までのサイズのアーカイブに対して使用できます。

S3 Glacier マルチパートアップロードの詳細については、「パート単位での大きなアーカイブのアップロード (マルチパートアップロード)」を参照してください。

トピック

- (前提条件) のセットアップ AWS CLI
- (前提条件) Python のインストール
- (前提条件) S3 Glacier ボールトの作成
- 例: を使用してパート単位で大きなアーカイブをアップロードする AWS CLI

(前提条件) のセットアップ AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

   のインストール AWS Command Line Interface

   の設定 AWS Command Line Interface

2. コマンドプロンプトで次のコマンドを入力して、 AWS CLI セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを使用してください。

     ```
     aws help
     ```

   - 設定したアカウントの S3 Glacier ボールトのリストを取得するには、`list-vaults` コマンドを使用します。*123456789012* 「」を AWS アカウント ID に置き換えます。

     ```
     aws glacier list-vaults --account-id 123456789012
     ```

   - の現在の設定データを表示するには AWS CLI、 `aws configure list` コマンドを使用します。

     ```
     aws configure list
     ```

(前提条件) Python のインストール

マルチパートアップロードを完了するには、アップロードするアーカイブの SHA256 木構造ハッシュを計算する必要があります。これは、アップロードするファイルの SHA256 木構造ハッシュを計算することとは異なります。アップロードするアーカイブの SHA256 木構造ハッシュを計算するには、Java、C# (.NET を使用)、または Python を使用できます。この例では、Python を使用します。Java または C# を使用する手順については、「チェックサムの計算」を参照してください。

Python のインストールの詳細については、「Boto3 デベロッパーガイド」の「Python のインストールまたは更新」を参照してください。

(前提条件) S3 Glacier ボールトの作成

次の例を使用するには、S3 Glacier ボールを少なくとも 1 つ作成しておく必要があります。ボールトの作成方法の詳細については、「Amazon S3 Glacier でボールトを作成する」を参照してください。

例: を使用してパート単位で大きなアーカイブをアップロードする AWS CLI

この例では、ファイルを作成し、マルチパートアップロード API オペレーションを使用してそのファイルをパート単位で Amazon S3 Glacierにアップロードします。

> **Important**
>
> この手順を開始する前に、前提条件となる手順をすべて実行しておくようにしてください。アーカイブをアップロードするには、ボールトを作成して AWS CLI を設定し、Java、C#、または Python を使用して SHA256 木構造ハッシュを計算できるように準備しておく必要があります。

次の手順では`initiate-multipart-upload`、、`upload-multipart-part`、および `complete-multipart-upload` AWS CLI コマンドを使用します。

これらのコマンドそれぞれの詳細については、「AWS CLI コマンドリファレンス」の「initiate-multipart-upload」、「upload-multipart-part」、「complete-multipart-upload」を参照してください。

1. initiate-multipart-upload コマンドを使用して、マルチパートアップロードリソースを作成します。リクエストで、パートサイズをバイト数で指定します。アップロードする各パートは、最後のパートを除き、すべてこのサイズになります。アップロードを開始する際にアーカイブ全体のサイズを把握している必要はありません。ただし、最後の手順でアップロードを完了するときには、各パートの合計サイズ (バイト単位) が必要になります。

   次のコマンドで、`--vault-name` パラメータと `--account-ID` パラメータの値を独自の情報に置き換えます。このコマンドは、ファイルごとに 1 メビバイト (MiB) (1024 x 1024 バイト) のパートサイズのアーカイブをアップロードするよう指定します。必要に応じてこの `--part-size` パラメータ値を置き換えます。

   ```
   aws glacier initiate-multipart-upload --vault-name awsexamplevault --part-
   size 1048576 --account-id 123456789012
   ```

   正常な出力:

```
{
"location": "/123456789012/vaults/awsexamplevault/multipart-uploads/uploadId",
"uploadId": "uploadId"
}
```

終了すると、コマンドはマルチパートアップロードリソースのアップロード ID と S3 Glacier 内の場所を出力します。後の手順で、このアップロード ID を使用します。

2. この例では、次のコマンドを使用して 4.4 MiB のファイルを作成し、1 MiB のチャンクに分割して、各チャンクをアップロードできます。独自のファイルをアップロードするには、データをチャンクに分割し、各パートをアップロードする、同様の手順に従います。

   Linux または macOS

   次のコマンドは、Linux または macOS 上に `file_to_upload` という名前の 4.4 MiB ファイルを作成します。

   ```
   mkfile -n 9000b file_to_upload
   ```

   Windows

   次のコマンドは、Windows 上に `file_to_upload` という名前の 4.4 MiB ファイルを作成します。

   ```
   fsutil file createnew file_to_upload 4608000
   ```

3. 次に、このファイルを 1 MiB のチャンクに分割します。

   ```
   split -b 1048576 file_to_upload chunk
   ```

   これで、次の 5 つのチャンクができます。最初の 4 つは 1 MiB で、最後の 1 つは約 400 キビバイト (KiB) です。

   ```
   chunkaa
   chunkab
   chunkac
   chunkad
   chunkae
   ```

4. upload-multipart-part コマンドを使用して、アーカイブの一部をアップロードします。アーカイブのパートは任意の順序でアップロードできます。パートを並行してアップロードすることもできます。マルチパートアップロードでは、最大 10,000 パートをアップロードできます。

   次のコマンドで、`--vault-name`、`--account-ID`、`--upload-id` のパラメータの値を置き換えます。アップロード ID は、`initiate-multipart-upload` コマンドの出力として指定された ID と一致する必要があります。`--range` パラメータは、サイズが 1 MiB (1024 x 1024 バイト) のパートをアップロードするよう指定します。このサイズは、`initiate-multipart-upload` コマンドで指定したサイズと一致する必要があります。必要に応じてこのサイズ値を調整します。`--body` パラメーターは、アップロードするパートの名前を指定します。

```
aws glacier upload-multipart-part --body chunkaa --range='bytes 0-1048575/*' --
vault-name awsexamplevault --account-id 123456789012 --upload-id upload_ID
```

   正常にアップロードできると、コマンドはアップロードされたパートのチェックサムを含む出力を生成します。

5. `upload-multipart-part` コマンドをもう一度実行して、マルチパートアップロードの残りのパートをアップロードします。アップロードするパートと一致するように、各コマンドの `--range` パラメータと `--body` パラメータの値を更新します。

```
aws glacier upload-multipart-part --body chunkab --range='bytes 1048576-2097151/*'
 --vault-name awsexamplevault --account-id 123456789012 --upload-id upload_ID
```

```
aws glacier upload-multipart-part --body chunkac --range='bytes 2097152-3145727/*'
 --vault-name awsexamplevault --account-id 123456789012 --upload-id upload_ID
```

```
aws glacier upload-multipart-part --body chunkad --range='bytes 3145728-4194303/*'
 --vault-name awsexamplevault --account-id 123456789012 --upload-id upload_ID
```

```
aws glacier upload-multipart-part --body chunkae --range='bytes 4194304-4607999/*'
 --vault-name awsexamplevault --account-id 123456789012 --upload-id upload_ID
```

> **Note**
>
> アップロードの最後のパートが 1 MiB 未満なので、最後のコマンドの `--range` パラメータ値は小さくなります。正常にアップロードできると、それぞれのコマンドはアップロードされた各パートのチェックサムを含む出力を生成します。

6. 次に、アーカイブを組み立てて、アップロードを終了します。アーカイブの合計サイズと SHA256 木構造ハッシュを含める必要があります。

   アーカイブの SHA256 木構造ハッシュを計算するには、Java、C#、または Python を使用できます。この例では、Python を使用します。Java または C# を使用する手順については、「チェックサムの計算」を参照してください。

   Python ファイル `checksum.py` を作成し、次のコードを挿入します。必要に応じて、元のファイルの名前を置き換えます。

```
from botocore.utils import calculate_tree_hash

checksum = calculate_tree_hash(open('file_to_upload', 'rb'))
print(checksum)
```

7. `checksum.py` を実行して SHA256 木構造ハッシュを計算します。次のハッシュは出力と一致しないことがあります。

```
$ python3 checksum.py
$ 3d760edb291bfc9d90d35809243de092aea4c47b308290ad12d084f69988ae0c
```

8. complete-multipart-upload コマンドを使用してアーカイブのアップロードを終了します。`--vault-name`、`--account-ID`、`--upload-ID`、`--checksum` のパラメータの値を置き換えます。`--archive` パラメータ値は、アーカイブの合計サイズをバイト単位で指定します。この値には、アップロードした個々のパートのすべてのサイズの合計値を指定する必要があります。必要に応じてこの値を置き換えます。

```
aws glacier complete-multipart-upload --archive-size 4608000 --vault-
name awsexamplevault --account-id 123456789012 --upload-id upload_ID --
checksum checksum
```

   終了すると、コマンドはアーカイブの ID、チェックサム、S3 Glacier 内の場所を出力します。

Amazon SDK for Java を使用してパート単位で大きなアーカイブをアップロードする

両方高レベル API と低レベル APIJava 版 Amazon SDK for Java で提供されている、大きなアーカイブをアップロードするためのメソッドがあります (「」Amazon S3 Glacier へのアーカイブのアップロード)。

- 高レベル API には、どのサイズのアーカイブのアップロードにも使用できるメソッドが用意されています。このメソッドは、アップロードするファイルに応じて、単一オペレーションでアーカイブをアップロードするか、Amazon S3 Glacier(S3 Glacier) のマルチパートアップロードのサポートを使用してパート単位でアーカイブをアップロードします。
- 低レベル API は、基本となる REST 実装にほぼ対応しています。つまり、1 回のオペレーションで小さいアーカイブをアップロードするメソッド、および大きなアーカイブに対してマルチパートアップロードをサポートするメソッドのグループが用意されています。このセクションでは、低レベル API を使用してパート単位で大きなアーカイブをアップロードする方法について説明します。

高レベル API と低レベル API の詳細については、「Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3 」を参照してください。

トピック

- の高レベル API を使用したパート単位での大きなアーカイブのアップロード AWS SDK for Java
- の低レベル API を使用して大規模なアーカイブをパート単位でアップロードする AWS SDK for Java

の高レベル API を使用したパート単位での大きなアーカイブのアップロード AWS SDK for Java

高レベル API の同じメソッドを使用して、小さいアーカイブまたは大きなアーカイブをアップロードします。高レベル API メソッドでは、アーカイブのサイズに基づいて、アーカイブを 1 回のオペレーションでアップロードするか、S3 Glacier に用意されているマルチパートアップロード API を使用するかを決定します。詳細については、「の高レベル API を使用したアーカイブのアップロード AWS SDK for Java」を参照してください。

の低レベル API を使用して大規模なアーカイブをパート単位でアップロードする AWS SDK for Java

アップロードを細かく制御するために、低レベル API を使用して、リクエストの設定やレスポンスの処理を行うことができます。以下に、 AWS SDK for Javaを使用してパート単位で大きなアーカイブをアップロードする手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   アーカイブを保存する AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `initiateMultipartUpload` メソッドを呼び出し、マルチパートアップロードを開始します。

   アーカイブのアップロード先となるボールト名、アーカイブのパートをアップロードするために使用するパートサイズ、およびオプションの説明を指定する必要があります。この情報は、`InitiateMultipartUploadRequest` クラスのインスタンスを作成することによって指定します。S3 Glacier により、レスポンスとしてアップロード ID が返されます。

3. `uploadMultipartPart` メソッドを呼び出し、パートをアップロードします。

   アップロードするパートごとに、ボールト名、このパートでアップロードされる最終的にアセンブルされたアーカイブ内のバイト範囲、パートデータのチェックサム、およびアップロード ID を指定する必要があります。

4. `completeMultipartUpload` メソッドを呼び出し、マルチパートアップロードを完了します。

   アップロード ID、アーカイブ全体のチェックサム、アーカイブのサイズ (アップロードしたすべてのパートを組み合わせたサイズ)、およびボールト名を指定する必要があります。S3 Glacier は、アップロードされたパートからアーカイブを構築し、アーカイブ ID を返します。

例: を使用したパート内の大きなアーカイブのアップロード AWS SDK for Java

次の Java コード例では、 AWS SDK for Java を使用してアーカイブをボールト () にアップロードします`examplevault`。この例を実行するための詳しい手順については、「Eclipse を使用した Amazon S3 Glacier の Java 実行例」を参照してください。ここに示したコードは、アップロードするファイルの名前で更新する必要があります。

> **Note**
>
> この例は、1 MB ~ 1 GB のパートサイズに対して有効です。ただし、S3 Glacier では 4 GB までのパートサイズをサポートしています。

Example

```
import java.io.ByteArrayInputStream;
import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.IOException;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Date;
import java.util.LinkedList;
import java.util.List;

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.TreeHashGenerator;
import com.amazonaws.services.glacier.model.CompleteMultipartUploadRequest;
import com.amazonaws.services.glacier.model.CompleteMultipartUploadResult;
import com.amazonaws.services.glacier.model.InitiateMultipartUploadRequest;
import com.amazonaws.services.glacier.model.InitiateMultipartUploadResult;
import com.amazonaws.services.glacier.model.UploadMultipartPartRequest;
import com.amazonaws.services.glacier.model.UploadMultipartPartResult;
import com.amazonaws.util.BinaryUtils;

public class ArchiveMPU {

    public static String vaultName = "examplevault";
    // This example works for part sizes up to 1 GB.
    public static String partSize = "1048576"; // 1 MB.
    public static String archiveFilePath = "*** provide archive file path ***";
    public static AmazonGlacierClient client;

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

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-west-2.amazonaws.com/");

        try {
            System.out.println("Uploading an archive.");
            String uploadId = initiateMultipartUpload();
            String checksum = uploadParts(uploadId);
            String archiveId = CompleteMultiPartUpload(uploadId, checksum);
            System.out.println("Completed an archive. ArchiveId: " + archiveId);

        } catch (Exception e) {
```

```
            System.err.println(e);
        }

    }

    private static String initiateMultipartUpload() {
        // Initiate
        InitiateMultipartUploadRequest request = new InitiateMultipartUploadRequest()
            .withVaultName(vaultName)
            .withArchiveDescription("my archive " + (new Date()))
            .withPartSize(partSize);

        InitiateMultipartUploadResult result = client.initiateMultipartUpload(request);

        System.out.println("ArchiveID: " + result.getUploadId());
        return result.getUploadId();
    }

    private static String uploadParts(String uploadId) throws AmazonServiceException,
 NoSuchAlgorithmException, AmazonClientException, IOException {

        int filePosition = 0;
        long currentPosition = 0;
        byte[] buffer = new byte[Integer.valueOf(partSize)];
        List<byte[]> binaryChecksums = new LinkedList<byte[]>();

        File file = new File(archiveFilePath);
        FileInputStream fileToUpload = new FileInputStream(file);
        String contentRange;
        int read = 0;
        while (currentPosition < file.length())
        {
            read = fileToUpload.read(buffer, filePosition, buffer.length);
            if (read == -1) { break; }
            byte[] bytesRead = Arrays.copyOf(buffer, read);

            contentRange = String.format("bytes %s-%s/*", currentPosition,
 currentPosition + read - 1);
            String checksum = TreeHashGenerator.calculateTreeHash(new
 ByteArrayInputStream(bytesRead));
            byte[] binaryChecksum = BinaryUtils.fromHex(checksum);
            binaryChecksums.add(binaryChecksum);
            System.out.println(contentRange);
```

```
            //Upload part.
            UploadMultipartPartRequest partRequest = new UploadMultipartPartRequest()
            .withVaultName(vaultName)
            .withBody(new ByteArrayInputStream(bytesRead))
            .withChecksum(checksum)
            .withRange(contentRange)
            .withUploadId(uploadId);

            UploadMultipartPartResult partResult =
 client.uploadMultipartPart(partRequest);
            System.out.println("Part uploaded, checksum: " + partResult.getChecksum());

            currentPosition = currentPosition + read;
        }
        fileToUpload.close();
        String checksum = TreeHashGenerator.calculateTreeHash(binaryChecksums);
        return checksum;
    }

    private static String CompleteMultiPartUpload(String uploadId, String checksum)
 throws NoSuchAlgorithmException, IOException {

        File file = new File(archiveFilePath);

        CompleteMultipartUploadRequest compRequest = new
 CompleteMultipartUploadRequest()
            .withVaultName(vaultName)
            .withUploadId(uploadId)
            .withChecksum(checksum)
            .withArchiveSize(String.valueOf(file.length()));

        CompleteMultipartUploadResult compResult =
 client.completeMultipartUpload(compRequest);
        return compResult.getLocation();
    }
}
```

を使用した大きなアーカイブのアップロード AWS SDK for .NET

両方高レベル API と低レベル API.NET 用の Amazon SDK で提供されているには、大きなアーカイブを部分的にアップロードするためのメソッドがあります (「」Amazon S3 Glacier へのアーカイブのアップロード)。

- 高レベル API には、どのサイズのアーカイブのアップロードにも使用できるメソッドが用意されています。このメソッドは、アップロードするファイルに応じて、単一オペレーションでアーカイブをアップロードするか、Amazon S3 Glacier (S3 Glacier) のマルチパートアップロードのサポートを使用してパート単位でアーカイブをアップロードします。
- 低レベル API は、基本となる REST 実装にほぼ対応しています。つまり、1 回のオペレーションで小さいアーカイブをアップロードするメソッド、および大きなアーカイブに対してマルチパートアップロードをサポートするメソッドのグループが用意されています。このセクションでは、低レベル API を使用してパート単位で大きなアーカイブをアップロードする方法について説明します。

高レベル API と低レベル API の詳細については、「Amazon S3 Glacier AWS SDK for .NET での の使用 Amazon S3 」を参照してください。

トピック

- の高レベル API を使用したパート単位での大きなアーカイブのアップロード AWS SDK for .NET
- の低レベル API を使用したパート単位での大きなアーカイブのアップロード AWS SDK for .NET

の高レベル API を使用したパート単位での大きなアーカイブのアップロード AWS SDK for .NET

高レベル API の同じメソッドを使用して、小さいアーカイブまたは大きなアーカイブをアップロードします。高レベル API メソッドでは、アーカイブのサイズに基づいて、アーカイブを 1 回のオペレーションでアップロードするか、S3 Glacier に用意されているマルチパートアップロード API を使用するかを決定します。詳細については、「の高レベル API を使用したアーカイブのアップロード AWS SDK for .NET」を参照してください。

の低レベル API を使用したパート単位での大きなアーカイブのアップロード AWS SDK for .NET

アップロードを細かくコントロールするために、低レベル API を使用して、リクエストの設定やレスポンスの処理を行うことができます。以下に、 AWS SDK for .NETを使用してパート単位で大きなアーカイブをアップロードする手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   アーカイブを保存する AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `InitiateMultipartUpload` メソッドを呼び出し、マルチパートアップロードを開始します。

アーカイブのアップロード先となるボールト名、アーカイブのパートをアップロードするために使用するパートサイズ、およびオプションの説明を指定する必要があります。この情報は、`InitiateMultipartUploadRequest` クラスのインスタンスを作成することによって指定します。S3 Glacier により、レスポンスとしてアップロード ID が返されます。

3. `UploadMultipartPart` メソッドを呼び出し、パートをアップロードします。

   アップロードするパートごとに、ボールト名、このパートでアップロードされる最終的にアセンブルされたアーカイブ内のバイト範囲、パートデータのチェックサム、およびアップロード ID を指定する必要があります。

4. `CompleteMultipartUpload` メソッドを呼び出し、マルチパートアップロードを完了します。

   アップロード ID、アーカイブ全体のチェックサム、アーカイブのサイズ (アップロードしたすべてのパートを組み合わせたサイズ)、およびボールト名を指定する必要があります。S3 Glacier は、アップロードされたパートからアーカイブを構築し、アーカイブ ID を返します。

例: .Amazon SDK for .NET を使用してパート単位で大きなアーカイブをアップロードする

次の C# コード例では AWS SDK for .NET 、 を使用してアーカイブをボールト () にアップロードしますexamplevault。この例を実行するための詳しい手順については、「コード例の実行」を参照してください。ここに示したコードは、アップロードするファイルの名前で更新する必要があります。

Example

```
using System;
using System.Collections.Generic;
using System.IO;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveUploadMPU
  {
    static string vaultName       = "examplevault";
    static string archiveToUpload = "*** Provide file name (with full path) to upload
 ***";
    static long partSize          = 4194304; // 4 MB.
```

```
    public static void Main(string[] args)
    {
      AmazonGlacierClient client;
      List<string> partChecksumList = new List<string>();
      try
      {
         using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
          Console.WriteLine("Uploading an archive.");
          string uploadId  = InitiateMultipartUpload(client);
          partChecksumList = UploadParts(uploadId, client);
          string archiveId = CompleteMPU(uploadId, client, partChecksumList);
          Console.WriteLine("Archive ID: {0}", archiveId);
        }
        Console.WriteLine("Operations successful. To continue, press Enter");
        Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static string InitiateMultipartUpload(AmazonGlacierClient client)
    {
      InitiateMultipartUploadRequest initiateMPUrequest = new
 InitiateMultipartUploadRequest()
      {

        VaultName = vaultName,
        PartSize = partSize,
        ArchiveDescription = "Test doc uploaded using MPU."
      };

      InitiateMultipartUploadResponse initiateMPUresponse =
 client.InitiateMultipartUpload(initiateMPUrequest);

      return initiateMPUresponse.UploadId;
    }

    static List<string> UploadParts(string uploadID, AmazonGlacierClient client)
    {
```

```
      List<string> partChecksumList = new List<string>();
      long currentPosition = 0;
      var buffer = new byte[Convert.ToInt32(partSize)];

      long fileLength = new FileInfo(archiveToUpload).Length;
      using (FileStream fileToUpload = new FileStream(archiveToUpload, FileMode.Open,
 FileAccess.Read))
      {
        while (fileToUpload.Position < fileLength)
        {
          Stream uploadPartStream = GlacierUtils.CreatePartStream(fileToUpload,
 partSize);
          string checksum = TreeHashGenerator.CalculateTreeHash(uploadPartStream);
          partChecksumList.Add(checksum);
          // Upload part.
          UploadMultipartPartRequest uploadMPUrequest = new
 UploadMultipartPartRequest()
          {

            VaultName = vaultName,
            Body = uploadPartStream,
            Checksum = checksum,
            UploadId = uploadID
          };
          uploadMPUrequest.SetRange(currentPosition, currentPosition +
 uploadPartStream.Length - 1);
          client.UploadMultipartPart(uploadMPUrequest);

          currentPosition = currentPosition + uploadPartStream.Length;
        }
      }
      return partChecksumList;
    }

    static string CompleteMPU(string uploadID, AmazonGlacierClient client, List<string>
 partChecksumList)
    {
      long fileLength = new FileInfo(archiveToUpload).Length;
      CompleteMultipartUploadRequest completeMPUrequest = new
 CompleteMultipartUploadRequest()
      {
        UploadId = uploadID,
        ArchiveSize = fileLength.ToString(),
        Checksum = TreeHashGenerator.CalculateTreeHash(partChecksumList),
```

```
        VaultName = vaultName
      };

      CompleteMultipartUploadResponse completeMPUresponse =
 client.CompleteMultipartUpload(completeMPUrequest);
      return completeMPUresponse.ArchiveId;
    }
  }
}
```

REST API を使用してパート単位で大きなアーカイブをアップロードする

「パート単位での大きなアーカイブのアップロード (マルチパートアップロード)」で説明しているように、マルチパートアップロードとは、アーカイブをパート単位でアップロードし、関連オペレーションを実行できる一連のオペレーションを指しています。これらのオペレーションの詳細については、次の API リファレンスのトピックを参照してください。

- マルチパートアップロードの開始 (POST multipart-uploads)
- パートのアップロード (PUT uploadID)
- マルチパートアップロードの完了 (POST uploadID)
- マルチパートアップロードの中止 (DELETE uploadID)
- パートのリスト (GET uploadID)
- マルチパートアップロードのリスト (GET multipart-uploads)

S3 Glacier でのアーカイブのダウンロード

Amazon S3 Glacier が備えている管理コンソールを使用して、ボールトの作成と削除を実行できます。ただし、管理コンソールを使用して S3 Glacier からアーカイブをダウンロードすることはできません。写真、動画、その他のドキュメントなどのデータをダウンロードするには、REST API を直接使用するか、 AWS SDKs を使用して、 AWS Command Line Interface （ AWS CLI) を使用するか、コードを記述してリクエストを行う必要があります。

で S3 Glacier を使用する方法については AWS CLI、AWS CLI S3 Glacier のリファレンス」を参照してください。をインストールするには AWS CLI、「」を参照してくださいAWS Command Line Interface。以下のトピックでは、 AWS SDK for Java、、および Amazon S3 Glacier REST API を使用して AWS SDK for .NETアーカイブを S3 Glacier にダウンロードする方法について説明します。Amazon S3

トピック

- S3 Glacier アーカイブの取得
- を使用した Amazon S3 Glacier でのアーカイブのダウンロード AWS SDK for Java
- Amazon S3 Glacier で AWS SDK for .NETを使用してアーカイブをダウンロードする
- Python での並列処理を使用した大規模なアーカイブのダウンロード
- REST API を使用したアーカイブのダウンロード
- を使用した Amazon S3 Glacier でのアーカイブのダウンロード AWS CLI

S3 Glacier アーカイブの取得

Amazon S3 Glacier からアーカイブから取り出すのは非同期オペレーションであり、最初にジョブを開始し、次にジョブが完了した後で出力をダウンロードします。アーカイブの取得ジョブを開始するには、、、または AWS SDKs ジョブの開始 (ジョブの POST) で REST API オペレーション AWS CLIまたは同等の を使用します。

トピック

- アーカイブの取り出しオプション
- アーカイブの取得範囲

S3 Glacier からアーカイブを取り出すプロセスは、2 つのステップに分かれます。このプロセスの概要を次に示します。

アーカイブを取り出すには

1. アーカイブの取得ジョブを開始します。

   a. 取得するアーカイブの ID を入手します。アーカイブ ID は、ボールトのインベントリから取得できます。アーカイブ ID は、REST API、 AWS CLI、または AWS SDKs を使用して取得できます。詳細については、「Amazon S3 Glacier でボールトインベントリをダウンロードする」を参照してください。

   b. ジョブの開始 (ジョブの POST) オペレーションを使用して、アーカイブの全体または一部を後にダウンロードするための準備をS3 Glacier にリクエストするジョブを開始します。

ジョブを開始すると、S3 Glacier ではレスポンスでジョブ ID を返し、ジョブを非同期的に実行します (ステップ 2 で説明したように、ジョブが完了するまではジョブの出力をダウンロードできません)。

> **Important**
>
> 標準取り出しの場合のみ、データ取り出しポリシーにより、`PolicyEnforcedException` 例外が発生して、`Initiate Job` リクエストが失敗することがあります。データ取り出しポリシーの詳細については、「S3 Glacier データ取得ポリシー」を参照してください。`PolicyEnforcedException` 例外の詳細については、「エラーレスポンス」を参照してください。

必要に応じて、S3 Glacier に保存されたデータの大きなセグメントを復元できます。S3 Glacier ストレージクラスからデータを復元する方法の詳細については、「Amazon Simple Storage Service ユーザーガイド」の「オブジェクトのアーカイブに適したストレージクラス」を参照してください。

2. ジョブが完了したら、ジョブの出力の取得 (GET output) オペレーションを使用してバイトをダウンロードします。

   全バイトをダウンロードすることも、バイト範囲を指定してジョブの出力の一部だけをダウンロードすることもできます。出力が大きい場合には、出力をチャンクに分けてダウンロードすると、ネットワーク障害など、ダウンロードに関する障害が発生したときに便利です。1 回のリクエストでジョブの出力を取得する場合に、ネットワーク障害が発生すると、最初から出力のダウンロードをやり直さなければならなくなります。これに対して、出力をチャンクに分けてダウンロードしていれば、障害が発生した場合でも、全体ではなく、出力の一部のダウンロードをやり直せば済みます。

S3 Glacier では、出力を取得する前にジョブを完了している必要があります。ジョブは、完了から少なくとも 24 時間は有効です。つまり、ジョブが完了してから 24 時間は出力をダウンロードできます。復元は、ジョブの完了から 24 時間後であればいつでも期限切れになります。ジョブが完了しているかどうかを判断するには、以下のオプションの 1 つを使用してジョブのステータスを確認します。

- ジョブの完了通知を待つ – ジョブの完了後に S3 Glacier が通知を投稿する Amazon Simple Notification Service (Amazon SNS) トピックを指定できます。S3 Glacier は、ジョブの完了後にのみ通知を送信します。

  ジョブを開始する際に、Amazon SNS トピックを指定できます。ジョブのリクエストで指定された Amazon SNS トピックのほか、ボールトにアーカイブの取り出しイベントに関する通知の設定がある場合には、S3 Glacier からその SNS トピックにも通知が発行されます。詳細については、「Amazon S3 Glacier でのボールト通知の設定」を参照してください。
- 明示的にジョブ情報をリクエストする - S3 Glacier `Describe Job` API オペレーション (ジョブの説明 (GET JobID)) を使用して、ジョブの情報を定期的にポーリングすることもできます。ただし、Amazon SNS 通知を使用することをお勧めします。

> **Note**
>
> Amazon SNS 通知を使用して取得する情報は、`Describe Job` API オペレーションを呼び出して取得する情報と同じです。

アーカイブの取り出しオプション

アーカイブの取り出しジョブを開始するときは、アクセス時間とコスト要件に基づいて、以下のいずれかの取り出しオプションを指定できます。取り出し料金については、「Amazon S3 Glacier の料金」を参照してください。

- 迅速 – 迅速取り出しを使用すると、アーカイブの復元に関する緊急のリクエストが臨時で必要になったときに、S3 Glacier Flexible Retrieval ストレージクラスまたは S3 Intelligent-Tiering Archive アクセス階層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 (250 MB 超) を除くすべてのアーカイブについては、迅速取り出しを使用して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。詳細については、「プロビジョンドキャパシティー」を参照してください。
- 標準 - 標準取り出しでは、数時間以内にすべてのアーカイブにアクセスできます。通常、標準取り出しは 3〜5 時間で完了します。標準は、取り出しオプションを指定しないで取り出しリクエストを行った場合にデフォルトで適用されます。
- 大容量 - 大容量取り出しは、S3 Glacier の最も安価な取り出しオプションであり、これを使用して大量のデータ (ペタバイトのデータを含む) を 1 日以内に低コストで取得できます。通常、大容量取り出しは 5〜12 時間で完了します。

次の表は、アーカイブの取り出しオプションをまとめたものです。料金については、「Amazon S3 Glacier の料金」を参照してください。

`Expedited`、、`Standard`または `Bulk`の取得を行うには、REST API RestoreObject オペレーション`Tier`リクエストのリクエスト要素を、必要なオプション、または AWS Command Line Interface (AWS CLI) または AWS SDKs の同等の に設定します。プロビジョンドキャパシティーを購入すると、すべての Expedited 取り出しはプロビジョンドキャパシティーを通じて自動的に提供されます。

プロビジョンドキャパシティー

プロビジョニングされたキャパシティーは、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。容量の各単位について 5 分ごとに 3 回以上の迅速取り出しを提供し、1 秒あたり最大 150 メガバイト (MBps) の取り出しスループットを提供します。

ワークロードからデータのサブセットにアクセスする際に非常に高い信頼性と予測可能性が求められる場合は、プロビジョニングされた取得容量を購入することをお勧めします。プロビジョンドキャパシティーがなくても、需要が異常に高い例外的な場合を除いては、通常は迅速取り出しが受け入れられます。ただし、環境を問わず、どのような場合でも迅速取り出しにアクセスするには、プロビジョニングされた取得容量を購入してください。

プロビジョニングされた容量の購入

プロビジョニングされたキャパシティユニットは、S3 Glacier コンソール、REST API プロビジョニングされた容量の購入 (POST provisioned-capacity) オペレーション、 AWS SDKs、または を使用して購入できます AWS CLI。プロビジョニングされた容量の料金情報については、「Amazon S3 Glacier の料金」を参照してください。

プロビジョニングされた容量単位は、購入日時から 1 か月間有効です。

開始日が 31 日の場合、有効期限は翌月の最終日となります。たとえば、開始日が 8 月 31 日の場合、有効期限は 9 月 30 日です。開始日が 1 月 31 日の場合、有効期限は 2 月 28 日です。

Amazon S3 Glacier コンソールを使用してプロビジョニングされたキャパシティーを購入する方法

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/glacier/home で S3 Glacier コンソールを開きます。
2. 左側のナビゲーションペインで、[データ取り出し設定] を選択します。
3. [プロビジョニングされたキャパシティユニット (PCU)] で [PCU の購入] を選択します。[PCU の購入] ダイアログボックスが表示されます。

4. プロビジョニングされたキャパシティーを購入する場合は、[購入を確認するには] ボックスに **confirm** と入力します。
5. [PCU の購入] を選択します。

アーカイブの取得範囲

S3 Glacier からアーカイブを取得するときには、取得するアーカイブの範囲 (部分) をオプションで指定することもできます。デフォルトでは、アーカイブの全体が取得されます。バイト範囲を指定すると、以下のことを行う場合に便利です。

- データのダウンロードの管理 - S3 Glacier では、取り出しリクエストが完了してから 24 時間、取得したデータをダウンロードできます。このため、アーカイブの一部だけを取得することによって、特定のダウンロード期間内のダウンロードのスケジュールを管理できます。
- サイズの大きなアーカイブの特定の一部のみ取得 - たとえば、以前に多くのファイルをまとめ、1 つのアーカイブとしてアップロードしたものの、その後、ファイルの一部のみを取得する必要が生じたとします。このような場合には、取得リクエストを 1 回使用して、必要なファイルが含まれるアーカイブから一定の範囲を指定できます。このほか、取得リクエストを複数回、1 回ごとに 1 つまたは複数のファイルから成る範囲を指定して送信する方法もあります。

範囲取得を使用して取得ジョブを開始した場合には、メガバイト単位に調整した範囲を指定する必要があります。つまり、バイト範囲の始点はゼロ (アーカイブの先頭) またはその後 1 MB 間隔 (1 MB、2 MB、3 MB など) の点を指定することができます。

レンジの終わりの値には、アーカイブの末尾、またはレンジの開始値より大きな任意の 1 MB 間隔の数値のいずれかを指定できます。このほか、(取得ジョブが完了した後で) データをダウンロードする際にチェックサムの値を取得する場合には、ジョブの開始時にリクエストする範囲が木構造ハッシュ可能になっている必要があります。チェックサムを使用すると、データが送信中に破損しなかったかどうか確認できます。メガバイト単位への調整と木構造ハッシュを可能にするための調整については、「データをダウンロードするときのチェックサムの受信」を参照してください。

を使用した Amazon S3 Glacier でのアーカイブのダウンロード AWS SDK for Java

両方高レベル API と低レベル APIAmazon SDK for Java で提供されているアーカイブをダウンロードする方法を提供します。

トピック

- の高レベル API を使用したアーカイブのダウンロード AWS SDK for Java
- の低レベル API を使用したアーカイブのダウンロード AWS SDK for Java

の高レベル API を使用したアーカイブのダウンロード AWS SDK for Java

高レベル API の `ArchiveTransferManager` クラスには、アーカイブのダウンロードに使用できる `download` メソッドが用意されています。

> **Important**
>
> -`ArchiveTransferManager`クラスは、Amazon Simple Notification Service (Amazon SNS) トピックと、そのトピックにサブスクライブされている Amazon Simple Queue Service (Amazon SQS) キューを作成します。その後、アーカイブの取り出しジョブを開始し、使用可能にするアーカイブを探してキューをポーリングします。アーカイブが使用可能になると、ダウンロードが開始されます。取得時間に関する詳細については、「アーカイブの取り出しオプション」を参照してください。

例: の高レベル API を使用したアーカイブのダウンロード AWS SDK for Java

次の Java コード例では、米国西部 (オレゴン リージョン (`us-west-2`) のボールト (`examplevault`) からアーカイブをダウンロードします。

このサンプルを実行するための詳しい手順については、「Eclipse を使用した Amazon S3 Glacier の Java 実行例」を参照してください。ここに示したコードは、既存のアーカイブ ID とダウンロードしたアーカイブを保存するローカルファイルパスで更新する必要があります。

Example

```
import java.io.File;
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.transfer.ArchiveTransferManager;
import com.amazonaws.services.sns.AmazonSNSClient;
import com.amazonaws.services.sqs.AmazonSQSClient;
```

```
public class ArchiveDownloadHighLevel {
    public static String vaultName = "examplevault";
    public static String archiveId = "*** provide archive ID ***";
    public static String downloadFilePath  = "*** provide location to download archive
 ***";

    public static AmazonGlacierClient glacierClient;
    public static AmazonSQSClient sqsClient;
    public static AmazonSNSClient snsClient;

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

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        glacierClient = new AmazonGlacierClient(credentials);

        sqsClient = new AmazonSQSClient(credentials);
        snsClient = new AmazonSNSClient(credentials);
        glacierClient.setEndpoint("glacier.us-west-2.amazonaws.com");
        sqsClient.setEndpoint("sqs.us-west-2.amazonaws.com");
        snsClient.setEndpoint("sns.us-west-2.amazonaws.com");

        try {
            ArchiveTransferManager atm = new ArchiveTransferManager(glacierClient,
 sqsClient, snsClient);

            atm.download(vaultName, archiveId, new File(downloadFilePath));
            System.out.println("Downloaded file to " + downloadFilePath);

        } catch (Exception e)
        {
            System.err.println(e);
        }
    }
}
```

の低レベル API を使用したアーカイブのダウンロード AWS SDK for Java

以下に、 AWS SDK for Java の低レベル API を使用してボールトインベントリを取得する手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

アーカイブをダウンロードする AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `archive-retrieval` メソッドを実行して、`initiateJob` ジョブを開始します。

   `InitiateJobRequest` クラスのインスタンスを作成することにより、ダウンロードするアーカイブのアーカイブ ID や、Amazon S3 Glacier (S3 Glacier) でジョブの完了メッセージを投稿する Amazon SNS トピック (オプション) などのジョブ情報を入力します。 は、レスポンスとしてジョブ ID を返します。S3 Glacier は、レスポンスとしてジョブ ID を返します。レスポンスは、`InitiateJobResult` クラスのインスタンスで使用できます。

```
JobParameters jobParameters = new JobParameters()
    .withArchiveId("*** provide an archive id ***")
    .withDescription("archive retrieval")
    .withRetrievalByteRange("*** provide a retrieval range***") // optional
    .withType("archive-retrieval");

InitiateJobResult initiateJobResult = client.initiateJob(new InitiateJobRequest()
    .withJobParameters(jobParameters)
    .withVaultName(vaultName));

String jobId = initiateJobResult.getJobId();
```

オプションで、バイト範囲を指定して、アーカイブの一部のみを準備するよう S3 Glacier にリクエストすることができます。たとえば、次のステートメントを追加すると、前のリクエストが更新され、アーカイブの 1 MB から 2 MB の部分のみを準備するよう S3 Glacier にリクエストできます。

```
int ONE_MEG = 1048576;
String retrievalByteRange = String.format("%s-%s", ONE_MEG, 2*ONE_MEG -1);

JobParameters jobParameters = new JobParameters()
    .withType("archive-retrieval")
    .withArchiveId(archiveId)
    .withRetrievalByteRange(retrievalByteRange)
    .withSNSTopic(snsTopicARN);

InitiateJobResult initiateJobResult = client.initiateJob(new InitiateJobRequest()
    .withJobParameters(jobParameters)
```

```
    .withVaultName(vaultName));

String jobId = initiateJobResult.getJobId();
```

3. ジョブが完了するまで待ちます。

   ジョブの出力をダウンロードする準備が整うまで待つ必要があります。ボールトの通知設定により Amazon Simple Notification Service (Amazon SNS) トピックを指定している場合、またはジョブを開始したときに Amazon SNS トピックを指定している場合は、ジョブの完了後に S3 Glacier によりそのトピックにメッセージが送信されます。

   また、`describeJob` メソッドを呼び出して S3 Glacier にポーリングすることで、ジョブの完了ステータスを調べることもできます。ただし、通知のために Amazon SNS トピックを使用することをお勧めします。

4. `getJobOutput` メソッドを実行して、ジョブの出力 (アーカイブデータ) をダウンロードします。

   `GetJobOutputRequest` クラスのインスタンスを作成することにより、ジョブ ID やボールト名などのリクエスト情報を指定します。S3 Glacier により返される出力は `GetJobOutputResult` オブジェクトで使用できます。

```
GetJobOutputRequest jobOutputRequest = new GetJobOutputRequest()
        .withJobId("*** provide a job ID ***")
        .withVaultName("*** provide a vault name ****");
GetJobOutputResult jobOutputResult = client.getJobOutput(jobOutputRequest);

// jobOutputResult.getBody() // Provides the input stream.
```

   前述のコードスニペットは、ジョブの出力全体をダウンロードします。このほか、`GetJobOutputRequest` でバイト範囲を指定することにより、出力の一部のみを取得したり、出力全体を小さなチャンクに分けてダウンロードしたりできます。

```
GetJobOutputRequest jobOutputRequest = new GetJobOutputRequest()
        .withJobId("*** provide a job ID ***")
        .withRange("bytes=0-1048575")   // Download only the first 1 MB of the
 output.
        .withVaultName("*** provide a vault name ****");
```

`GetJobOutput` 呼び出しに対するレスポンスとして、 S3 Glacier では、特定の条件が満たされた場合に、ダウンロードしたデータの一部のチェックサムを返します。詳細については、「データをダウンロードするときのチェックサムの受信」を参照してください。

ダウンロードにエラーがないことを確認するために、クライアント側でチェックサムを計算し、S3 Glacier からレスポンスとして送信されたチェックサムと比較できます。

オプションの範囲が指定されたアーカイブの取得ジョブの場合は、ジョブの説明を取得すると、取得する範囲のチェックサム (SHA256TreeHash) が含まれます。この値を使用して、後でダウンロードするバイト範囲全体の正確性を詳しく確認できます。たとえば、木構造ハッシュ可能なアーカイブ範囲を取得するジョブを開始してから、出力を複数のチャンクに分けてダウンロードし、`GetJobOutput` リクエストでそれぞれチェックサムが返されるようにした場合は、クライアント側でダウンロードした各部分のチェックサムを計算してから木構造ハッシュを計算することができます。この計算結果を、S3 Glacier でジョブの説明リクエストに対して返されたレスポンスのチェックサムと比較して、ダウンロードしたバイト範囲全体が S3 Glacier に格納されているバイト範囲と同じであることを確認できます。

実例については、「例 2: の低レベル API を使用してアーカイブを取得する AWS SDK for Java— 出力をチャンク単位でダウンロードする 」を参照してください。

例 1: の低レベル API を使用してアーカイブを取得する AWS SDK for Java

次の Java コード例では、指定したボールトからアーカイブをダウンロードします。この例では、ジョブが完了した後で、単一の `getJobOutput` 呼び出しで出力全体をダウンロードします。出力をチャンクに分けてダウンロードする例については、「例 2: の低レベル API を使用してアーカイブを取得する AWS SDK for Java— 出力をチャンク単位でダウンロードする 」を参照してください。

この例では次のタスクを実行しています。

- Amazon Simple Notification Service (Amazon SNS) のトピック を作成する

  S3 Glacier は、ジョブの完了後、このトピックに通知を送信します。
- Amazon Simple Queue Service (Amazon SQS) キューを作成する

  この例では、ポリシーをキューにアタッチして、Amazon SNS トピックでメッセージをキューに投稿できるようにします。
- 指定したアーカイブをダウンロードするジョブを開始します。

ジョブのリクエストでは、ジョブの完了後に S3 Glacier がトピックへの通知を発行できるように、作成した Amazon SNS トピックを指定しています。

- Amazon SQS キューにジョブ ID を含むメッセージがあるかどうかを定期的に確認します。

  メッセージがある場合は、JSON を解析し、ジョブが正常に完了したかどうかを確認します。正常に完了している場合は、アーカイブをダウンロードします。

- Amazon SNS トピックおよび作成された Amazon SQS キューを削除して、クリーンアップします。

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.codehaus.jackson.JsonFactory;
import org.codehaus.jackson.JsonNode;
import org.codehaus.jackson.JsonParseException;
import org.codehaus.jackson.JsonParser;
import org.codehaus.jackson.map.ObjectMapper;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
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.SQSActions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.GetJobOutputRequest;
import com.amazonaws.services.glacier.model.GetJobOutputResult;
```

```
import com.amazonaws.services.glacier.model.InitiateJobRequest;
import com.amazonaws.services.glacier.model.InitiateJobResult;
import com.amazonaws.services.glacier.model.JobParameters;
import com.amazonaws.services.sns.AmazonSNSClient;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
import com.amazonaws.services.sns.model.DeleteTopicRequest;
import com.amazonaws.services.sns.model.SubscribeRequest;
import com.amazonaws.services.sns.model.SubscribeResult;
import com.amazonaws.services.sns.model.UnsubscribeRequest;
import com.amazonaws.services.sqs.AmazonSQSClient;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.CreateQueueResult;
import com.amazonaws.services.sqs.model.DeleteQueueRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;


public class AmazonGlacierDownloadArchiveWithSQSPolling {

    public static String archiveId = "*** provide archive ID ****";
    public static String vaultName = "*** provide vault name ***";
    public static String snsTopicName = "*** provide topic name ***";
    public static String sqsQueueName = "*** provide queue name ***";
    public static String sqsQueueARN;
    public static String sqsQueueURL;
    public static String snsTopicARN;
    public static String snsSubscriptionARN;
    public static String fileName = "*** provide file name ***";
    public static String region = "*** region ***";
    public static long sleepTime = 600;
    public static AmazonGlacierClient client;
    public static AmazonSQSClient sqsClient;
    public static AmazonSNSClient snsClient;

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

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier." + region + ".amazonaws.com");
```

```
        sqsClient = new AmazonSQSClient(credentials);
        sqsClient.setEndpoint("https://sqs." + region + ".amazonaws.com");
        snsClient = new AmazonSNSClient(credentials);
        snsClient.setEndpoint("https://sns." + region + ".amazonaws.com");

        try {
            setupSQS();

            setupSNS();

            String jobId = initiateJobRequest();
            System.out.println("Jobid = " + jobId);

            Boolean success = waitForJobToComplete(jobId, sqsQueueURL);
            if (!success) { throw new Exception("Job did not complete
successfully."); }

            downloadJobOutput(jobId);

            cleanUp();

        } catch (Exception e) {
            System.err.println("Archive retrieval failed.");
            System.err.println(e);
        }
    }

    private static void setupSQS() {
        CreateQueueRequest request = new CreateQueueRequest()
            .withQueueName(sqsQueueName);
        CreateQueueResult result = sqsClient.createQueue(request);
        sqsQueueURL = result.getQueueUrl();

        GetQueueAttributesRequest qRequest = new GetQueueAttributesRequest()
            .withQueueUrl(sqsQueueURL)
            .withAttributeNames("QueueArn");

        GetQueueAttributesResult qResult = sqsClient.getQueueAttributes(qRequest);
        sqsQueueARN = qResult.getAttributes().get("QueueArn");

        Policy sqsPolicy =
            new Policy().withStatements(
                    new Statement(Effect.Allow)
                    .withPrincipals(Principal.AllUsers)
```

```
                    .withActions(SQSActions.SendMessage)
                    .withResources(new Resource(sqsQueueARN)));
        Map<String, String> queueAttributes = new HashMap<String, String>();
        queueAttributes.put("Policy", sqsPolicy.toJson());
        sqsClient.setQueueAttributes(new SetQueueAttributesRequest(sqsQueueURL,
 queueAttributes));

    }
    private static void setupSNS() {
        CreateTopicRequest request = new CreateTopicRequest()
            .withName(snsTopicName);
        CreateTopicResult result = snsClient.createTopic(request);
        snsTopicARN = result.getTopicArn();

        SubscribeRequest request2 = new SubscribeRequest()
            .withTopicArn(snsTopicARN)
            .withEndpoint(sqsQueueARN)
            .withProtocol("sqs");
        SubscribeResult result2 = snsClient.subscribe(request2);

        snsSubscriptionARN = result2.getSubscriptionArn();
    }
    private static String initiateJobRequest() {

        JobParameters jobParameters = new JobParameters()
            .withType("archive-retrieval")
            .withArchiveId(archiveId)
            .withSNSTopic(snsTopicARN);

        InitiateJobRequest request = new InitiateJobRequest()
            .withVaultName(vaultName)
            .withJobParameters(jobParameters);

        InitiateJobResult response = client.initiateJob(request);

        return response.getJobId();
    }

    private static Boolean waitForJobToComplete(String jobId, String sqsQueueUrl)
 throws InterruptedException, JsonParseException, IOException {

        Boolean messageFound = false;
        Boolean jobSuccessful = false;
        ObjectMapper mapper = new ObjectMapper();
```

```
        JsonFactory factory = mapper.getJsonFactory();

        while (!messageFound) {
            List<Message> msgs = sqsClient.receiveMessage(
               new
 ReceiveMessageRequest(sqsQueueUrl).withMaxNumberOfMessages(10)).getMessages();

            if (msgs.size() > 0) {
                for (Message m : msgs) {
                    JsonParser jpMessage = factory.createJsonParser(m.getBody());
                    JsonNode jobMessageNode = mapper.readTree(jpMessage);
                    String jobMessage = jobMessageNode.get("Message").getTextValue();

                    JsonParser jpDesc = factory.createJsonParser(jobMessage);
                    JsonNode jobDescNode = mapper.readTree(jpDesc);
                    String retrievedJobId = jobDescNode.get("JobId").getTextValue();
                    String statusCode = jobDescNode.get("StatusCode").getTextValue();
                    if (retrievedJobId.equals(jobId)) {
                        messageFound = true;
                        if (statusCode.equals("Succeeded")) {
                            jobSuccessful = true;
                        }
                    }
                }

            } else {
              Thread.sleep(sleepTime * 1000);
            }
          }
        return (messageFound && jobSuccessful);
    }

    private static void downloadJobOutput(String jobId) throws IOException {

        GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
            .withVaultName(vaultName)
            .withJobId(jobId);
        GetJobOutputResult getJobOutputResult =
 client.getJobOutput(getJobOutputRequest);

        InputStream input = new BufferedInputStream(getJobOutputResult.getBody());
        OutputStream output = null;
        try {
            output = new BufferedOutputStream(new FileOutputStream(fileName));
```

```
            byte[] buffer = new byte[1024 * 1024];

            int bytesRead = 0;
            do {
                bytesRead = input.read(buffer);
                if (bytesRead <= 0) break;
                output.write(buffer, 0, bytesRead);
            } while (bytesRead > 0);
        } catch (IOException e) {
            throw new AmazonClientException("Unable to save archive", e);
        } finally {
            try {input.close();}  catch (Exception e) {}
            try {output.close();} catch (Exception e) {}
        }
        System.out.println("Retrieved archive to " + fileName);
    }

    private static void cleanUp() {
        snsClient.unsubscribe(new UnsubscribeRequest(snsSubscriptionARN));
        snsClient.deleteTopic(new DeleteTopicRequest(snsTopicARN));
        sqsClient.deleteQueue(new DeleteQueueRequest(sqsQueueURL));
    }
}
```

例 2: の低レベル API を使用してアーカイブを取得する AWS SDK for Java— 出力をチャンク単位でダウンロードする

次の Java コード例では、S3 Glacier からアーカイブを取得します。このコード例では、`GetJobOutputRequest` オブジェクトのバイト範囲を指定することにより、ジョブの出力をチャンクに分けてダウンロードします。

```
import java.io.BufferedInputStream;
import java.io.ByteArrayInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParseException;
```

```
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;

import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Principal;
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.SQSActions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.TreeHashGenerator;
import com.amazonaws.services.glacier.model.GetJobOutputRequest;
import com.amazonaws.services.glacier.model.GetJobOutputResult;
import com.amazonaws.services.glacier.model.InitiateJobRequest;
import com.amazonaws.services.glacier.model.InitiateJobResult;
import com.amazonaws.services.glacier.model.JobParameters;
import com.amazonaws.services.sns.AmazonSNSClient;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
import com.amazonaws.services.sns.model.DeleteTopicRequest;
import com.amazonaws.services.sns.model.SubscribeRequest;
import com.amazonaws.services.sns.model.SubscribeResult;
import com.amazonaws.services.sns.model.UnsubscribeRequest;
import com.amazonaws.services.sqs.AmazonSQSClient;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.CreateQueueResult;
import com.amazonaws.services.sqs.model.DeleteQueueRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesRequest;
import com.amazonaws.services.sqs.model.GetQueueAttributesResult;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;


public class ArchiveDownloadLowLevelWithRange {

    public static String vaultName = "*** provide vault name ***";
    public static String archiveId = "*** provide archive id ***";
    public static String snsTopicName = "glacier-temp-sns-topic";
    public static String sqsQueueName = "glacier-temp-sqs-queue";
    public static long downloadChunkSize = 4194304; // 4 MB
    public static String sqsQueueARN;
```

```
    public static String sqsQueueURL;
    public static String snsTopicARN;
    public static String snsSubscriptionARN;
    public static String fileName = "*** provide file name to save archive to ***";
    public static String region   = "*** region ***";
    public static long sleepTime  = 600;

    public static AmazonGlacierClient client;
    public static AmazonSQSClient sqsClient;
    public static AmazonSNSClient snsClient;

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

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier." + region + ".amazonaws.com");
        sqsClient = new AmazonSQSClient(credentials);
        sqsClient.setEndpoint("https://sqs." + region + ".amazonaws.com");
        snsClient = new AmazonSNSClient(credentials);
        snsClient.setEndpoint("https://sns." + region + ".amazonaws.com");

        try {
            setupSQS();

            setupSNS();

            String jobId = initiateJobRequest();
            System.out.println("Jobid = " + jobId);

            long archiveSizeInBytes = waitForJobToComplete(jobId, sqsQueueURL);
            if (archiveSizeInBytes==-1) { throw new Exception("Job did not complete
successfully."); }

            downloadJobOutput(jobId, archiveSizeInBytes);

            cleanUp();

        } catch (Exception e) {
            System.err.println("Archive retrieval failed.");
            System.err.println(e);
        }
    }
```

```
    private static void setupSQS() {
        CreateQueueRequest request = new CreateQueueRequest()
            .withQueueName(sqsQueueName);
        CreateQueueResult result = sqsClient.createQueue(request);
        sqsQueueURL = result.getQueueUrl();

        GetQueueAttributesRequest qRequest = new GetQueueAttributesRequest()
            .withQueueUrl(sqsQueueURL)
            .withAttributeNames("QueueArn");

        GetQueueAttributesResult qResult = sqsClient.getQueueAttributes(qRequest);
        sqsQueueARN = qResult.getAttributes().get("QueueArn");

        Policy sqsPolicy =
            new Policy().withStatements(
                    new Statement(Effect.Allow)
                    .withPrincipals(Principal.AllUsers)
                    .withActions(SQSActions.SendMessage)
                    .withResources(new Resource(sqsQueueARN)));
        Map<String, String> queueAttributes = new HashMap<String, String>();
        queueAttributes.put("Policy", sqsPolicy.toJson());
        sqsClient.setQueueAttributes(new SetQueueAttributesRequest(sqsQueueURL,
 queueAttributes));

    }
    private static void setupSNS() {
        CreateTopicRequest request = new CreateTopicRequest()
            .withName(snsTopicName);
        CreateTopicResult result = snsClient.createTopic(request);
        snsTopicARN = result.getTopicArn();

        SubscribeRequest request2 = new SubscribeRequest()
            .withTopicArn(snsTopicARN)
            .withEndpoint(sqsQueueARN)
            .withProtocol("sqs");
        SubscribeResult result2 = snsClient.subscribe(request2);

        snsSubscriptionARN = result2.getSubscriptionArn();
    }
    private static String initiateJobRequest() {

        JobParameters jobParameters = new JobParameters()
            .withType("archive-retrieval")
            .withArchiveId(archiveId)
```

```
            .withSNSTopic(snsTopicARN);

        InitiateJobRequest request = new InitiateJobRequest()
            .withVaultName(vaultName)
            .withJobParameters(jobParameters);

        InitiateJobResult response = client.initiateJob(request);

        return response.getJobId();
    }

    private static long waitForJobToComplete(String jobId, String sqsQueueUrl) throws
 InterruptedException, JsonParseException, IOException {

        Boolean messageFound = false;
        Boolean jobSuccessful = false;
        long archiveSizeInBytes = -1;
        ObjectMapper mapper = new ObjectMapper();
        JsonFactory factory = mapper.getFactory();

        while (!messageFound) {
            List<Message> msgs = sqsClient.receiveMessage(
               new
 ReceiveMessageRequest(sqsQueueUrl).withMaxNumberOfMessages(10)).getMessages();

            if (msgs.size() > 0) {
                for (Message m : msgs) {
                    JsonParser jpMessage = factory.createJsonParser(m.getBody());
                    JsonNode jobMessageNode = mapper.readTree(jpMessage);
                    String jobMessage = jobMessageNode.get("Message").textValue();

                    JsonParser jpDesc = factory.createJsonParser(jobMessage);
                    JsonNode jobDescNode = mapper.readTree(jpDesc);
                    String retrievedJobId = jobDescNode.get("JobId").textValue();
                    String statusCode = jobDescNode.get("StatusCode").textValue();
                    archiveSizeInBytes =
 jobDescNode.get("ArchiveSizeInBytes").longValue();
                    if (retrievedJobId.equals(jobId)) {
                        messageFound = true;
                        if (statusCode.equals("Succeeded")) {
                            jobSuccessful = true;
                        }
                    }
                }
```

```
            } else {
              Thread.sleep(sleepTime * 1000);
            }
          }
        return (messageFound && jobSuccessful) ? archiveSizeInBytes : -1;
    }

    private static void downloadJobOutput(String jobId, long archiveSizeInBytes) throws
 IOException {

        if (archiveSizeInBytes < 0) {
            System.err.println("Nothing to download.");
            return;
        }

        System.out.println("archiveSizeInBytes: " + archiveSizeInBytes);
        FileOutputStream fstream = new FileOutputStream(fileName);
        long startRange = 0;
        long endRange = (downloadChunkSize > archiveSizeInBytes) ? archiveSizeInBytes
 -1 : downloadChunkSize - 1;

        do {

            GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
                .withVaultName(vaultName)
                .withRange("bytes=" + startRange + "-" + endRange)
                .withJobId(jobId);
            GetJobOutputResult getJobOutputResult =
 client.getJobOutput(getJobOutputRequest);

            BufferedInputStream is = new
 BufferedInputStream(getJobOutputResult.getBody());
            byte[] buffer = new byte[(int)(endRange - startRange + 1)];

            System.out.println("Checksum received: " +
 getJobOutputResult.getChecksum());
            System.out.println("Content range " +
 getJobOutputResult.getContentRange());


            int totalRead = 0;
            while (totalRead < buffer.length) {
                int bytesRemaining = buffer.length - totalRead;
```

```
                int read = is.read(buffer, totalRead, bytesRemaining);
                if (read > 0) {
                    totalRead = totalRead + read;
                } else {
                    break;
                }

            }
            System.out.println("Calculated checksum: " +
 TreeHashGenerator.calculateTreeHash(new ByteArrayInputStream(buffer)));
            System.out.println("read = " + totalRead);
            fstream.write(buffer);

            startRange = startRange + (long)totalRead;
            endRange = ((endRange + downloadChunkSize) >  archiveSizeInBytes) ?
 archiveSizeInBytes : (endRange + downloadChunkSize);
            is.close();
        } while (endRange <= archiveSizeInBytes  && startRange < archiveSizeInBytes);

        fstream.close();
        System.out.println("Retrieved file to " + fileName);

    }

    private static void cleanUp() {
        snsClient.unsubscribe(new UnsubscribeRequest(snsSubscriptionARN));
        snsClient.deleteTopic(new DeleteTopicRequest(snsTopicARN));
        sqsClient.deleteQueue(new DeleteQueueRequest(sqsQueueURL));
    }
}
```

Amazon S3 Glacier で AWS SDK for .NETを使用してアーカイブをダウンロードする

両方高レベル API と低レベル API.NET 用の Amazon SDK で提供されており、アーカイブをダウンロードする方法を提供します。

トピック

- の高レベル API を使用したアーカイブのダウンロード AWS SDK for .NET
- の低レベル API を使用したアーカイブのダウンロード AWS SDK for .NET

の高レベル API を使用したアーカイブのダウンロード AWS SDK for .NET

高レベル API の `ArchiveTransferManager` クラスには、アーカイブのダウンロードに使用できる `Download` メソッドが用意されています。

> **Important**
>
> -`ArchiveTransferManager`クラスは、Amazon Simple Notification Service (Amazon SNS) トピックと、そのトピックにサブスクライブされている Amazon Simple Queue Service (Amazon SQS) キューを作成します。その後、アーカイブの取り出しジョブを開始し、使用可能にするアーカイブを探してキューをポーリングします。アーカイブが使用可能になると、ダウンロードが開始されます。取得時間に関する詳細については、「アーカイブの取り出しオプション」を参照してください。

例: の高レベル API を使用したアーカイブのダウンロード AWS SDK for .NET

次の C# コード例では、米国西部 (オレゴン リージョン (`examplevault`) のボールト からアーカイブをダウンロードします。

この例を実行するための詳しい手順については、「コード例の実行」を参照してください。ここに示したコードは、既存のアーカイブ ID とダウンロードしたアーカイブを保存するローカルファイルパスで更新する必要があります。

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveDownloadHighLevel
  {
    static string vaultName        = "examplevault";
    static string archiveId        = "*** Provide archive ID ***";
    static string downloadFilePath = "*** Provide the file name and path to where to
 store the download ***";

    public static void Main(string[] args)
    {
      try
```

```
      {
        var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);

        var options = new DownloadOptions();
        options.StreamTransferProgress += ArchiveDownloadHighLevel.progress;
        // Download an archive.
        Console.WriteLine("Intiating the archive retrieval job and then polling SQS
 queue for the archive to be available.");
        Console.WriteLine("Once the archive is available, downloading will begin.");
        manager.Download(vaultName, archiveId, downloadFilePath, options);
        Console.WriteLine("To continue, press Enter");
        Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static int currentPercentage = -1;
    static void progress(object sender, StreamTransferProgressArgs args)
    {
      if (args.PercentDone != currentPercentage)
      {
        currentPercentage = args.PercentDone;
        Console.WriteLine("Downloaded {0}%", args.PercentDone);
      }
    }
  }
}
```

の低レベル API を使用したアーカイブのダウンロード AWS SDK for .NET

以下に、 AWS SDK for .NETの低レベル API を使用して、Amazon S3 Glacier (S3 Glacier) アーカイブをダウンロードするためのステップを示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   アーカイブをダウンロードする AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `archive-retrieval` メソッドを実行して、`InitiateJob` ジョブを開始します。

`InitiateJobRequest` クラスのインスタンスを作成することにより、ダウンロードするアーカイブのアーカイブ ID や、 S3 Glacier がジョブの完了メッセージを投稿する Amazon SNS トピック (オプション) などのジョブ情報を入力します。 は、レスポンスとしてジョブ ID を返します。S3 Glacier は、レスポンスとしてジョブ ID を返します。レスポンスは、`InitiateJobResponse` クラスのインスタンスで使用できます。

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);

InitiateJobRequest initJobRequest = new InitiateJobRequest()
{
  VaultName = vaultName,
  JobParameters = new JobParameters()
  {
    Type      = "archive-retrieval",
    ArchiveId = "*** Provide archive id ***",
    SNSTopic  = "*** Provide Amazon SNS topic ARN ***",
  }
};

InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
string jobId = initJobResponse.JobId;
```

必要に応じて、以下のリクエストに示すように、バイト範囲を指定して、アーカイブの一部のみを準備するよう S3 Glacier にリクエストすることもできます。リクエストでは S3 Glacier に対し、アーカイブの 1 MB から 2 MB までの部分のみを準備するように指定しています。

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);



InitiateJobRequest initJobRequest = new InitiateJobRequest()
{
  VaultName = vaultName,
  JobParameters = new JobParameters()
  {
    Type      = "archive-retrieval",
    ArchiveId = "*** Provide archive id ***",
    SNSTopic  = "*** Provide Amazon SNS topic ARN ***",
  }
};
```

```
// Specify byte range.
int ONE_MEG = 1048576;
initJobRequest.JobParameters.RetrievalByteRange = string.Format("{0}-{1}", ONE_MEG, 2
 * ONE_MEG -1);

InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
string jobId = initJobResponse.JobId;
```

3. ジョブが完了するまで待ちます。

   ジョブの出力をダウンロードする準備が整うまで待つ必要があります。ボールトの通知設定により Amazon Simple Notification Service (Amazon SNS) トピックを指定している場合、またはジョブを開始したときに Amazon SNS トピックを指定している場合は、ジョブの完了後に S3 Glacier によりそのトピックにメッセージが送信されます。以下のセクションに示しているコード例では、Amazon SNS を使用して、 S3 Glacier でメッセージを発行します。

   また、`DescribeJob` メソッドを呼び出して S3 Glacier にポーリングすることで、ジョブの完了ステータスを調べることもできます。ただし、通知のために Amazon SNS トピックを使用することをお勧めします。

4. `GetJobOutput` メソッドを実行して、ジョブの出力 (アーカイブデータ) をダウンロードします。

   `GetJobOutputRequest` クラスのインスタンスを作成することにより、ジョブ ID やボールト名などのリクエスト情報を指定します。S3 Glacier により返される出力は `GetJobOutputResponse` オブジェクトで使用できます。

```
GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
{
  JobId = jobId,
  VaultName = vaultName
};

GetJobOutputResponse getJobOutputResponse = client.GetJobOutput(getJobOutputRequest);
using (Stream webStream = getJobOutputResponse.Body)
{
  using (Stream fileToSave = File.OpenWrite(fileName))
  {
     CopyStream(webStream, fileToSave);
  }
}
```

前述のコードスニペットは、ジョブの出力全体をダウンロードします。このほか、GetJobOutputRequest でバイト範囲を指定することにより、出力の一部のみを取得したり、出力全体を小さなチャンクに分けてダウンロードしたりできます。

```
GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
{
  JobId = jobId,
  VaultName = vaultName
};
getJobOutputRequest.SetRange(0, 1048575); // Download only the first 1 MB chunk of
 the output.
```

GetJobOutput 呼び出しに対するレスポンスとして、 S3 Glacier では、特定の条件が満たされた場合に、ダウンロードしたデータの一部のチェックサムを返します。詳細については、「データをダウンロードするときのチェックサムの受信」を参照してください。

ダウンロードにエラーがないことを確認するために、クライアント側でチェックサムを計算し、S3 Glacier からレスポンスとして送信されたチェックサムと比較できます。

オプションの範囲が指定されたアーカイブの取得ジョブの場合は、ジョブの説明を取得したときに取得する範囲のチェックサムが含まれます (SHA256TreeHash) 。この値によって、後でダウンロードするバイト範囲全体の正確性を詳しく確認できます。たとえば、木構造ハッシュ可能なアーカイブ範囲を取得するジョブを開始してから、出力を複数のチャンクに分けてダウンロードし、GetJobOutput リクエストでそれぞれチェックサムが返されるようにした場合は、クライアント側でダウンロードした各部分のチェックサムを計算してから木構造ハッシュを計算することができます。この計算結果を、S3 Glacier でジョブの説明リクエストに対して返されたレスポンスのチェックサムと比較して、ダウンロードしたバイト範囲全体が S3 Glacier に格納されているバイト範囲と同じであることを確認できます。

実例については、「例 2: の低レベル API を使用してアーカイブを取得する AWS SDK for .NET—出力をチャンク単位でダウンロードする」を参照してください。

例 1: の低レベル API を使用してアーカイブを取得する AWS SDK for .NET

次の C# コード例は、指定したボールトからアーカイブをダウンロードします。この例では、ジョブが完了した後で、単一の GetJobOutput 呼び出しで出力全体をダウンロードします。出力をチャン

クに分けてダウンロードする例については、「例 2: の低レベル API を使用してアーカイブを取得する AWS SDK for .NET— 出力をチャンク単位でダウンロードする」を参照してください。

この例では次のタスクを実行しています。

- Amazon Simple Notification Service (Amazon SNS) のトピックを設定する

  S3 Glacier は、ジョブの完了後、このトピックに通知を送信します。
- Amazon Simple Queue Service (Amazon SQS) キューを設定する

  この例では、ポリシーをキューにアタッチして、Amazon SNS トピックでメッセージを投稿できるようにします。
- 指定したアーカイブをダウンロードするジョブを開始します。

  この例では、ジョブのリクエストとして、S3 Glacier によりジョブの完了後にメッセージが送信されるように Amazon SNS トピックを指定します。
- Amazon SQS キューにメッセージがあるかどうかを定期的に確認します。

  メッセージがある場合は、JSON を解析し、ジョブが正常に完了したかどうかを確認します。正常に完了している場合は、アーカイブをダウンロードします。コード例では、JSON.NET ライブラリ (JSON.NET 参照) を使用して JSON を解析しています。
- Amazon SNS トピックおよび作成された Amazon SQS キューを削除して、クリーンアップします。

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Runtime;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;
using Amazon.SQS;
using Amazon.SQS.Model;
using Newtonsoft.Json;

namespace glacier.amazon.com.docsamples
{
  class ArchiveDownloadLowLevelUsingSNSSQS
```

```
  {
    static string topicArn;
    static string queueUrl;
    static string queueArn;
    static string vaultName = "*** Provide vault name ***";
    static string archiveID = "*** Provide archive ID ***";
    static string fileName  = "*** Provide the file name and path to where to store
 downloaded archive ***";
    static AmazonSimpleNotificationServiceClient snsClient;
    static AmazonSQSClient sqsClient;
    const string SQS_POLICY =
        "{" +
        "    \"Version\" : \"2012-10-17\"," +
        "    \"Statement\" : [" +
        "        {" +
        "            \"Sid\" : \"sns-rule\"," +
        "            \"Effect\" : \"Allow\"," +
        "            \"Principal\" : {\"Service\" : \"sns.amazonaws.com\" }," +
        "            \"Action\"    : \"sqs:SendMessage\"," +
        "            \"Resource\"  : \"{QueueArn}\"," +
        "            \"Condition\" : {" +
        "                \"ArnLike\" : {" +
        "                    \"aws:SourceArn\" : \"{TopicArn}\"" +
        "                }" +
        "            }" +
        "        }" +
        "    ]" +
        "}";

    public static void Main(string[] args)
    {
      AmazonGlacierClient client;
      try
      {
        using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
          Console.WriteLine("Setup SNS topic and SQS queue.");
          SetupTopicAndQueue();
          Console.WriteLine("To continue, press Enter"); Console.ReadKey();
          Console.WriteLine("Retrieving...");
          RetrieveArchive(client);
        }
        Console.WriteLine("Operations successful. To continue, press Enter");
        Console.ReadKey();
```

```
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      finally
      {
        // Delete SNS topic and SQS queue.
        snsClient.DeleteTopic(new DeleteTopicRequest() { TopicArn = topicArn });
        sqsClient.DeleteQueue(new DeleteQueueRequest() { QueueUrl = queueUrl });
      }
    }

    static void SetupTopicAndQueue()
    {
      snsClient = new
 AmazonSimpleNotificationServiceClient(Amazon.RegionEndpoint.USWest2);
      sqsClient = new AmazonSQSClient(Amazon.RegionEndpoint.USWest2);

      long ticks = DateTime.Now.Ticks;
      topicArn = snsClient.CreateTopic(new CreateTopicRequest { Name =
 "GlacierDownload-" + ticks }).TopicArn;
      Console.Write("topicArn: "); Console.WriteLine(topicArn);

      CreateQueueRequest createQueueRequest = new CreateQueueRequest();
      createQueueRequest.QueueName = "GlacierDownload-" + ticks;
      CreateQueueResponse createQueueResponse =
 sqsClient.CreateQueue(createQueueRequest);
      queueUrl = createQueueResponse.QueueUrl;
      Console.Write("QueueURL: "); Console.WriteLine(queueUrl);

      GetQueueAttributesRequest getQueueAttributesRequest = new
 GetQueueAttributesRequest();
      getQueueAttributesRequest.AttributeNames = new List<string> { "QueueArn" };
      getQueueAttributesRequest.QueueUrl = queueUrl;
      GetQueueAttributesResponse response =
 sqsClient.GetQueueAttributes(getQueueAttributesRequest);
      queueArn = response.QueueARN;
      Console.Write("QueueArn: "); Console.WriteLine(queueArn);

      // Setup the Amazon SNS topic to publish to the SQS queue.
      snsClient.Subscribe(new SubscribeRequest()
      {
        Protocol = "sqs",
        Endpoint = queueArn,
```

```
        TopicArn = topicArn
      });

      // Add policy to the queue so SNS can send messages to the queue.
      var policy = SQS_POLICY.Replace("{TopicArn}", topicArn).Replace("{QueueArn}",
queueArn);

      sqsClient.SetQueueAttributes(new SetQueueAttributesRequest()
      {
          QueueUrl = queueUrl,
          Attributes = new Dictionary<string, string>
          {
              { QueueAttributeName.Policy, policy }
          }
      });
    }

    static void RetrieveArchive(AmazonGlacierClient client)
    {
      // Initiate job.
      InitiateJobRequest initJobRequest = new InitiateJobRequest()
      {
        VaultName = vaultName,
        JobParameters = new JobParameters()
        {
          Type = "archive-retrieval",
          ArchiveId = archiveID,
          Description = "This job is to download archive.",
          SNSTopic = topicArn,
        }
      };
      InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
      string jobId = initJobResponse.JobId;

      // Check queue for a message and if job completed successfully, download archive.
      ProcessQueue(jobId, client);
    }

    private static void ProcessQueue(string jobId, AmazonGlacierClient client)
    {
      ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest()
{ QueueUrl = queueUrl, MaxNumberOfMessages = 1 };
      bool jobDone = false;
      while (!jobDone)
```

```
      {
        Console.WriteLine("Poll SQS queue");
        ReceiveMessageResponse receiveMessageResponse =
sqsClient.ReceiveMessage(receiveMessageRequest);
        if (receiveMessageResponse.Messages.Count == 0)
        {
          Thread.Sleep(10000 * 60);
          continue;
        }
        Console.WriteLine("Got message");
        Message message = receiveMessageResponse.Messages[0];
        Dictionary<string, string> outerLayer =
JsonConvert.DeserializeObject<Dictionary<string, string>>(message.Body);
        Dictionary<string, object> fields =
JsonConvert.DeserializeObject<Dictionary<string, object>>(outerLayer["Message"]);
        string statusCode = fields["StatusCode"] as string;

        if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_SUCCEEDED,
StringComparison.InvariantCultureIgnoreCase))
        {
          Console.WriteLine("Downloading job output");
          DownloadOutput(jobId, client); // Save job output to the specified file
location.
        }
        else if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_FAILED,
StringComparison.InvariantCultureIgnoreCase))
          Console.WriteLine("Job failed... cannot download the archive.");

        jobDone = true;
        sqsClient.DeleteMessage(new DeleteMessageRequest() { QueueUrl = queueUrl,
ReceiptHandle = message.ReceiptHandle });
      }
    }

    private static void DownloadOutput(string jobId, AmazonGlacierClient client)
    {
      GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
      {
        JobId = jobId,
        VaultName = vaultName
      };

      GetJobOutputResponse getJobOutputResponse =
client.GetJobOutput(getJobOutputRequest);
```

```
      using (Stream webStream = getJobOutputResponse.Body)
      {
          using (Stream fileToSave = File.OpenWrite(fileName))
          {
              CopyStream(webStream, fileToSave);
          }
      }
    }

    public static void CopyStream(Stream input, Stream output)
    {
      byte[] buffer = new byte[65536];
      int length;
      while ((length = input.Read(buffer, 0, buffer.Length)) > 0)
      {
        output.Write(buffer, 0, length);
      }
    }
  }
}
```

例 2: の低レベル API を使用してアーカイブを取得する AWS SDK for .NET— 出力をチャンク単位でダウンロードする

次の C# コード例では、S3 Glacier からアーカイブを取得しています。このコード例では、`GetJobOutputRequest` オブジェクトのバイト範囲を指定することにより、ジョブの出力をチャンクに分けてダウンロードします。

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;
using Amazon.SQS;
using Amazon.SQS.Model;
using Newtonsoft.Json;
using System.Collections.Specialized;
```

```
namespace glacier.amazon.com.docsamples
{
  class ArchiveDownloadLowLevelUsingSQLSNSOutputUsingRange
  {
    static string topicArn;
    static string queueUrl;
    static string queueArn;
    static string vaultName = "*** Provide vault name ***";
    static string archiveId = "*** Provide archive ID ***";
    static string fileName  = "*** Provide the file name and path to where to store
 downloaded archive ***";
    static AmazonSimpleNotificationServiceClient snsClient;
    static AmazonSQSClient sqsClient;
    const string SQS_POLICY =
        "{" +
        "    \"Version\" : \"2012-10-17\"," +
        "    \"Statement\" : [" +
        "        {" +
        "            \"Sid\" : \"sns-rule\"," +
        "            \"Effect\" : \"Allow\"," +
        "            \"Principal\" : {\"AWS\" : \"arn:aws:iam::123456789012:root\" },"
 +
        "            \"Action\"  : \"sqs:SendMessage\"," +
        "            \"Resource\"  : \"{QuernArn}\"," +
        "            \"Condition\" : {" +
        "                \"ArnLike\" : {" +
        "                    \"aws:SourceArn\" : \"{TopicArn}\"" +
        "                }" +
        "            }" +
        "        }" +
        "    ]" +
        "}";

    public static void Main(string[] args)
    {
      AmazonGlacierClient client;

      try
      {
          using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
          {
              Console.WriteLine("Setup SNS topic and SQS queue.");
              SetupTopicAndQueue();
              Console.WriteLine("To continue, press Enter"); Console.ReadKey();
```

```
            Console.WriteLine("Download archive");
            DownloadAnArchive(archiveId, client);
        }
        Console.WriteLine("Operations successful. To continue, press Enter");
        Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      finally
      {
        // Delete SNS topic and SQS queue.
        snsClient.DeleteTopic(new DeleteTopicRequest() { TopicArn = topicArn });
        sqsClient.DeleteQueue(new DeleteQueueRequest() { QueueUrl = queueUrl });
      }
    }

       static void SetupTopicAndQueue()
    {
      long ticks = DateTime.Now.Ticks;

      // Setup SNS topic.
      snsClient = new
 AmazonSimpleNotificationServiceClient(Amazon.RegionEndpoint.USWest2);
      sqsClient = new AmazonSQSClient(Amazon.RegionEndpoint.USWest2);

      topicArn = snsClient.CreateTopic(new CreateTopicRequest { Name =
 "GlacierDownload-" + ticks }).TopicArn;
      Console.Write("topicArn: "); Console.WriteLine(topicArn);

      CreateQueueRequest createQueueRequest = new CreateQueueRequest();
      createQueueRequest.QueueName = "GlacierDownload-" + ticks;
      CreateQueueResponse createQueueResponse =
 sqsClient.CreateQueue(createQueueRequest);
      queueUrl = createQueueResponse.QueueUrl;
      Console.Write("QueueURL: "); Console.WriteLine(queueUrl);

      GetQueueAttributesRequest getQueueAttributesRequest = new
 GetQueueAttributesRequest();
      getQueueAttributesRequest.AttributeNames = new List<string> { "QueueArn" };
      getQueueAttributesRequest.QueueUrl = queueUrl;
      GetQueueAttributesResponse response =
 sqsClient.GetQueueAttributes(getQueueAttributesRequest);
```

```
      queueArn = response.QueueARN;
      Console.Write("QueueArn: "); Console.WriteLine(queueArn);

      // Setup the Amazon SNS topic to publish to the SQS queue.
      snsClient.Subscribe(new SubscribeRequest()
      {
        Protocol = "sqs",
        Endpoint = queueArn,
        TopicArn = topicArn
      });

      // Add the policy to the queue so SNS can send messages to the queue.
      var policy = SQS_POLICY.Replace("{TopicArn}", topicArn).Replace("{QuernArn}",
  queueArn);

      sqsClient.SetQueueAttributes(new SetQueueAttributesRequest()
      {
          QueueUrl = queueUrl,
          Attributes = new Dictionary<string, string>
          {
              { QueueAttributeName.Policy, policy }
          }
      });
    }

    static void DownloadAnArchive(string archiveId, AmazonGlacierClient client)
    {
      // Initiate job.
      InitiateJobRequest initJobRequest = new InitiateJobRequest()
      {

        VaultName = vaultName,
        JobParameters = new JobParameters()
        {
          Type = "archive-retrieval",
          ArchiveId = archiveId,
          Description = "This job is to download the archive.",
          SNSTopic = topicArn,
        }
      };
      InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
      string jobId = initJobResponse.JobId;

      // Check queue for a message and if job completed successfully, download archive.
```

```
      ProcessQueue(jobId, client);
    }

    private static void ProcessQueue(string jobId, AmazonGlacierClient client)
    {
        var receiveMessageRequest = new ReceiveMessageRequest() { QueueUrl = queueUrl,
MaxNumberOfMessages = 1 };
        bool jobDone = false;
        while (!jobDone)
        {
            Console.WriteLine("Poll SQS queue");
            ReceiveMessageResponse receiveMessageResponse =
sqsClient.ReceiveMessage(receiveMessageRequest);
            if (receiveMessageResponse.Messages.Count == 0)
            {
                Thread.Sleep(10000 * 60);
                continue;
            }
            Console.WriteLine("Got message");
            Message message = receiveMessageResponse.Messages[0];
            Dictionary<string, string> outerLayer =
JsonConvert.DeserializeObject<Dictionary<string, string>>(message.Body);
            Dictionary<string, object> fields =
JsonConvert.DeserializeObject<Dictionary<string, object>>(outerLayer["Message"]);
            string statusCode = fields["StatusCode"] as string;
            if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_SUCCEEDED,
StringComparison.InvariantCultureIgnoreCase))
            {
                long archiveSize = Convert.ToInt64(fields["ArchiveSizeInBytes"]);
                Console.WriteLine("Downloading job output");
                DownloadOutput(jobId, archiveSize, client); // This where we save job
output to the specified file location.
            }
            else if (string.Equals(statusCode, GlacierUtils.JOB_STATUS_FAILED,
StringComparison.InvariantCultureIgnoreCase))
                Console.WriteLine("Job failed... cannot download the archive.");
            jobDone = true;
            sqsClient.DeleteMessage(new DeleteMessageRequest() { QueueUrl = queueUrl,
ReceiptHandle = message.ReceiptHandle });
        }
    }

    private static void DownloadOutput(string jobId, long archiveSize,
AmazonGlacierClient client)
```

```
    {
      long partSize = 4 * (long)Math.Pow(2, 20);  // 4 MB.
      using (Stream fileToSave = new FileStream(fileName, FileMode.Create,
 FileAccess.Write))
      {

        long currentPosition = 0;
        do
        {
          GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
          {
            JobId = jobId,
            VaultName = vaultName
          };

          long endPosition = currentPosition + partSize - 1;
          if (endPosition > archiveSize)
            endPosition = archiveSize;

          getJobOutputRequest.SetRange(currentPosition, endPosition);
          GetJobOutputResponse getJobOutputResponse =
 client.GetJobOutput(getJobOutputRequest);

          using (Stream webStream = getJobOutputResponse.Body)
          {
            CopyStream(webStream, fileToSave);
          }
          currentPosition += partSize;
        } while (currentPosition < archiveSize);
      }
    }

    public static void CopyStream(Stream input, Stream output)
    {
      byte[] buffer = new byte[65536];
      int length;
      while ((length = input.Read(buffer, 0, buffer.Length)) > 0)
      {
        output.Write(buffer, 0, length);
      }
    }
  }
}
```

Python での並列処理を使用した大規模なアーカイブのダウンロード

このトピックでは、Python での並列処理を使用して Amazon S3 Glacier (S3 Glacier) から大きなアーカイブをダウンロードする方法について説明します。このアプローチにより、個別に処理できる小さな部分に分割することで、任意のサイズのアーカイブを確実にダウンロードできます。

概要

この例で提供されている Python スクリプトは、次のタスクを実行します。

1. 通知に必要な AWS リソース (Amazon SNS トピックと Amazon SQS キュー) を設定します。
2. S3 Glacier でアーカイブ取得ジョブを開始します
3. ジョブ完了通知の Amazon SQS キューをモニタリングします
4. 大きなアーカイブを管理可能なチャンクに分割します
5. 複数のワーカースレッドを使用してチャンクを並行してダウンロードします
6. 後で再組み立てできるように、各チャンクをディスクに保存します

前提条件

開始する前に、以下を確認してください。

- Python 3.6 以降がインストールされている
- AWS SDK for Python (Boto3) がインストールされている
- AWS S3 Glacier、Amazon SNS、Amazon SQS の適切なアクセス許可で設定された 認証情報
- ダウンロードしたアーカイブチャンクを保存するのに十分なディスク容量

例: Python での並列処理を使用したアーカイブのダウンロード

次の Python スクリプトは、並列処理を使用して S3 Glacier から大きなアーカイブをダウンロードする方法を示しています。

```
import boto3
import time
import json
import jmespath
import re
import concurrent.futures
```

```
import os

output_file_path = "output_directory_path"
vault_name = "vault_name"

chunk_size = 1000000000 #1gb - size of chunks for parallel download.
notify_queue_name = 'GlacierJobCompleteNotifyQueue' # SQS queue for Glacier recall
 notification
chunk_download_queue_name='GlacierChunkReadyNotifyQueue' # SQS queue for chunks
sns_topic_name = 'GlacierRecallJobCompleted' # the SNS topic to be notified when
 Glacier archive is restored.
chunk_queue_visibility_timeout = 7200 # 2 hours - this may need to be adjusted.
region = 'us-east-1'
archive_id = "archive_id_to_restore"
retrieve_archive = True # set to false if you do not want to restore from Glacier -
 useful for restarting or parallel processing of the chunk queue.
workers = 12 # the number of parallel worker threads for downloading chunks.

def setup_queues_and_topic():
    sqs = boto3.client('sqs')
    sns = boto3.client('sns')

    # Create the SNS topic
    topic_response = sns.create_topic(
        Name=sns_topic_name
    )
    topic_arn = topic_response['TopicArn']
    print("Creating the SNS topic " + topic_arn)

    # Create the notification queue
    notify_queue_response = sqs.create_queue(
        QueueName=notify_queue_name,
        Attributes={
            'VisibilityTimeout': '300',  # 5 minutes
            'ReceiveMessageWaitTimeSeconds': '20'  # Enable long polling
        }
    )
    notify_queue_url = notify_queue_response['QueueUrl']
    print("Creating the archive-retrieval notification queue " + notify_queue_url)

    # Create the chunk download queue
    chunk_queue_response = sqs.create_queue(
        QueueName=chunk_download_queue_name,
        Attributes={
```

```
            'VisibilityTimeout': str(chunk_queue_visibility_timeout),  # 5 minutes
            'ReceiveMessageWaitTimeSeconds': '0'
        }
    )
    chunk_queue_url = chunk_queue_response['QueueUrl']

    print("Creating the chunk ready notification queue " + chunk_queue_url)


   # Get the ARN for the notification queue
    notify_queue_attributes = sqs.get_queue_attributes(
        QueueUrl=notify_queue_url,
        AttributeNames=['QueueArn']
    )
    notify_queue_arn = notify_queue_attributes['Attributes']['QueueArn']

    # Set up the SNS topic policy on the notification queue
    queue_policy = {
        "Version": "2012-10-17",
        "Statement": [{
            "Sid": "allow-sns-messages",
            "Effect": "Allow",
            "Principal": {"AWS": "*"},
            "Action": "SQS:SendMessage",
            "Resource": notify_queue_arn,
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": topic_arn
                }
            }
        }]
    }

    # Set the queue policy
    sqs.set_queue_attributes(
        QueueUrl=notify_queue_url,
        Attributes={
            'Policy': json.dumps(queue_policy)
        }
    )

    # Subscribe the notification queue to the SNS topic
    sns.subscribe(
        TopicArn=topic_arn,
```

```
        Protocol='sqs',
        Endpoint=notify_queue_arn
    )

    return {
        'topic_arn': topic_arn,
        'notify_queue_url': notify_queue_url,
        'chunk_queue_url': chunk_queue_url
    }


def split_and_send_chunks(archive_size, job_id,chunk_queue_url):
    ranges = []
    current = 0
    chunk_number = 0

    while current < archive_size:
        chunk_number += 1
        next_range = min(current + chunk_size - 1, archive_size - 1)
        ranges.append((current, next_range, chunk_number))
        current = next_range + 1

    # Send messages to SQS queue
    for start, end, chunk_number in ranges:
        body = {"start": start, "end": end, "job_id": job_id, "chunk_number":
 chunk_number}
        body = json.dumps(body)
        print("Sending SQS message for range:" + str(body))
        response = sqs.send_message(
            QueueUrl=chunk_queue_url,
            MessageBody=str(body)
        )

def GetJobOutputChunks(job_id, byterange, chunk_number):
    glacier = boto3.client('glacier')
    response = glacier.get_job_output(
        vaultName=vault_name,
        jobId=job_id,
        range=byterange,

    )

    with open(os.path.join(output_file_path,str(chunk_number)+".chunk"), 'wb') as
 output_file:
```

```
        output_file.write(response['body'].read())

    return response

def ReceiveArchiveReadyMessages(notify_queue_url,chunk_queue_url):

    response = sqs.receive_message(
        QueueUrl=notify_queue_url,
        AttributeNames=['All'],
        MaxNumberOfMessages=1,
        WaitTimeSeconds=20,
        MessageAttributeNames=['Message']
    )
    print("Polling archive retrieval job ready queue...")
    # Checking that there is a Messages key before proceeding. No 'Messages' key likely
 means the queue is empty

    if 'Messages' in response:
        print("Received a message from the archive retrieval job queue")
        jsonresponse = response
        # Loading the string into JSON and checking that ArchiveSizeInBytes key is
 present before continuing.
        jsonresponse=json.loads(jsonresponse['Messages'][0]['Body'])
        jsonresponse=json.loads(jsonresponse['Message'])
        if 'ArchiveSizeInBytes' in jsonresponse:
            receipt_handle = response['Messages'][0]['ReceiptHandle']
            if jsonresponse['ArchiveSizeInBytes']:
                archive_size = jsonresponse['ArchiveSizeInBytes']

                print(f'Received message: {response}')
                if archive_size > chunk_size:
                    split_and_send_chunks(archive_size,
 jsonresponse['JobId'],chunk_queue_url)

                    sqs.delete_message(
                    QueueUrl=notify_queue_url,
                    ReceiptHandle=receipt_handle)

            else:
                print("No ArchiveSizeInBytes value found in message")
                print(response)

    else:
        print('No messages available in the queue at this time.')
```

```
    time.sleep(1)

def ReceiveArchiveChunkMessages(chunk_queue_url):
    response = sqs.receive_message(
        QueueUrl=chunk_queue_url,
        AttributeNames=['All'],
        MaxNumberOfMessages=1,
        WaitTimeSeconds=0,
        MessageAttributeNames=['Message']
    )
    print("Polling archive chunk queue...")
    print(response)
    # Checking that there is a Messages key before proceeding. No 'Messages' key likely
 means the queue is empty
    if 'Messages' in response:
        jsonresponse = response
        # Loading the string into JSON and checking that ArchiveSizeInBytes key is
 present before continuing.
        jsonresponse=json.loads(jsonresponse['Messages'][0]['Body'])
        if 'job_id' in jsonresponse: #checking that there is a job id before continuing
            job_id = jsonresponse['job_id']
            byterange = "bytes="+str(jsonresponse['start']) + '-' +
 str(jsonresponse['end'])
            chunk_number = jsonresponse['chunk_number']
            receipt_handle = response['Messages'][0]['ReceiptHandle']
            if jsonresponse['job_id']:
                print(f'Received message: {response}')
                GetJobOutputChunks(job_id,byterange,chunk_number)
                sqs.delete_message(
                QueueUrl=chunk_queue_url,
                ReceiptHandle=receipt_handle)
    else:
        print('No messages available in the chunk queue at this time.')

def initiate_archive_retrieval(archive_id, topic_arn):
    glacier = boto3.client('glacier')

    job_parameters = {
        "Type": "archive-retrieval",
        "ArchiveId": archive_id,
        "Description": "Archive retrieval job",
        "SNSTopic": topic_arn,
```

```
        "Tier": "Bulk"  # You can change this to "Standard" or "Expedited" based on
 your needs
    }

    try:
        response = glacier.initiate_job(
            vaultName=vault_name,
            jobParameters=job_parameters
        )

        print("Archive retrieval job initiated:")
        print(f"Job ID: {response['jobId']}")
        print(f"Job parameters: {job_parameters}")
        print(f"Complete response: {json.dumps(response, indent=2)}")

        return response['jobId']

    except Exception as e:
        print(f"Error initiating archive retrieval job: {str(e)}")
        raise

def run_async_tasks(chunk_queue_url, workers):
    max_workers = workers  # Set the desired maximum number of concurrent tasks
    with concurrent.futures.ThreadPoolExecutor(max_workers=max_workers) as executor:
        for _ in range(max_workers):
            executor.submit(ReceiveArchiveChunkMessages, chunk_queue_url)

# One time setup of the necessary queues and topics.
queue_and_topic_atts = setup_queues_and_topic()

topic_arn = queue_and_topic_atts['topic_arn']
notify_queue_url = queue_and_topic_atts['notify_queue_url']
chunk_queue_url = queue_and_topic_atts['chunk_queue_url']

if retrieve_archive:
    print("Retrieving the defined archive... The topic arn we will notify when
 recalling the archive is: "+topic_arn)
    job_id = initiate_archive_retrieval(archive_id, topic_arn)
else:
    print("Retrieve archive is false, polling queues and downloading only.")

while True:
   ReceiveArchiveReadyMessages(notify_queue_url,chunk_queue_url)
   run_async_tasks(chunk_queue_url,workers)
```

スクリプトの使用

このスクリプトを使用するには、次の手順に従います。

1. スクリプトのプレースホルダー値を特定の情報に置き換えます。
   - *output_file_path*: チャンクファイルが保存されるディレクトリ
   - *vault_name*: S3 Glacier ボールトの名前
   - *notify_queue_name*: ジョブ通知キューの名前
   - *chunk_download_queue_name*: チャンクダウンロードキューの名前
   - *sns_topic_name*: SNS トピックの名前
   - *region*: ボールトがある AWS リージョン
   - *archive_id*: 取得するアーカイブの ID
2. スクリプトを実行します。

```
python download_large_archive.py
```

3. すべてのチャンクをダウンロードしたら、次のようなコマンドを使用して 1 つのファイルにまとめることができます。

```
cat /path/to/chunks/*.chunk > complete_archive.file
```

重要な考慮事項

このスクリプトを使用する場合は、次の点に注意してください。

- 選択した取得階層によっては、S3 Glacier からのアーカイブの取得が完了するまでに数時間かかる場合があります。
- スクリプトは無期限に実行され、キューを継続的にポーリングします。特定の要件に基づいて終了条件を追加することもできます。
- アーカイブのすべてのチャンクを保存するのに十分なディスク容量があることを確認します。
- スクリプトが中断された場合は、 で再起動`retrieve_archive=False`して、新しい取得ジョブを開始せずにチャンクのダウンロードを続行できます。

- ネットワーク帯域幅とシステムリソースに基づいて、*chunk_size* パラメータと *workers* パラメータを調整します。
- Amazon S3 の取得、Amazon SNS、Amazon SQS の使用には、標準 AWS 料金が適用されます。

REST API を使用したアーカイブのダウンロード

REST API を使用してアーカイブをダウンロードする方法

アーカイブをダウンロードするプロセスは、2 つのステップに分けることができます。

1. `archive-retrieval` タイプのジョブを開始します。詳細については、「ジョブの開始 (ジョブの POST)」を参照してください。
2. ジョブが完了したら、アーカイブデータをダウンロードします。詳細については、「ジョブの出力の取得 (GET output)」を参照してください。

を使用した Amazon S3 Glacier でのアーカイブのダウンロード AWS CLI

AWS Command Line Interface () を使用して、Amazon S3 Glacier (S3 Glacier) でアーカイブをダウンロードできますAWS CLI。

トピック

- (前提条件) のセットアップ AWS CLI
- 例: を使用してアーカイブをダウンロードする AWS CLI

(前提条件) のセットアップ AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

   のインストール AWS Command Line Interface

   の設定 AWS Command Line Interface
2. コマンドプロンプトで次のコマンドを入力して、 AWS CLI セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを使用してください。

```
aws help
```

- 設定したアカウントの S3 Glacier ボールトのリストを取得するには、`list-vaults` コマンドを使用します。*123456789012* 「」を AWS アカウント ID に置き換えます。

```
aws glacier list-vaults --account-id 123456789012
```

- の現在の設定データを表示するには AWS CLI、 `aws configure list` コマンドを使用します。

```
aws configure list
```

例: を使用してアーカイブをダウンロードする AWS CLI

> **Note**
> アーカイブをダウンロードするには、アーカイブ ID がわかっている必要があります。ステップ 1～4 でアーカイブ ID を取得します。ダウンロードするアーカイブ ID がすでにわかっている場合は、ステップ 5 に進みます。

1. インベントリ取得ジョブを開始するには、`initiate-job` コマンドを使用します。インベントリレポートにアーカイブ ID が一覧表示されます。

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
job-parameters="{\"Type\":\"inventory-retrieval\"}"
```

正常な出力:

```
{
    "location": "/111122223333/vaults/awsexamplevault/jobs/*** jobid ***",
    "jobId": "*** jobid ***"
}
```

2. 以前の ジョブのステータスをチェックするには、`describe-job` コマンドを使用します。

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

正常な出力:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "*** vault arn ***",
    "Completed": false,
    "JobId": "*** jobid ***",
    "Action": "InventoryRetrieval",
    "CreationDate": "*** job creation date ***",
    "StatusCode": "InProgress"
}
```

3. ジョブが完了するまで待ちます。

   ジョブの出力をダウンロードする準備が整うまで待つ必要があります。ボールトに通知設定を指定している場合、またはジョブを開始したときに Amazon Simple Notification Service (Amazon SNS) トピックを指定している場合は、ジョブの完了後に S3 Glacier からそのトピックにメッセージが送信されます。

   ボールトに特定のイベントに対する通知設定を指定できます。詳細については、「Amazon S3 Glacier でのボールト通知の設定」を参照してください。S3 Glacier は、特定のイベントが発生するたびに、指定された SNS トピックにメッセージを送信します。

4. 完了したら、`get-job-output` コマンドを使用して、取得ジョブをファイル `output.json` にダウンロードします。このファイルにアーカイブ ID が含まれます。

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output.json
```

このコマンドは、次のフィールドを含むファイルを生成します。

```
{
"VaultARN":"arn:aws:glacier:region:111122223333:vaults/awsexamplevault",
```

```
"InventoryDate":"*** job completion date ***",
"ArchiveList":[
{"ArchiveId":"*** archiveid ***",
"ArchiveDescription":*** archive description (if set) ***,
"CreationDate":"*** archive creation date ***",
"Size":"*** archive size (in bytes) ***",
"SHA256TreeHash":"*** archive hash ***"
}
{"ArchiveId":
...
]}
```

5. `initiate-job` コマンドを使用して、ボールトから各アーカイブを取得するプロセスを開始します。以下に示すようにジョブパラメータを `archive-retrieval` と指定する必要があります。

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333
 --job-parameters="{\"Type\":\"archive-retrieval\",\"ArchiveId\":\"*** archiveId
 ***\"}"
```

6. `archive-retrieval` ジョブが完了するまで待ちます。以前のコマンドのステータスをチェックするには、`describe-job` コマンドを使用します。

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

7. 上記のジョブが完了したら、`get-job-output` コマンドを使用してアーカイブをダウンロードします。

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output_file_name
```

Amazon S3 Glacier でアーカイブを削除する

Amazon S3 Glacier (S3 Glacier) 管理コンソールを使用してアーカイブを削除することはできません。アーカイブを削除するには、 AWS Command Line Interface (CLI) を使用するか、コードを記述して、REST API を直接使用するか、 AWS SDK for Java および .NET ラッパーライブラリを使用して削除リクエストを行う必要があります。以下のトピックでは、 AWS SDK for Java および .NET ラッパーライブラリ、REST API、および を使用する方法について説明します AWS CLI。

トピック

- を使用した Amazon S3 Glacier でのアーカイブの削除 AWS SDK for Java
- を使用した Amazon S3 Glacier でのアーカイブの削除 AWS SDK for .NET
- REST API を使用した Amazon S3 Glacier アーカイブの削除
- AWS Command Line Interfaceを使用して Amazon S3 Glacier でアーカイブを削除する

ボールトから削除できるアーカイブは一度に 1 つです。アーカイブを削除するには、削除リクエストにアーカイブ ID を指定する必要があります。アーカイブ ID は、そのアーカイブを含むボールトのボールトインベントリをダウンロードすることで取得できます。ボールトインベントリのダウンロードの詳細については、「Amazon S3 Glacier でボールトインベントリをダウンロードする」を参照してください。

アーカイブを削除した後でも、削除したアーカイブの取得ジョブを開始することはリクエストできますが、アーカイブの取得ジョブ自体は失敗します。

アーカイブを削除する際に、該当するアーカイブ ID のアーカイブが取得中であった場合、取得は以下のシナリオに応じて成功する場合と成功しない場合があります。

- S3 Glacier がアーカイブの削除リクエストを受け取ったときに、アーカイブの取得ジョブがダウンロード用のデータを準備している最中であった場合には、アーカイブの取得オペレーションが失敗することがあります。
- S3 Glacier がアーカイブの削除リクエストを受け取ったときに、アーカイブの取得ジョブがダウンロード対象のアーカイブの準備を完了していた場合には、出力をダウンロードできます。

アーカイブの取得に関する詳細については、「S3 Glacier でのアーカイブのダウンロード」を参照してください。

このオペレーションはべき等です。既に削除されたアーカイブを削除しようとした場合には、エラーは発生しません。

アーカイブの削除後、すぐにボールトインベントリをダウンロードした場合、リストに削除したアーカイブが含まれることがあります。S3 Glacier がボールトインベントリを更新するのは、約 1 日 1 回であるためです。

> **Note**
>
> ボールトアーカイブの自動削除については、Amazon S3 Glacier でのボールトアーカイブの自動削除」を参照してください。

を使用した Amazon S3 Glacier でのアーカイブの削除 AWS SDK for Java

以下は、 AWS SDK for Java 低レベル API を使用してアーカイブを削除するステップです。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   削除するアーカイブが保存される AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `DeleteArchiveRequest` クラスのインスタンスを作成することにより、リクエスト情報を指定します。

   アーカイブ ID、ボールト名、およびアカウント ID を指定する必要があります。アカウント ID を指定しなかった場合は、リクエストに署名する際に指定した認証情報に関連づけられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3 」を参照してください。

3. リクエストオブジェクトをパラメータとして指定して、`deleteArchive` メソッドを実行します。

以下の Java コードスニペットは、前述の手順を示しています。

```
AmazonGlacierClient client;

DeleteArchiveRequest request = new DeleteArchiveRequest()
    .withVaultName("*** provide a vault name ***")
    .withArchiveId("*** provide an archive ID ***");

client.deleteArchive(request);
```

> **Note**
>
> 基本となる REST API については、「アーカイブの削除 (DELETE archive)」を参照してください。

例: を使用したアーカイブの削除 AWS SDK for Java

次の Java コード例では、 AWS SDK for Java を使用してアーカイブを削除します。この例を実行するための詳しい手順については、「Eclipse を使用した Amazon S3 Glacier の Java 実行例」を参照してください。ここに示したコードは、ボールトの名前と、削除するアーカイブのアーカイブ ID で更新する必要があります。

Example

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.glacier.AmazonGlacierClient;
import com.amazonaws.services.glacier.model.DeleteArchiveRequest;

public class ArchiveDelete {

    public static String vaultName = "*** provide vault name ****";
    public static String archiveId = "*** provide archive ID***";
    public static AmazonGlacierClient client;

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

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();

        client = new AmazonGlacierClient(credentials);
        client.setEndpoint("https://glacier.us-east-1.amazonaws.com/");

        try {

            // Delete the archive.
            client.deleteArchive(new DeleteArchiveRequest()
                .withVaultName(vaultName)
                .withArchiveId(archiveId));

            System.out.println("Deleted archive successfully.");
```

```
        } catch (Exception e) {
            System.err.println("Archive not deleted.");
            System.err.println(e);
        }
    }
}
```

を使用した Amazon S3 Glacier でのアーカイブの削除 AWS SDK for .NET

両方高レベル API と低レベル API.NET 用の Amazon SDK で提供されており、アーカイブを削除する方法を提供します。

トピック

- の高レベル API を使用したアーカイブの削除 AWS SDK for .NET
- 低レベル API を使用したアーカイブの削除 AWS SDK for .NET

の高レベル API を使用したアーカイブの削除 AWS SDK for .NET

高レベル API の `ArchiveTransferManager` クラスには、アーカイブの削除に使用できる `DeleteArchive` メソッドが用意されています。

例: の高レベル API を使用したアーカイブの削除 AWS SDK for .NET

次の C# コード例では、 の高レベル API AWS SDK for .NET を使用してアーカイブを削除します。この例を実行するための詳しい手順については、「コード例の実行」を参照してください。ここに示したコードは、削除するアーカイブのアーカイブ ID で、更新する必要があります。

Example

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveDeleteHighLevel
  {
```

```
    static string vaultName = "examplevault";
    static string archiveId = "*** Provide archive ID ***";

    public static void Main(string[] args)
    {
      try
      {
        var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
        manager.DeleteArchive(vaultName, archiveId);
        Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }
  }
}
```

低レベル API を使用したアーカイブの削除 AWS SDK for .NET

以下に、 AWS SDK for .NETを使用してアーカイブを削除する手順を示します。

1. `AmazonGlacierClient` クラスのインスタンス (クライアント) を作成します。

   削除するアーカイブが保存される AWS リージョンを指定する必要があります。このクライアントを使用して実行するすべてのオペレーションは、その AWS リージョンに適用されます。

2. `DeleteArchiveRequest` クラスのインスタンスを作成することにより、リクエスト情報を指定します。

   アーカイブ ID、ボールト名、およびアカウント ID を指定する必要があります。アカウント ID を指定しなかった場合は、リクエストに署名する際に指定した認証情報に関連づけられているアカウント ID が使用されます。詳細については、「Amazon S3 Glacier での AWS SDKs の使用 Amazon S3 」を参照してください。

3. リクエストオブジェクトをパラメータとして指定して、`DeleteArchive` メソッドを実行します。

例: の低レベル API を使用したアーカイブの削除 AWS SDK for .NET

以下の C# の例は、前述の手順を示しています。この例では、 の低レベル API AWS SDK for .NET を使用してアーカイブを削除します。

> **Note**
>
> 基本となる REST API については、「アーカイブの削除 (DELETE archive)」を参照してください。

この例を実行するための詳しい手順については、「コード例の実行」を参照してください。ここに示したコードは、削除するアーカイブのアーカイブ ID で、更新する必要があります。

Example

```
using System;
using Amazon.Glacier;
using Amazon.Glacier.Model;
using Amazon.Runtime;

namespace glacier.amazon.com.docsamples
{
  class ArchiveDeleteLowLevel
  {
    static string vaultName = "examplevault";
    static string archiveId = "*** Provide archive ID ***";

    public static void Main(string[] args)
    {
      AmazonGlacierClient client;
      try
      {
        using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2))
        {
          Console.WriteLine("Deleting the archive");
          DeleteAnArchive(client);
        }
        Console.WriteLine("Operations successful. To continue, press Enter");
        Console.ReadKey();
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
```

```
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static void DeleteAnArchive(AmazonGlacierClient client)
    {
      DeleteArchiveRequest request = new DeleteArchiveRequest()
      {
        VaultName = vaultName,
        ArchiveId = archiveId
      };
      DeleteArchiveResponse response = client.DeleteArchive(request);
    }
  }
}
```

REST API を使用した Amazon S3 Glacier アーカイブの削除

アーカイブ削除 API を使用して、アーカイブを削除できます。

- アーカイブ削除 API の詳細については、「アーカイブの削除 (DELETE archive)」を参照してください。
- REST API の使用については、「Amazon S3 Glacier の API リファレンス」を参照してください。

AWS Command Line Interfaceを使用して Amazon S3 Glacier でアーカイブを削除する

AWS Command Line Interface () を使用して、Amazon S3 Glacier (S3 Glacier) のアーカイブを削除できますAWS CLI。

トピック

- (前提条件) のセットアップ AWS CLI
- 例: を使用したアーカイブの削除 AWS CLI

(前提条件) のセットアップ AWS CLI

1. AWS CLIをダウンロードして設定します。手順については、「AWS Command Line Interface ユーザーガイド」の次のトピックを参照してください。

のインストール AWS Command Line Interface

の設定 AWS Command Line Interface

2. コマンドプロンプトで次のコマンドを入力して、 AWS CLI セットアップを確認します。これらのコマンドは、いずれも認証情報を明示的に提供しないため、デフォルトプロファイルの認証情報が使用されます。

   - help コマンドを使用してください。

   ```
   aws help
   ```

   - 設定したアカウントの S3 Glacier ボールトのリストを取得するには、`list-vaults` コマンドを使用します。*123456789012* 「」を AWS アカウント ID に置き換えます。

   ```
   aws glacier list-vaults --account-id 123456789012
   ```

   - の現在の設定データを表示するには AWS CLI、 `aws configure list` コマンドを使用します。

   ```
   aws configure list
   ```

例: を使用したアーカイブの削除 AWS CLI

1. インベントリ取得ジョブを開始するには、initiate-job コマンドを使用します。

   ```
   aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333 --
   job-parameters="{\"Type\":\"inventory-retrieval\"}"
   ```

   正常な出力:

   ```
   {
       "location": "/111122223333/vaults/awsexamplevault/jobs/*** jobid ***",
       "jobId": "*** jobid ***"
   }
   ```

2. 以前の取得ジョブのステータスをチェックするには、describe-job コマンドを使用します。

```
aws glacier describe-job --vault-name awsexamplevault --account-id 111122223333 --
job-id *** jobid ***
```

正常な出力:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "*** vault arn ***",
    "Completed": false,
    "JobId": "*** jobid ***",
    "Action": "InventoryRetrieval",
    "CreationDate": "*** job creation date ***",
    "StatusCode": "InProgress"
}
```

3. ジョブが完了するまで待ちます。

   ジョブの出力をダウンロードする準備が整うまで待つ必要があります。ボールトに通知設定を指定している場合、またはジョブを開始したときに Amazon Simple Notification Service (Amazon SNS) トピックを指定している場合は、ジョブの完了後に S3 Glacier からそのトピックにメッセージが送信されます。

   ボールトに特定のイベントに対する通知設定を指定できます。詳細については、「Amazon S3 Glacier でのボールト通知の設定」を参照してください。S3 Glacier は、特定のイベントが発生するたびに、指定された SNS トピックにメッセージを送信します。

4. 完了したら、get-job-output コマンドを使用して、取得ジョブをファイル `output.json` にダウンロードします。

```
aws glacier get-job-output --vault-name awsexamplevault --account-id 111122223333
 --job-id *** jobid *** output.json
```

このコマンドは、次のフィールドを含むファイルを生成します。

```
{
```

```
"VaultARN":"arn:aws:glacier:region:111122223333:vaults/awsexamplevault",
"InventoryDate":"*** job completion date ***",
"ArchiveList":[
{"ArchiveId":"*** archiveid ***",
"ArchiveDescription":*** archive description (if set) ***,
"CreationDate":"*** archive creation date ***",
"Size":"*** archive size (in bytes) ***",
"SHA256TreeHash":"*** archive hash ***"
}
{"ArchiveId":
...
]}
```

5. `delete-archive` コマンドを使用して、ボールトから各アーカイブを削除します。

```
aws glacier delete-archive --vault-name awsexamplevault --account-id 111122223333
 --archive-id *** archiveid ***
```

Amazon S3 Glacier での AWS SDKs の使用 Amazon S3

AWS はSDKs を提供します。 Amazon S3 SDK ライブラリは基本となる S3 Glacier API をラップし、プログラミングタスクを簡素化します。たとえば、S3 Glacier に送信する各リクエストに対して、リクエストを認証するために署名を含める必要があります。SDK ライブラリを使用する場合は、コードに AWS セキュリティ認証情報のみを指定する必要があります。ライブラリは必要な署名を計算し、S3 Glacier に送信されるリクエストに含めます。 AWS SDKs は、基盤となる REST API にマッピングするライブラリを提供し、リクエストを簡単に構築してレスポンスを処理するために使用できるオブジェクトを提供します。

トピック

- AWS Java および .NET 用の SDK ライブラリ
- AWS SDK での S3 Glacier の使用
- Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3
- Amazon S3 Glacier AWS SDK for .NET での の使用 Amazon S3

AWS Command Line Interface （ AWS CLI) は、S3 Glacier を含む AWS のサービスを管理するための統合ツールです。のダウンロードについては AWS CLI、「」を参照してくださいAWS Command Line Interface。S3 Glacier CLI コマンドのリストについては、「AWS CLI コマンドリファレンス」を参照してください。

AWS Java および .NET 用の SDK ライブラリ

Java および .NET 用 AWS SDKs は、高レベルおよび低レベルのラッパーライブラリを提供します。

このデベロッパーガイド AWS SDK for .NET 全体で AWS SDK for Java と を使用して、Amazon S3 Glacier の使用例を確認できます。

低レベル API とは

低レベルのラッパーライブラリは、S3 Glacier でサポートされる基本となる REST API (Amazon S3 Glacier の API リファレンス) に厳密にマップしています。低レベル API では、各 S3 Glacier REST オペレーションに対して、対応するメソッド、リクエスト情報を指定するリクエストオブジェクト、および S3 Glacier レスポンスを処理するレスポンスオブジェクトを提供します。低レベルのラッパーライブラリは、基本となる S3 Glacier オペレーションの最も完全な実装です。

これらの SDK ライブラリの詳細については、「Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3 」および「Amazon S3 Glacier AWS SDK for .NET での の使用 Amazon S3 」を参照してください。

高レベル API とは

アプリケーション開発をさらに簡素化するために、これらのライブラリでは一部のオペレーションに対して高レベルの抽象化を提供します。例:

- アーカイブのアップロード - 低レベル API を使用して、ファイル名およびアーカイブの保存先となるボールトの名前とともにアーカイブをアップロードする場合は、ペイロードのチェックサム (SHA-256 木構造ハッシュ) を指定する必要があります。一方、高レベル API ではチェックサムが自動的に計算されます。
- アーカイブまたはボールトインベントリのダウンロード - 低レベル API を使用して、アーカイブをダウンロードする場合、まずジョブを開始し、ジョブが完了するまで待機してから、ジョブの出力を取得します。S3 Glacier でジョブの完了時に通知されるように、Amazon Simple Notification Service (Amazon SNS) トピックを設定する追加コードを記述する必要があります。また、ジョブの完了メッセージがトピックに投稿されたかどうかを確認するポーリングのメカニズムも必要です。高レベル API には、これらのすべてのステップに対応した、アーカイブをダウンロードするためのメソッドが用意されています。必要な操作は、アーカイブ ID と、ダウンロードしたデータを保存するフォルダーのパスを指定することのみです。

これらの SDK ライブラリの詳細については、「Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3 」および「Amazon S3 Glacier AWS SDK for .NET での の使用 Amazon S3 」を参照してください。

高レベル API と低レベル API を使用する場合

一般的に、オペレーションの実行に必要なメソッドが高レベル API に用意されている場合は、シンプルな高レベル API を使用してください。ただし、高レベル API でその機能が提供されていない場合は、低レベル API を使用できます。また、低レベル API では、エラー発生時の再試行ロジックなど、オペレーションを細かく制御できます。たとえば、アーカイブをアップロードするときに、高レベル API ではファイルのサイズに応じて、単一オペレーションでアーカイブをアップロードするか、マルチパートアップロード API を使用するかを決定します。APIには、アップロードが失敗した場合の再試行ロジックも組み込まれています。ただし、アプリケーションでこれらの決定を細かく制御する必要がある場合は、低レベル API を使用します。

AWS SDK での S3 Glacier の使用

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

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

S3 Glacier に固有の例については、「SDK を使用した S3 Glacier のコード例 AWS SDKs」を参照してください。

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

Amazon S3 Glacier AWS SDK for Java での の使用 Amazon S3

AWS SDK for Java は、APIs の両方を提供します Amazon S3 Glacier での AWS SDKs の使用 Amazon S3 。 Amazon S3 のダウンロードの詳細については AWS SDK for Java、「Amazon SDK for Java」を参照してください。

> **Note**
>
> AWS SDK for Java は、S3 Glacier にアクセスするためのスレッドセーフなクライアントを提供します。最善の方法としては、ご利用のアプリケーションでクライアントを 1 つ作成し、そのクライアントをスレッド間で再利用することです。

トピック

- 低レベル API の使用
- 高レベル API の使用
- Eclipse を使用した Amazon S3 Glacier の Java 実行例
- エンドポイントの設定

低レベル API の使用

低レベルの `AmazonGlacierClient` クラスには、S3 Glacier の基盤となる REST オペレーションに対応するメソッドがすべて用意されています (Amazon S3 Glacier の API リファレンス)。これらのメソッドを呼び出すときには、対応するリクエストオブジェクトを作成するとともに、そのメソッドがオペレーションに S3 Glacier のレスポンスを返すためのレスポンスオブジェクトを指定する必要があります。

たとえば、`AmazonGlacierClient` クラスには、ボールト作成のための `createVault` メソッドがあります。このメソッドは、ボールトの作成 REST オペレーションに対応するものです (「ボールトの作成 (PUT vault)」を参照してください)。このメソッドを使用するには、以下の Java コードスニ

ペットに示すように、S3 Glacier レスポンスを受け取る `CreateVaultResult` オブジェクトのインスタンスを作成する必要があります。

```
AmazonGlacierClient client = new AmazonGlacierClient(credentials);
client.setEndpoint("https://glacier.us-west-2.amazonaws.com/");

CreateVaultRequest request = new CreateVaultRequest()
    .withAccountId("-")
    .withVaultName(vaultName);
CreateVaultResult result = client.createVault(createVaultRequest);
```

このガイドの低レベル API の例では、すべてこのパターンを使用しています。

> **Note**
>
> ここに挙げたコードスニペットでは、リクエストの作成時に `AccountID` を指定しています。ただし、 を使用する場合 AWS SDK for Java、リクエスト`AccountId`の はオプションであるため、このガイドのすべての低レベル例ではこの値は設定されません。`AccountId` は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで「-」を指定できます。この場合、S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフンを含めないでください。を使用する場合 AWS SDK for Java、アカウント ID を指定しない場合、ライブラリはアカウント ID を「-」に設定します。

高レベル API の使用

アプリケーション開発をさらに簡素化するために、 AWS SDK for Java には、低レベル API の一部のメソッドに対して高レベルの抽象化を実装する `ArchiveTransferManager` クラスが用意されています。このクラスには、`upload`、`download` など、アーカイブオペレーションのための便利なメソッドがあります。

たとえば、以下の Java コードスニペットでは、アーカイブのアップロードに `upload` 高レベルメソッドを使用しています。

```
String vaultName = "examplevault";
String archiveToUpload = "c:/folder/exampleArchive.zip";
```

```
ArchiveTransferManager atm = new ArchiveTransferManager(client, credentials);
String archiveId = atm.upload(vaultName, "Tax 2012 documents", new
 File(archiveToUpload)).getArchiveId();
```

実行するオペレーションは、`ArchiveTransferManager`オブジェクトの作成時に指定した AWS リージョンに適用されることに注意してください。 AWS リージョンを指定しない場合、 はデフォルトの AWS リージョン`us-east-1`として AWS SDK for Java を設定します。

このガイドの高レベル API の例では、すべてこのパターンを使用しています。

> **Note**
>
> 高レベルの `ArchiveTransferManager` クラスは、`AmazonGlacierClient` インスタンスまたは `AWSCredentials` インスタンスを使用して構築できます。

Eclipse を使用した Amazon S3 Glacier の Java 実行例

Java コード例の使用を最も手早く開始する方法は、最新の AWS Toolkit for Eclipseをインストールすることです。最新ツールキットのインストールと更新については、http:///eclipse を参照してください。以下のタスクは、このセクションに示した Java コード例を作成およびテストする手順を示しています。

Java コード例作成の一般的な手順

1	Amazon SDK for Java の「認証情報の提供 AWS SDK for Java 」トピックの説明に従って、 AWS 認証情報のデフォルトの認証情報プロファイルを作成します。 AWS
2	Eclipse AWS で新しい Java プロジェクトを作成します。プロジェクトは AWS SDK for Java用にあらかじめ設定されています。
3	任意のセクションからプロジェクトにコードをコピーします。
4	必要なデータを指定してコードを修正します。たとえばファイルをアップロードする場合は、ファイルのパスとバケットの名前を指定します。
5	コードを実行します。 AWS Management Consoleを使用して、オブジェクトが作成されることを確認します。の詳細については AWS Management Console、「https://http://aws.amazon.com/console/.com」を参照してください。

エンドポイントの設定

デフォルトでは、 はエンドポイント AWS SDK for Java を使用しますhttps://glacier.us-east-1.amazonaws.com。以下の Java コードスニペットに示すように、エンドポイントは明示的に設定できます。

以下のスニペットには、低レベル API で 米国西部 (オレゴン リージョン (us-west-2) にエンドポイントを設定する方法を示しています。

Example

```
client = new AmazonGlacierClient(credentials);
client.setEndpoint("glacier.us-west-2.amazonaws.com");
```

以下のスニペットには、高レベル API で 米国西部(オレゴン) リージョンにエンドポイントを設定する方法を示しています。

```
glacierClient = new AmazonGlacierClient(credentials);
sqsClient = new AmazonSQSClient(credentials);
snsClient = new AmazonSNSClient(credentials);

glacierClient.setEndpoint("glacier.us-west-2.amazonaws.com");
sqsClient.setEndpoint("sqs.us-west-2.amazonaws.com");
snsClient.setEndpoint("sns.us-west-2.amazonaws.com");

ArchiveTransferManager atm = new ArchiveTransferManager(glacierClient, sqsClient,
 snsClient);
```

サポートされている AWS リージョンとエンドポイントのリストについては、「」を参照してくださいAmazon S3 Glacier へのアクセス。

Amazon S3 Glacier AWS SDK for .NET での の使用 Amazon S3

AWS SDK for .NET API は で使用できますAWSSDK.dll。のダウンロードについては AWS SDK for .NET、「サンプルコードライブラリ」を参照してください。で説明されているようにAmazon S3 Glacier での AWS SDKs の使用 Amazon S3 、 AWS SDK for .NET には高レベル API と低レベル APIs。

> **Note**
>
> 低レベル API と高レベル API により、S3 Glacier へのアクセスを目的としたスレッドセーフのクライアントが提供されます。最善の方法としては、ご利用のアプリケーションでクライアントを 1 つ作成し、そのクライアントをスレッド間で再利用することです。

トピック

- 低レベル API の使用
- 高レベル API の使用
- コード例の実行
- エンドポイントの設定

低レベル API の使用

低レベルの `AmazonGlacierClient` クラスには、Amazon S3 Glacier (S3 Glacier) の基盤となる REST オペレーションにマッピングするメソッドがすべて用意されています ()。これらのメソッドを呼び出すときには、対応するリクエストオブジェクトを作成するとともに、そのメソッドがオペレーションに S3 Glacier のレスポンスを返すためのレスポンスオブジェクトを指定する必要があります。

たとえば、`AmazonGlacierClient` クラスには、ボールト作成のための `CreateVault` メソッドがあります。このメソッドは、ボールトの作成 REST オペレーションに対応するものです (「ボールトの作成 (PUT vault)」を参照してください)。このメソッドを使用するには、次の C# コードスニペットに示すように、`CreateVaultRequest` クラスと `CreateVaultResponse` クラスのインスタンスを作成して、リクエスト情報を指定し、S3 Glacier のレスポンスを受け取る必要があります。

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1);

CreateVaultRequest request = new CreateVaultRequest()
{
  AccountId = "-",
  VaultName = "*** Provide vault name ***"
};
```

```
CreateVaultResponse response = client.CreateVault(request);
```

このガイドの低レベル API の例では、すべてこのパターンを使用しています。

> **Note**
>
> ここに挙げたコードスニペットでは、リクエストの作成時に `AccountId` を指定しています。ただし、 を使用する場合 AWS SDK for .NET、リクエスト`AccountId`の はオプションであるため、このガイドのすべての低レベルの例ではこの値を設定しません。`AccountId` は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで「-」を指定できます。この場合、S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフンを含めないでください。を使用する場合 AWS SDK for .NET、アカウント ID を指定しない場合、ライブラリはアカウント ID を「-」に設定します。

高レベル API の使用

アプリケーション開発をさらに簡素化するために、 AWS SDK for .NET には、低レベル API の一部のメソッドに対して高レベルの抽象化を実装する `ArchiveTransferManager` クラスが用意されています。このクラスには、`Upload`、`Download` など、アーカイブオペレーションのための便利なメソッドがあります。

たとえば、以下の C# コードスニペットでは、アーカイブのアップロードに `Upload` 高レベルメソッドを使用しています。

```
string vaultName = "examplevault";
string archiveToUpload = "c:\folder\exampleArchive.zip";

var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USEast1);
string archiveId = manager.Upload(vaultName, "archive description",
 archiveToUpload).ArchiveId;
```

実行するオペレーションは、`ArchiveTransferManager`オブジェクトの作成時に指定した AWS リージョンに適用されることに注意してください。このガイドの高レベル API の例では、すべてこのパターンを使用しています。

> **Note**
>
> 高レベルの `ArchiveTransferManager` クラスには、依然として低レベルの `AmazonGlacierClient` クライアントが必要です。このクライアントは、明示的に渡すことや、`ArchiveTransferManager` で作成することができます。

コード例の実行

.NET コード例の使用を最も手早く開始する方法は、 AWS SDK for .NETをインストールすることです。詳細については、「Amazon SDK for .NET」を参照してください。

以下の手順では、このガイドに示しているコード例をテストするためのステップを示しています。

.NET コード例作成の一般的な手順 (Visual Studio 使用)

1	Amazon SDK for .NET トピック「認証情報の設定」の説明に従って、 AWS 認証情報の認証情報プロファイルを作成します。 AWS
2	AWS 空の プロジェクト テンプレートを使用して、新しい Visual Studio プロジェクトを作成します。
3	プロジェクトファイル `Program.cs` 内のコードを、任意のセクションのコードで置き換えます。
4	コードを実行します。 AWS Management Consoleを使用して、オブジェクトが作成されることを確認します。詳細については AWS Management Console、http://aws.amazon.com/console/ を参照してください。

エンドポイントの設定

デフォルトでは、 はエンドポイントを米国西部 (オレゴン) リージョン () AWS SDK for .NET に設定しますhttps://glacier.us-west-2.amazonaws.com。次の C# スニペットに示すように、エンドポイントを他の AWS リージョンに設定できます。

以下のスニペットには、低レベル API で 米国西部 (オレゴン リージョン (us-west-2) にエンドポイントを設定する方法を示しています。

Example

```
AmazonGlacierClient client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);
```

以下のスニペットには、高レベル API で 米国西部(オレゴン) リージョンにエンドポイントを設定する方法を示しています。

```
var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
```

サポートされている AWS リージョンとエンドポイントの現在のリストについては、「」を参照してくださいAmazon S3 Glacier へのアクセス。

SDK を使用した S3 Glacier のコード例 AWS SDKs

次のコード例は、 AWS Software Development Kit (SDK) で S3 Glacier を使用する方法を示しています。

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

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK での S3 Glacier の使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

開始方法

Hello Amazon S3 Glacier

次のコード例は、Amazon S3 Glacier 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。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using Amazon.Glacier;
using Amazon.Glacier.Model;

namespace GlacierActions;

public static class HelloGlacier
{
    static async Task Main()
```

```
    {
        var glacierService = new AmazonGlacierClient();

        Console.WriteLine("Hello Amazon Glacier!");
        Console.WriteLine("Let's list your Glacier vaults:");

        // You can use await and any of the async methods to get a response.
        // Let's get the vaults using a paginator.
        var glacierVaultPaginator = glacierService.Paginators.ListVaults(
            new ListVaultsRequest { AccountId = "-" });

        await foreach (var vault in glacierVaultPaginator.VaultList)
        {
            Console.WriteLine($"{vault.CreationDate}:{vault.VaultName}, ARN:
{vault.VaultARN}");
        }
    }
}
```

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

コードの例

- SDK を使用した S3 Glacier の基本的な例 AWS SDKs
  - Hello Amazon S3 Glacier
  - SDK を使用した S3 Glacier のアクション AWS SDKs
    - AWS SDK または CLI AddTagsToVaultで を使用する
    - AWS SDK または CLI CreateVaultで を使用する
    - AWS SDK または CLI DeleteArchiveで を使用する
    - AWS SDK または CLI DeleteVaultで を使用する
    - AWS SDK または CLI DeleteVaultNotificationsで を使用する
    - AWS SDK または CLI DescribeJobで を使用する
    - AWS SDK または CLI DescribeVaultで を使用する
    - AWS SDK または CLI GetJobOutputで を使用する
    - AWS SDK または CLI GetVaultNotificationsで を使用する
    - AWS SDK または CLI InitiateJobで を使用する

    - AWS SDK または CLI ListJobsで を使用する
    - AWS SDK または CLI ListTagsForVaultで を使用する
    - AWS SDK または CLI ListVaultsで を使用する
    - AWS SDK または CLI SetVaultNotificationsで を使用する
    - AWS SDK または CLI UploadArchiveで を使用する
    - AWS SDK または CLI UploadMultipartPartで を使用する
- SDK を使用した S3 Glacier のシナリオ AWS SDKs
  - AWS SDK を使用してファイルを Amazon S3 Glacier にアーカイブし、通知を取得し、ジョブを開始する
  - AWS SDK を使用して Amazon S3 Glacier アーカイブコンテンツを取得し、アーカイブを削除する

SDK を使用した S3 Glacier の基本的な例 AWS SDKs

以下のコード例は、 AWS SDK での Amazon S3 Glacier の基本的な使用方法を示しています。

例

- Hello Amazon S3 Glacier
- SDK を使用した S3 Glacier のアクション AWS SDKs
  - AWS SDK または CLI AddTagsToVaultで を使用する
  - AWS SDK または CLI CreateVaultで を使用する
  - AWS SDK または CLI DeleteArchiveで を使用する
  - AWS SDK または CLI DeleteVaultで を使用する
  - AWS SDK または CLI DeleteVaultNotificationsで を使用する
  - AWS SDK または CLI DescribeJobで を使用する
  - AWS SDK または CLI DescribeVaultで を使用する
  - AWS SDK または CLI GetJobOutputで を使用する
  - AWS SDK または CLI GetVaultNotificationsで を使用する
  - AWS SDK または CLI InitiateJobで を使用する
  - AWS SDK または CLI ListJobsで を使用する
  - AWS SDK または CLI ListTagsForVaultで を使用する
  - AWS SDK または CLI ListVaultsで を使用する

- AWS SDK または CLI SetVaultNotificationsで を使用する
- AWS SDK または CLI UploadArchiveで を使用する
- AWS SDK または CLI UploadMultipartPartで を使用する

Hello Amazon S3 Glacier

次のコード例は、Amazon S3 Glacier 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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using Amazon.Glacier;
using Amazon.Glacier.Model;

namespace GlacierActions;

public static class HelloGlacier
{
    static async Task Main()
    {
        var glacierService = new AmazonGlacierClient();

        Console.WriteLine("Hello Amazon Glacier!");
        Console.WriteLine("Let's list your Glacier vaults:");

        // You can use await and any of the async methods to get a response.
        // Let's get the vaults using a paginator.
        var glacierVaultPaginator = glacierService.Paginators.ListVaults(
            new ListVaultsRequest { AccountId = "-" });

        await foreach (var vault in glacierVaultPaginator.VaultList)
        {
```

```
            Console.WriteLine($"{vault.CreationDate}:{vault.VaultName}, ARN:
{vault.VaultARN}");
        }
    }
}
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK を使用した S3 Glacier のアクション AWS SDKs

次のコード例は、 AWS SDKs で個々の S3 Glacier アクションを実行する方法を示しています。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

これらは S3 Glacier 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 SDK を使用した S3 Glacier のシナリオ AWS SDKs のコンテキスト内で確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細な一覧については、「Amazon S3 API リファレンス」を参照してください。

例

- AWS SDK または CLI AddTagsToVaultで を使用する
- AWS SDK または CLI CreateVaultで を使用する
- AWS SDK または CLI DeleteArchiveで を使用する
- AWS SDK または CLI DeleteVaultで を使用する
- AWS SDK または CLI DeleteVaultNotificationsで を使用する
- AWS SDK または CLI DescribeJobで を使用する
- AWS SDK または CLI DescribeVaultで を使用する
- AWS SDK または CLI GetJobOutputで を使用する

- AWS SDK または CLI GetVaultNotificationsで を使用する
- AWS SDK または CLI InitiateJobで を使用する
- AWS SDK または CLI ListJobsで を使用する
- AWS SDK または CLI ListTagsForVaultで を使用する
- AWS SDK または CLI ListVaultsで を使用する
- AWS SDK または CLI SetVaultNotificationsで を使用する
- AWS SDK または CLI UploadArchiveで を使用する
- AWS SDK または CLI UploadMultipartPartで を使用する

AWS SDK または CLI **AddTagsToVault**で を使用する

次のサンプルコードは、AddTagsToVault 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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// Add tags to the items in an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the vault to add tags to.</param>
    /// <param name="key">The name of the object to tag.</param>
    /// <param name="value">The tag value to add.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddTagsToVaultAsync(string vaultName, string key,
string value)
    {
        var request = new AddTagsToVaultRequest
        {
            Tags = new Dictionary<string, string>
                {
                    { key, value },
```

```
                },
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.AddTagsToVaultAsync(request);
        return response.HttpStatusCode == HttpStatusCode.NoContent;
    }
```

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

CLI

AWS CLI

次のコマンドは、`my-vault` という名前のボールトに 2 つのタグを追加します。

```
aws glacier add-tags-to-vault --account-id - --vault-name my-vault --
tags id=1234,date=july2015
```

Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **`CreateVault`**で を使用する

次のサンプルコードは、`CreateVault` 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。

- ファイルのアーカイブ、通知の取得、ジョブの開始

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/// <summary>
/// Create an Amazon S3 Glacier vault.
/// </summary>
/// <param name="vaultName">The name of the vault to create.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateVaultAsync(string vaultName)
{
    var request = new CreateVaultRequest
    {
        // Setting the AccountId to "-" means that
        // the account associated with the current
        // account will be used.
        AccountId = "-",
        VaultName = vaultName,
    };

    var response = await _glacierService.CreateVaultAsync(request);

    Console.WriteLine($"Created {vaultName} at: {response.Location}");

    return response.HttpStatusCode == HttpStatusCode.Created;
}
```

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

CLI

AWS CLI

次のコマンドでは、`my-vault` という名前の新しいボールトが作成されます。

```
aws glacier create-vault --vault-name my-vault --account-id -
```

Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

- API の詳細については、「AWS CLI コマンドリファレンス」の「CreateVault」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.CreateVaultRequest;
import software.amazon.awssdk.services.glacier.model.CreateVaultResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;

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

```
        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to create.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void createGlacierVault(GlacierClient glacier, String
 vaultName) {
        try {
            CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            CreateVaultResponse createVaultResult =
 glacier.createVault(vaultRequest);
            System.out.println("The URI of the new vault is " +
 createVaultResult.location());

        } catch (GlacierException 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「CreateVault」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

クライアントを作成する

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

ボールトを作成します。

```
// Load the SDK for JavaScript
import { CreateVaultCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME
const params = { vaultName: vaultname };

const run = async () => {
  try {
    const data = await glacierClient.send(new CreateVaultCommand(params));
    console.log("Success, vault created!");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error");
  }
```

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

- 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CreateVaultを参照してください。

SDK for JavaScript (v2)

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
// Call Glacier to create the vault
glacier.createVault({ vaultName: "YOUR_VAULT_NAME" }, function (err) {
  if (!err) {
    console.log("Created vault!");
  }
});
```

- 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CreateVaultを参照してください。

PowerShell

Tools for PowerShell V4

例 1: ユーザーアカウント用に新しいボールトを作成します。-AccountId パラメータに値が指定されていないため、コマンドレットは現在のアカウントを示すデフォルトの「-」を使用します。

```
New-GLCVault -VaultName myvault
```

出力:

```
/01234567812/vaults/myvault
```

- 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CreateVault」を参照してください。

Tools for PowerShell V5

例 1: ユーザーアカウント用に新しいボールトを作成します。-AccountId パラメータに値が指定されていないため、コマンドレットは現在のアカウントを示すデフォルトの「-」を使用します。

```
New-GLCVault -VaultName myvault
```

出力:

```
/01234567812/vaults/myvault
```

- 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CreateVault」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteArchive**で を使用する

次のサンプルコードは、`DeleteArchive` 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。

- アーカイブコンテンツの取得とアーカイブの削除

CLI

AWS CLI

ボールトからアーカイブを削除するには

次の `delete-archive` の例では、`example_vault` から指定されたアーカイブを削除します。

```
aws glacier delete-archive \
    --account-id 111122223333  \
    --vault-name example_vault \
    --archive-id Sc0u9ZP8yaWkmh-XGlIvAVprtLhaLCGnNwNl5I5x9HqPIkX5mjc0DrId3Ln-
Gi_k2HzmlIDZUz117KSdVMdMXLuFWi9PJUitxWO73edQ43eTlMWkH0pd9zVSAuV_XXZBVhKhyGhJ7w
```

このコマンドは何も出力しません。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteArchive」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteArchiveRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * 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 DeleteArchive {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <vaultName> <accountId> <archiveId>

                Where:
                   vaultName - The name of the vault that contains the archive to
 delete.
                   accountId - The account ID value.
                   archiveId - The archive ID value.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String archiveId = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierArchive(glacier, vaultName, accountId, archiveId);
        glacier.close();
    }

    public static void deleteGlacierArchive(GlacierClient glacier, String
 vaultName, String accountId,
            String archiveId) {
        try {
            DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
                    .vaultName(vaultName)
                    .accountId(accountId)
                    .archiveId(archiveId)
                    .build();
```

```
            glacier.deleteArchive(delArcRequest);
            System.out.println("The archive was deleted.");

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

- 詳細については、「AWS SDK for Java 2.x API リファレンス」の「deleteArchive」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
```

```
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id,
archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteVault**で を使用する

次のサンプルコードは、DeleteVault 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。

- アーカイブコンテンツの取得とアーカイブの削除

CLI

AWS CLI

次のコマンドでは、my-vault という名前のボールトが削除されます。

```
aws glacier delete-vault --vault-name my-vault --account-id -
```

このコマンドでは、出力が生成されません。Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteVault」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DeleteVaultRequest;
import software.amazon.awssdk.services.glacier.model.GlacierException;

/**
 * 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 DeleteVault {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <vaultName>

                Where:
                   vaultName - The name of the vault to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        String vaultName = args[0];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteGlacierVault(glacier, vaultName);
        glacier.close();
    }

    public static void deleteGlacierVault(GlacierClient glacier, String
 vaultName) {
        try {
            DeleteVaultRequest delVaultRequest = DeleteVaultRequest.builder()
                    .vaultName(vaultName)
                    .build();

            glacier.deleteVault(delVaultRequest);
            System.out.println("The vault was deleted!");

        } catch (GlacierException 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「DeleteVault」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def delete_vault(vault):
        """
        Deletes a vault.

        :param vault: The vault to delete.
        """
        try:
            vault.delete()
            logger.info("Deleted vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteVaultNotifications**で を使用する

次のサンプルコードは、`DeleteVaultNotifications` を使用する方法を説明しています。

CLI

AWS CLI

ボールトの SNS 通知を削除するには

次の `delete-vault-notifications` の例は、指定されたボールトに対して Amazon Simple Notification Service (Amazon SNS) で送信される通知を削除します。

```
aws glacier delete-vault-notifications \
    --account-id 111122223333 \
    --vault-name example_vault
```

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「DeleteVaultNotifications」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def stop_notifications(notification):
        """
        Stops notifications to the configured Amazon SNS topic.

        :param notification: The notification configuration to remove.
        """
        try:
            notification.delete()
```

```
            logger.info("Notifications stopped.")
        except ClientError:
            logger.exception("Couldn't stop notifications.")
            raise
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeJob**で 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。

CLI

AWS CLI

次のコマンドは、my-vault という名前のボールトでのインベントリ取得ジョブに関する情報を取得します。

```
aws glacier describe-job --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW
```

出力:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
    "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-vault",
    "Completed": false,
    "JobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
    "Action": "InventoryRetrieval",
```

```
    "CreationDate": "2015-07-17T20:23:41.616Z",
    "StatusCode": "InProgress"
}
```

ジョブ ID は、`aws glacier initiate-job` と `aws glacier list-jobs` の出力にあります。Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

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

PowerShell

Tools for PowerShell V4

例 1: 指定したジョブの詳細を返します。ジョブが正常に完了したら、Read-GCJobOutput コマンドレットを使用して、ジョブの内容 (アーカイブまたはインベントリのリスト) をローカルファイルシステムに取得できます。

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

出力:

```
Action                       : ArchiveRetrieval
ArchiveId                    : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash        : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes           : 38034480
Completed                    : False
CompletionDate               : 1/1/0001 12:00:00 AM
CreationDate                 : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters :
InventorySizeInBytes         : 0
JobDescription               :
JobId                        : op1x...JSbthM
JobOutputPath                :
OutputLocation               :
RetrievalByteRange           : 0-38034479
SelectParameters             :
SHA256TreeHash               : 79f3ea754c02f58...dc57bf4395b
SNSTopic                     :
StatusCode                   : InProgress
```

```
StatusMessage                 :
Tier                          : Standard
VaultARN                      : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- 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DescribeJob」を参照してください。

Tools for PowerShell V5

例 1: 指定したジョブの詳細を返します。ジョブが正常に完了したら、Read-GCJobOutput コマンドレットを使用して、ジョブの内容 (アーカイブまたはインベントリのリスト) をローカルファイルシステムに取得できます。

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

出力:

```
Action                        : ArchiveRetrieval
ArchiveId                     : o9O9j...X-TpIhQJw
ArchiveSHA256TreeHash         : 79f3ea754c02f58...dc57bf4395b
ArchiveSizeInBytes            : 38034480
Completed                     : False
CompletionDate                : 1/1/0001 12:00:00 AM
CreationDate                  : 12/13/2018 11:00:14 AM
InventoryRetrievalParameters  :
InventorySizeInBytes          : 0
JobDescription                :
JobId                         : op1x...JSbthM
JobOutputPath                 :
OutputLocation                :
RetrievalByteRange            : 0-38034479
SelectParameters              :
SHA256TreeHash                : 79f3ea754c02f58...dc57bf4395b
SNSTopic                      :
StatusCode                    : InProgress
StatusMessage                 :
Tier                          : Standard
VaultARN                      : arn:aws:glacier:us-west-2:012345678912:vaults/test
```

- 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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_status(job):
        """
        Gets the status of a job.

        :param job: The job to query.
        :return: The current status of the job.
        """
        try:
            job.load()
            logger.info(
                "Job %s is performing action %s and has status %s.",
                job.id,
                job.action,
                job.status_code,
            )
        except ClientError:
            logger.exception("Couldn't get status for job %s.", job.id)
            raise
        else:
            return job.status_code
```

- 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。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeVault**で を使用する

次のサンプルコードは、DescribeVault 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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/// <summary>
/// Describe an Amazon S3 Glacier vault.
/// </summary>
/// <param name="vaultName">The name of the vault to describe.</param>
/// <returns>The Amazon Resource Name (ARN) of the vault.</returns>
public async Task<string> DescribeVaultAsync(string vaultName)
{
    var request = new DescribeVaultRequest
    {
        AccountId = "-",
        VaultName = vaultName,
    };

    var response = await _glacierService.DescribeVaultAsync(request);

    // Display the information about the vault.
    Console.WriteLine($"{response.VaultName}\tARN: {response.VaultARN}");
```

```
        Console.WriteLine($"Created on: {response.CreationDate}\tNumber
 of Archives: {response.NumberOfArchives}\tSize (in bytes):
 {response.SizeInBytes}");
        if (response.LastInventoryDate != DateTime.MinValue)
        {
            Console.WriteLine($"Last inventory: {response.LastInventoryDate}");
        }

        return response.VaultARN;
    }
```

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

CLI

AWS CLI

次のコマンドは、`my-vault` という名前のボールトに関するデータを取得します。

```
aws glacier describe-vault --vault-name my-vault --account-id -
```

Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **`GetJobOutput`**で を使用する

次のサンプルコードは、`GetJobOutput` 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。

- アーカイブコンテンツの取得とアーカイブの削除

CLI

AWS CLI

次のコマンドは、ボールトインベントリジョブの出力を、現在のディレクトリの output.json という名前のファイルに保存します。

```
aws glacier get-job-output --account-id - --vault-name my-vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW output.json
```

job-id は、aws glacier list-jobs の出力にあります。出力ファイル名はオプション名の前に付いていない位置引数であることに注意してください。Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

出力:

```
{
    "status": 200,
    "acceptRanges": "bytes",
    "contentType": "application/json"
}
```

output.json:

```
{"VaultARN":"arn:aws:glacier:us-west-2:0123456789012:vaults/
my-vault","InventoryDate":"2015-04-07T00:26:18Z","ArchiveList":
[{"ArchiveId":"kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw","ArchiveDescription":"multipart
 upload
 test","CreationDate":"2015-04-06T22:24:34Z","Size":3145728,"SHA256TreeHash":"9628195fcdb
```

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

PowerShell

Tools for PowerShell V4

例 1: 指定したジョブで取得がスケジュールされているアーカイブコンテンツをダウンロードし、コンテンツをディスク上のファイルに保存します。ダウンロード時に、チェックサムがある場合は自動的に検証されます。必要に応じて **-Select '*'** を指定すると、チェックサムを含むレスポンス全体を返すことができます。

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- API の詳細については、「 コマンドレットリファレンス (V4)」のGetJobOutput」を参照してください。 AWS Tools for PowerShell

Tools for PowerShell V5

例 1: 指定したジョブで取得がスケジュールされているアーカイブコンテンツをダウンロードし、コンテンツをディスク上のファイルに保存します。ダウンロード時に、チェックサムがある場合は自動的に検証されます。必要に応じて **-Select '*'** を指定すると、チェックサムを含むレスポンス全体を返すことができます。

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- 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GetJobOutput」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GlacierWrapper:
```

```
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
        else:
            return out_bytes
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetVaultNotifications**で を使用する

次のサンプルコードは、GetVaultNotifications 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、my-vault という名前のボールトの通知設定の説明を取得します。

```
aws glacier get-vault-notifications --account-id - --vault-name my-vault
```

出力:

```
{
    "vaultNotificationConfig": {
        "Events": [
            "InventoryRetrievalCompleted",
            "ArchiveRetrievalCompleted"
        ],
        "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault"
    }
}
```

ボールトに通知が設定されていない場合、エラーが返されます。Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetVaultNotifications」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def get_notification(vault):
        """
        Gets the currently notification configuration for a vault.

        :param vault: The vault to query.
        :return: The notification configuration for the specified vault.
        """
        try:
            notification = vault.Notification()
            logger.info(
                "Vault %s notifies %s on %s events.",
                vault.name,
                notification.sns_topic,
                notification.events,
            )
        except ClientError:
            logger.exception("Couldn't get notification data for %s.",
 vault.name)
            raise
        else:
            return notification
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **InitiateJob**で を使用する

次のサンプルコードは、InitiateJob 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。

- ファイルのアーカイブ、通知の取得、ジョブの開始

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ボールトからアーカイブを取得します。この例では ArchiveTransferManager クラスを使用しています。API の詳細については、「ArchiveTransferManager」を参照してください。

```
    /// <summary>
    /// Download an archive from an Amazon S3 Glacier vault using the Archive
    /// Transfer Manager.
    /// </summary>
    /// <param name="vaultName">The name of the vault containing the object.</
param>
    /// <param name="archiveId">The Id of the archive to download.</param>
    /// <param name="localFilePath">The local directory where the file will
    /// be stored after download.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> DownloadArchiveWithArchiveManagerAsync(string
 vaultName, string archiveId, string localFilePath)
    {
        try
        {
            var manager = new ArchiveTransferManager(_glacierService);

            var options = new DownloadOptions
            {
```

```
                StreamTransferProgress = Progress!,
            };

            // Download an archive.
            Console.WriteLine("Initiating the archive retrieval job and then
polling SQS queue for the archive to be available.");
            Console.WriteLine("When the archive is available, downloading will
begin.");
            await manager.DownloadAsync(vaultName, archiveId, localFilePath,
options);

            return true;
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Event handler to track the progress of the Archive Transfer Manager.
    /// </summary>
    /// <param name="sender">The object that raised the event.</param>
    /// <param name="args">The argument values from the object that raised the
    /// event.</param>
    static void Progress(object sender, StreamTransferProgressArgs args)
    {
        if (args.PercentDone != _currentPercentage)
        {
            _currentPercentage = args.PercentDone;
            Console.WriteLine($"Downloaded {_currentPercentage}%");
        }
    }
```

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

CLI

AWS CLI

次のコマンドは、ボールト my-vault のインベントリを取得するジョブを開始します。

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-parameters
 '{"Type": "inventory-retrieval"}'
```

出力:

```
{
    "location": "/0123456789012/vaults/my-vault/jobs/
zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
    "jobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW"
}
```

Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

次のコマンドは、ボールト my-vault からアーカイブを取得するジョブを開始します。

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-
parameters file://job-archive-retrieval.json
```

job-archive-retrieval.json はローカルフォルダ内の JSON ファイルで、ジョブのタイプ、アーカイブ ID、いくつかのオプションパラメータを指定する以下の内容を含みます。

```
{
  "Type": "archive-retrieval",
  "ArchiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
  "Description": "Retrieve archive on 2015-07-17",
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-topic"
}
```

アーカイブ ID は、aws glacier upload-archive および aws glacier get-job-output の出力で確認できます。

出力:

```
{
    "location": "/011685312445/vaults/mwunderl/jobs/l7IL5-
EkXyEY9Ws95fClzIbk2O5uLYaFdAYOi-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav",
    "jobId": "l7IL5-EkXy2O5uLYaFdAYOiEY9Ws95fClzIbk-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav"
}
```

ジョブパラメータの形式の詳細については、「Amazon Glacier API リファレンス」の「ジョブの開始」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「InitiateJob」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ボールトインベントリを取得します。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.JobParameters;
import software.amazon.awssdk.services.glacier.model.InitiateJobResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import software.amazon.awssdk.services.glacier.model.InitiateJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobRequest;
import software.amazon.awssdk.services.glacier.model.DescribeJobResponse;
import software.amazon.awssdk.services.glacier.model.GetJobOutputRequest;
import software.amazon.awssdk.services.glacier.model.GetJobOutputResponse;
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.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ArchiveDownload {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <vaultName> <accountId> <path>

                Where:
                   vaultName - The name of the vault.
                   accountId - The account ID value.
                   path - The path where the file is written to.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String vaultName = args[0];
        String accountId = args[1];
        String path = args[2];
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String jobNum = createJob(glacier, vaultName, accountId);
        checkJob(glacier, jobNum, vaultName, accountId, path);
        glacier.close();
    }

    public static String createJob(GlacierClient glacier, String vaultName,
 String accountId) {
        try {
```

```
            JobParameters job = JobParameters.builder()
                    .type("inventory-retrieval")
                    .build();

            InitiateJobRequest initJob = InitiateJobRequest.builder()
                    .jobParameters(job)
                    .accountId(accountId)
                    .vaultName(vaultName)
                    .build();

            InitiateJobResponse response = glacier.initiateJob(initJob);
            System.out.println("The job ID is: " + response.jobId());
            System.out.println("The relative URI path of the job is: " +
 response.location());
            return response.jobId();

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

        }
        return "";
    }

    // Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
    // Documentation.
    public static void checkJob(GlacierClient glacier, String jobId, String name,
 String account, String path) {
        try {
            boolean finished = false;
            String jobStatus;
            int yy = 0;

            while (!finished) {
                DescribeJobRequest jobRequest = DescribeJobRequest.builder()
                        .jobId(jobId)
                        .accountId(account)
                        .vaultName(name)
                        .build();

                DescribeJobResponse response = glacier.describeJob(jobRequest);
                jobStatus = response.statusCodeAsString();

                if (jobStatus.compareTo("Succeeded") == 0)
```

```
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + jobStatus);
                    Thread.sleep(1000);
                }
                yy++;
            }

            System.out.println("Job has Succeeded");
            GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
                    .jobId(jobId)
                    .vaultName(name)
                    .accountId(account)
                    .build();

            ResponseBytes<GetJobOutputResponse> objectBytes =
 glacier.getJobOutputAsBytes(jobOutputRequest);
            // Write the data to a local file.
            byte[] data = objectBytes.asByteArray();
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from a Glacier
 vault");
            os.close();

        } catch (GlacierException | InterruptedException | IOException e) {
            System.out.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「InitiateJob」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定したユーザー所有のボールトからアーカイブを取得するジョブを開始します。ジョブのステータスは、Get-GLCJob コマンドレットを使用して確認できます。ジョブが正常に完了したら、Read-GCJobOutput コマンドレットを使用して、アーカイブの内容をローカルファイルシステムに取得できます。

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

出力:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- API の詳細については、「 コマンドレットリファレンス (V4)」のInitiateJob」を参照してください。 AWS Tools for PowerShell

Tools for PowerShell V5

例 1: 指定したユーザー所有のボールトからアーカイブを取得するジョブを開始します。ジョブのステータスは、Get-GLCJob コマンドレットを使用して確認できます。ジョブが正常に完了したら、Read-GCJobOutput コマンドレットを使用して、アーカイブの内容をローカルファイルシステムに取得できます。

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

出力:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- 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InitiateJob」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

ボールトインベントリを取得します。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_inventory_retrieval(vault):
        """
        Initiates an inventory retrieval job. The inventory describes the
 contents
        of the vault. Standard retrievals typically complete within 3–5 hours.
        When the job completes, you can get the inventory by calling
 get_output().

        :param vault: The vault to inventory.
        :return: The inventory retrieval job.
        """
        try:
            job = vault.initiate_inventory_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on vault %s.", vault.name)
            raise
        else:
            return job
```

ボールトからアーカイブを取得します。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def initiate_archive_retrieval(archive):
        """
        Initiates an archive retrieval job. Standard retrievals typically
 complete
        within 3-5 hours. When the job completes, you can get the archive
 contents
        by calling get_output().

        :param archive: The archive to retrieve.
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on archive %s.", archive.id)
            raise
        else:
            return job
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListJobs**で を使用する

次のサンプルコードは、ListJobs 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。

- ファイルのアーカイブ、通知の取得、ジョブの開始
- アーカイブコンテンツの取得とアーカイブの削除

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// List Amazon S3 Glacier jobs.
    /// </summary>
    /// <param name="vaultName">The name of the vault to list jobs for.</param>
    /// <returns>A list of Amazon S3 Glacier jobs.</returns>
    public async Task<List<GlacierJobDescription>> ListJobsAsync(string
vaultName)
    {
        var request = new ListJobsRequest
        {
            // Using a hyphen "-" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the current account.
            AccountId = "-",
            VaultName = vaultName,
        };
```

```
        var response = await _glacierService.ListJobsAsync(request);

        return response.JobList;
    }
```

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

CLI

AWS CLI

次のコマンドは、my-vault という名前のボールトで進行中のジョブと最近完了したジョブを一覧表示します。

```
aws glacier list-jobs --account-id - --vault-name my-vault
```

出力:

```
{
    "JobList": [
        {
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "RetrievalByteRange": "0-3145727",
            "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
            "Completed": false,
            "SHA256TreeHash":
 "9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
            "JobId": "l7IL5-EkXyEY9Ws95fClzIbk2O5uLYaFdAYOi-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav",
            "ArchiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--
zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
            "JobDescription": "Retrieve archive on 2015-07-17",
            "ArchiveSizeInBytes": 3145728,
            "Action": "ArchiveRetrieval",
            "ArchiveSHA256TreeHash":
 "9628195fcdbcbbe76cdde932d4646fa7de5f219fb39823836d81f0cc0e18aa67",
```

```
            "CreationDate": "2015-07-17T21:16:13.840Z",
            "StatusCode": "InProgress"
        },
        {
            "InventoryRetrievalParameters": {
                "Format": "JSON"
            },
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "Completed": false,
            "JobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
            "Action": "InventoryRetrieval",
            "CreationDate": "2015-07-17T20:23:41.616Z",
            "StatusCode": ""InProgress""
        }
    ]
}
```

Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

- API の詳細については、「AWS CLI コマンドリファレンス」の「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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
```

```
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
        else:
            return job_list
```

- 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 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListTagsForVault**で を使用する

次のサンプルコードは、ListTagsForVault 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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /// <summary>
    /// List tags for an Amazon S3 Glacier vault.
    /// </summary>
    /// <param name="vaultName">The name of the vault to list tags for.</param>
    /// <returns>A dictionary listing the tags attached to each object in the
    /// vault and its tags.</returns>
    public async Task<Dictionary<string, string>> ListTagsForVaultAsync(string
vaultName)
    {
        var request = new ListTagsForVaultRequest
        {
            // Using a hyphen "-" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the default user.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await _glacierService.ListTagsForVaultAsync(request);

        return response.Tags;
    }
```

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

CLI

AWS CLI

次のコマンドは、`my-vault` という名前のボールトに適用されたタグを一覧表示します。

```
aws glacier list-tags-for-vault --account-id - --vault-name my-vault
```

出力:

```
{
    "Tags": {
        "date": "july2015",
        "id": "1234"
    }
}
```

Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **`ListVaults`**で を使用する

次のサンプルコードは、`ListVaults` 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。

- ファイルのアーカイブ、通知の取得、ジョブの開始

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/// <summary>
/// List the Amazon S3 Glacier vaults associated with the current account.
/// </summary>
/// <returns>A list containing information about each vault.</returns>
public async Task<List<DescribeVaultOutput>> ListVaultsAsync()
{
    var glacierVaultPaginator = _glacierService.Paginators.ListVaults(
        new ListVaultsRequest { AccountId = "-" });
    var vaultList = new List<DescribeVaultOutput>();

    await foreach (var vault in glacierVaultPaginator.VaultList)
    {
        vaultList.Add(vault);
    }

    return vaultList;
}
```

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

CLI

AWS CLI

次のコマンドは、デフォルトのアカウントとリージョンのボールトを一覧表示します。

```
aws glacier list-vaults --account-id -
```

出力:

```
{
    "VaultList": [
        {
            "SizeInBytes": 3178496,
            "VaultARN": "arn:aws:glacier:us-west-2:0123456789012:vaults/my-
vault",
            "LastInventoryDate": "2015-04-07T00:26:19.028Z",
            "VaultName": "my-vault",
            "NumberOfArchives": 1,
            "CreationDate": "2015-04-06T21:23:45.708Z"
        }
    ]
}
```

Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

- API の詳細については、「AWS CLI コマンドリファレンス」の「ListVaults」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.model.ListVaultsRequest;
import software.amazon.awssdk.services.glacier.model.ListVaultsResponse;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.DescribeVaultOutput;
import software.amazon.awssdk.services.glacier.model.GlacierException;
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 ListVaults {
    public static void main(String[] args) {
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllVault(glacier);
        glacier.close();
    }

    public static void listAllVault(GlacierClient glacier) {
        boolean listComplete = false;
        String newMarker = null;
        int totalVaults = 0;
        System.out.println("Your Amazon Glacier vaults:");
        try {
            while (!listComplete) {
                ListVaultsResponse response = null;
                if (newMarker != null) {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = glacier.listVaults(request);
                } else {
                    ListVaultsRequest request = ListVaultsRequest.builder()
                            .build();
                    response = glacier.listVaults(request);
                }

                List<DescribeVaultOutput> vaultList = response.vaultList();
                for (DescribeVaultOutput v : vaultList) {
                    totalVaults += 1;
                    System.out.println("* " + v.vaultName());
                }
```

```
                // Check for further results.
                newMarker = response.marker();
                if (newMarker == null) {
                    listComplete = true;
                }
            }

            if (totalVaults == 0) {
                System.out.println("No vaults found.");
            }

        } catch (GlacierException 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「ListVaults」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource
```

```
    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **SetVaultNotifications**で を使用する

次のサンプルコードは、SetVaultNotifications 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。

- ファイルのアーカイブ、通知の取得、ジョブの開始

CLI

AWS CLI

次のコマンドは、my-vault という名前のボールトの SNS 通知を設定します。

```
aws glacier set-vault-notifications --account-id - --vault-name my-vault --vault-
notification-config file://notificationconfig.json
```

notificationconfig.json は、現在のフォルダにある JSON ファイルで、公開する SNS トピックとイベントを指定します。

```
{
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
  "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

- API の詳細については、「AWS CLI コマンドリファレンス」の「SetVaultNotifications」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
```

```
        """
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn,
vault.name
            )
            raise
        else:
            return notification
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **UploadArchive**で を使用する

次のサンプルコードは、UploadArchive 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。

- ファイルのアーカイブ、通知の取得、ジョブの開始

.NET

SDK for .NET

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   /// <summary>
   /// Upload an object to an Amazon S3 Glacier vault.
   /// </summary>
   /// <param name="vaultName">The name of the Amazon S3 Glacier vault to upload
   /// the archive to.</param>
   /// <param name="archiveFilePath">The file path of the archive to upload to
the vault.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<string> UploadArchiveWithArchiveManager(string vaultName,
string archiveFilePath)
   {
       try
       {
           var manager = new ArchiveTransferManager(_glacierService);

           // Upload an archive.
           var response = await manager.UploadAsync(vaultName, "upload archive
test", archiveFilePath);
           return response.ArchiveId;
       }
       catch (AmazonGlacierException ex)
       {
           Console.WriteLine(ex.Message);
           return string.Empty;
       }
   }
```

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

CLI

AWS CLI

次のコマンドは、`archive.zip` という名前の現在のフォルダにあるアーカイブを、`my-vault` という名前のボールトにアップロードします。

```
aws glacier upload-archive --account-id - --vault-name my-vault --
body archive.zip
```

出力:

```
{
    "archiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--
zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
    "checksum":
 "969fb39823836d81f0cc028195fcdbcbbe76cdde932d4646fa7de5f21e18aa67",
    "location": "/0123456789012/vaults/my-vault/archives/
kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw"
}
```

Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

アップロードしたアーカイブを取得するには、aws glacier initiate-job コマンドを使用して取得ジョブを開始します。

- API の詳細については、「AWS CLI コマンドリファレンス」の「UploadArchive」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glacier.GlacierClient;
import software.amazon.awssdk.services.glacier.model.UploadArchiveRequest;
import software.amazon.awssdk.services.glacier.model.UploadArchiveResponse;
import software.amazon.awssdk.services.glacier.model.GlacierException;
import java.io.File;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * 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 UploadArchive {

    static final int ONE_MB = 1024 * 1024;

    public static void main(String[] args) {
        final String usage = """

                Usage:   <strPath> <vaultName>\s

                Where:
```

```
                    strPath - The path to the archive to upload (for example, C:\
\AWS\\test.pdf).
                    vaultName - The name of the vault.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String strPath = args[0];
        String vaultName = args[1];
        File myFile = new File(strPath);
        Path path = Paths.get(strPath);
        GlacierClient glacier = GlacierClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String archiveId = uploadContent(glacier, path, vaultName, myFile);
        System.out.println("The ID of the archived item is " + archiveId);
        glacier.close();
    }

    public static String uploadContent(GlacierClient glacier, Path path, String
 vaultName, File myFile) {
        // Get an SHA-256 tree hash value.
        String checkVal = computeSHA256(myFile);
        try {
            UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
                    .vaultName(vaultName)
                    .checksum(checkVal)
                    .build();

            UploadArchiveResponse res = glacier.uploadArchive(uploadRequest,
 path);
            return res.archiveId();

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

```
    private static String computeSHA256(File inputFile) {
        try {
            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
            return toHex(treeHash);

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s",
inputFile, ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
            System.exit(-1);
        }
        return "";
    }

    public static byte[] computeSHA256TreeHash(File inputFile) throws
IOException,
            NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk, even if it's smaller than 1 MB.
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }
```

```
        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;

            while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                            ioe.getMessage());
                }
            }
        }
    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     */
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[][] prevLvlHashes = chunkSHA256Hashes;
        while (prevLvlHashes.length > 1) {
            int len = prevLvlHashes.length / 2;
            if (prevLvlHashes.length % 2 != 0) {
                len++;
```

```
            }

            byte[][] currLvlHashes = new byte[len][];
            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

                // If there are at least two elements remaining.
                if (prevLvlHashes.length - i > 1) {

                    // Calculate a digest of the concatenated nodes.
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();

                } else { // Take care of the remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }

            prevLvlHashes = currLvlHashes;
        }

        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);
        for (byte datum : data) {
            String hex = Integer.toHexString(datum & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「UploadArchive」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

クライアントの作成

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

アーカイブのアップロード

```
// Load the SDK for JavaScript
import { UploadArchiveCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME

// Create a new service object and buffer
const buffer = new Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer
const params = { vaultName: vaultname, body: buffer };

const run = async () => {
  try {
```

```
    const data = await glacierClient.send(new UploadArchiveCommand(params));
    console.log("Archive ID", data.archiveId);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error uploading archive!", err);
  }
};
run();
```

- 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「UploadArchive」を参照してください。

SDK for JavaScript (v2)

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// Load the SDK for JavaScript
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create a new service object and buffer
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" });
buffer = Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer

var params = { vaultName: "YOUR_VAULT_NAME", body: buffer };
// Call Glacier to upload the archive.
glacier.uploadArchive(params, function (err, data) {
  if (err) {
    console.log("Error uploading archive!", err);
  } else {
    console.log("Archive ID", data.archiveId);
  }
});
```

- 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「UploadArchive」を参照してください。

PowerShell

Tools for PowerShell V4

例 1: 指定したボールトに単一のファイルをアップロードし、アーカイブ ID と計算されたチェックサムを返します。

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

出力:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

例 2: フォルダ階層の内容を、ユーザーアカウント内の指定したボールトにアップロードします。アップロードされたファイルごとに、コマンドレットはファイル名、対応するアーカイブ ID、アーカイブの計算されたチェックサムを出力します。

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

出力:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin           qXAfOdSG...czo729UHXrw d50a1...9184b9
C:\temp\lum.bin             39aNifP3...q9nb8nZkFIg 28886...5c3e27
C:\temp\red.bin             vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt   _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
C:\temp\Folder2\file2.iso   -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- API の詳細については、「 コマンドレットリファレンス (V4)」のUploadArchive」を参照してください。 AWS Tools for PowerShell

Tools for PowerShell V5

例 1: 指定したボールトに単一のファイルをアップロードし、アーカイブ ID と計算されたチェックサムを返します。

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

出力:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

例 2: フォルダ階層の内容を、ユーザーアカウント内の指定したボールトにアップロードします。アップロードされたファイルごとに、コマンドレットはファイル名、対応するアーカイブ ID、アーカイブの計算されたチェックサムを出力します。

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

出力:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
C:\temp\green.bin           qXAfOdSG...czo729UHXrw d50a1...9184b9
C:\temp\lum.bin             39aNifP3...q9nb8nZkFIg 28886...5c3e27
C:\temp\red.bin             vp7E6rU_...Ejk_HhjAxKA e05f7...4e34f5
C:\temp\Folder1\file1.txt   _eRINlip...5Sxy7dD2BaA d0d2a...c8a3ba
C:\temp\Folder2\file2.iso   -Ix3jlmu...iXiDh-XfOPA 7469e...3e86f1
```

- 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 UploadArchive」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。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
```

```
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
            raise
        else:
            return archive
```

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

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **UploadMultipartPart**で を使用する

次のサンプルコードは、UploadMultipartPart 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、アーカイブの最初の 1 MiB (1024 x 1024 バイト) の部分をアップロードします。

```
aws glacier upload-multipart-part --body part1 --range 'bytes
 0-1048575/*' --account-id - --vault-name my-vault --upload-
id 19gaRezEXAMPLES6Ry5YYdqthHOC_kGRCT03L9yetr220UmPtBYKk-
OssZtLqyFu7sY1_lR7vgFuJV6NtcV5zpsJ
```

Amazon Glacier では、オペレーションを実行する際にアカウント ID 引数が必要ですが、ハイフンを使用して使用中のアカウントを指定できます。

body パラメータは、ローカルファイルシステム上のパートファイルへのパスを受け取ります。range パラメータは、完了したアーカイブ内でそのパートが占めるバイト数を示す HTTP コンテンツ範囲を受け取ります。アップロード ID は aws glacier initiate-multipart-upload コマンドによって返され、aws glacier list-multipart-uploads を使用して取得することもできます。

CLI を使用した Amazon Glacier AWS へのマルチパートアップロードの詳細については、AWS 「 CLI ユーザーガイド」のAmazon Glacier 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「UploadMultipartPart」を参照してください。

JavaScript

SDK for JavaScript (v2)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

Buffer オブジェクトの 1 メガバイトのチャンクのマルチパートアップロードを作成します。

```
// Create a new service object and some supporting variables
var glacier = new AWS.Glacier({ apiVersion: "2012-06-01" }),
  vaultName = "YOUR_VAULT_NAME",
  buffer = new Buffer(2.5 * 1024 * 1024), // 2.5MB buffer
  partSize = 1024 * 1024, // 1MB chunks,
  numPartsLeft = Math.ceil(buffer.length / partSize),
  startTime = new Date(),
  params = { vaultName: vaultName, partSize: partSize.toString() };

// Compute the complete SHA-256 tree hash so we can pass it
// to completeMultipartUpload request at the end
var treeHash = glacier.computeChecksums(buffer).treeHash;

// Initiate the multipart upload
console.log("Initiating upload to", vaultName);
// Call Glacier to initiate the upload.
glacier.initiateMultipartUpload(params, function (mpErr, multipart) {
  if (mpErr) {
    console.log("Error!", mpErr.stack);
    return;
  }
  console.log("Got upload ID", multipart.uploadId);

  // Grab each partSize chunk and upload it as a part
```

```
  for (var i = 0; i < buffer.length; i += partSize) {
    var end = Math.min(i + partSize, buffer.length),
      partParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        range: "bytes " + i + "-" + (end - 1) + "/*",
        body: buffer.slice(i, end),
      };

    // Send a single part
    console.log("Uploading part", i, "=", partParams.range);
    glacier.uploadMultipartPart(partParams, function (multiErr, mData) {
      if (multiErr) return;
      console.log("Completed part", this.request.params.range);
      if (--numPartsLeft > 0) return; // complete only when all parts uploaded

      var doneParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        archiveSize: buffer.length.toString(),
        checksum: treeHash, // the computed tree hash
      };

      console.log("Completing upload...");
      glacier.completeMultipartUpload(doneParams, function (err, data) {
        if (err) {
          console.log("An error occurred while uploading the archive");
          console.log(err);
        } else {
          var delta = (new Date() - startTime) / 1000;
          console.log("Completed upload in", delta, "seconds");
          console.log("Archive ID:", data.archiveId);
          console.log("Checksum:  ", data.checksum);
        }
      });
    });
  }
});
```

- 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「UploadMultipartPart」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

SDK を使用した S3 Glacier のシナリオ AWS SDKs

次のコード例は、 AWS SDKs を使用して S3 Glacier で一般的なシナリオを実装する方法を示しています。これらのシナリオでは、S3 Glacier 内で複数の関数を呼び出したり、他の AWS のサービスと組み合わせたりすることで、特定のタスクを実行する方法を紹介します。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 を使用してファイルを Amazon S3 Glacier にアーカイブし、通知を取得し、ジョブを開始する
- AWS SDK を使用して Amazon S3 Glacier アーカイブコンテンツを取得し、アーカイブを削除する

AWS SDK を使用してファイルを Amazon S3 Glacier にアーカイブし、通知を取得し、ジョブを開始する

次のコードサンプルは、以下の操作方法を示しています。

- Amazon S3 Glacier ボールトを作成します。
- ボールトを設定して、Amazon SNS トピックに通知を発行します。
- ボールトにアーカイブファイルをアップロードします。
- アーカイブの取得ジョブを開始します。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

S3 Glacier オペレーションをラップするクラスを作成します。

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource


    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
```

```
            raise
        else:
            return vault


    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise


    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
                archiveDescription=archive_description, body=archive_file
            )
            logger.info(
                "Uploaded %s with ID %s to vault %s.",
                archive_description,
                archive.id,
                vault.name,
            )
        except ClientError:
            logger.exception(
                "Couldn't upload %s to %s.", archive_description, vault.name
            )
            raise
        else:
            return archive
```

```
    @staticmethod
    def initiate_archive_retrieval(archive):
        """
        Initiates an archive retrieval job. Standard retrievals typically
complete
        within 3-5 hours. When the job completes, you can get the archive
contents
        by calling get_output().

        :param archive: The archive to retrieve.
        :return: The archive retrieval job.
        """
        try:
            job = archive.initiate_archive_retrieval()
            logger.info("Started %s job with ID %s.", job.action, job.id)
        except ClientError:
            logger.exception("Couldn't start job on archive %s.", archive.id)
            raise
        else:
            return job


    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
```

```
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
        else:
            return job_list


    def set_notifications(self, vault, sns_topic_arn):
        """
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
        for notifications. Amazon S3 Glacier publishes messages to this topic for
        the configured list of events.

        :param vault: The vault to set up to publish notifications.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                              receives notifications.
        :return: Data about the new notification configuration.
        """
        try:
            notification = self.glacier_resource.Notification("-", vault.name)
            notification.set(
                vaultNotificationConfig={
                    "SNSTopic": sns_topic_arn,
                    "Events": [
                        "ArchiveRetrievalCompleted",
                        "InventoryRetrievalCompleted",
                    ],
                }
            )
            logger.info(
                "Notifications will be sent to %s for events %s from %s.",
                notification.sns_topic,
                notification.events,
                notification.vault_name,
            )
        except ClientError:
```

```
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn,
 vault.name
            )
            raise
        else:
            return notification
```

ラッパークラスの関数を呼び出して、ボールトを作成してファイルをアップロードし、通知を公開し、アーカイブを取得するジョブを開始するようにボールトを設定します。

```
def upload_demo(glacier, vault_name, topic_arn):
    """
    Shows how to:
    * Create a vault.
    * Configure the vault to publish notifications to an Amazon SNS topic.
    * Upload an archive.
    * Start a job to retrieve the archive.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to create.
    :param topic_arn: The ARN of an Amazon SNS topic that receives notification
 of
                      Amazon S3 Glacier events.
    """
    print(f"\nCreating vault {vault_name}.")
    vault = glacier.create_vault(vault_name)
    print("\nList of vaults in your account:")
    glacier.list_vaults()
    print(f"\nUploading glacier_basics.py to {vault.name}.")
    with open("glacier_basics.py", "rb") as upload_file:
        archive = glacier.upload_archive(vault, "glacier_basics.py", upload_file)
    print(
        "\nStarting an archive retrieval request to get the file back from the "
        "vault."
    )
    glacier.initiate_archive_retrieval(archive)
    print("\nListing in progress jobs:")
    glacier.list_jobs(vault, "in_progress")
    print(
        "\nBecause Amazon S3 Glacier is intended for infrequent retrieval, an "
```

```
        "archive request with Standard retrieval typically completes within 3-5 "
        "hours."
    )
    if topic_arn:
        notification = glacier.set_notifications(vault, topic_arn)
        print(
            f"\nVault {vault.name} is configured to notify the "
            f"{notification.sns_topic} topic when {notification.events} "
            f"events occur. You can subscribe to this topic to receive "
            f"a message when the archive retrieval completes.\n"
        )
    else:
        print(
            f"\nVault {vault.name} is not configured to notify an Amazon SNS
topic "
            f"when the archive retrieval completes so wait a few hours."
        )
    print("\nRetrieve your job output by running this script with the --retrieve
flag.")
```

- 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。
  - CreateVault
  - InitiateJob
  - ListJobs
  - ListVaults
  - SetVaultNotifications
  - UploadArchive

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK を使用して Amazon S3 Glacier アーカイブコンテンツを取得し、アーカイブを削除する

次のコードサンプルは、以下の操作方法を示しています。

- Amazon S3 Glacier ボールトのジョブをリストし、ジョブのステータスを取得します。
- 完了したアーカイブの取得ジョブの出力を取得します。
- アーカイブを削除します。
- ボールトを削除します。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

S3 Glacier オペレーションをラップするクラスを作成します。

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""

    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource
```

```
    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == "all":
                jobs = vault.jobs.all()
            elif job_type == "in_progress":
                jobs = vault.jobs_in_progress.all()
            elif job_type == "completed":
                jobs = vault.completed_jobs.all()
            elif job_type == "succeeded":
                jobs = vault.succeeded_jobs.all()
            elif job_type == "failed":
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type,
vault.name)
            raise
        else:
            return job_list


    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
```

```
        """
        try:
            response = job.get_output()
            out_bytes = response["body"].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if "archiveDescription" in response:
                logger.info(
                    "These bytes are described as '%s'",
response["archiveDescription"]
                )
        except ClientError:
            logger.exception("Couldn't get output for job %s.", job.id)
            raise
        else:
            return out_bytes


    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id,
archive.vault_name
            )
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise


    @staticmethod
    def delete_vault(vault):
        """
        Deletes a vault.

        :param vault: The vault to delete.
        """
        try:
            vault.delete()
```

```
            logger.info("Deleted vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't delete vault %s.", vault.name)
            raise
```

Wrapper クラスの関数を呼び出して、完了したジョブからアーカイブコンテンツを取得し、アーカイブを削除します。

```
def retrieve_demo(glacier, vault_name):
    """
    Shows how to:
    * List jobs for a vault and get job status.
    * Get the output of a completed archive retrieval job.
    * Delete an archive.
    * Delete a vault.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to query for jobs.
    """
    vault = glacier.glacier_resource.Vault("-", vault_name)
    try:
        vault.load()
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            print(
                f"\nVault {vault_name} doesn't exist. You must first run this
 script "
                f"with the --upload flag to create the vault."
            )
            return
        else:
            raise

    print(f"\nGetting completed jobs for {vault.name}.")
    jobs = glacier.list_jobs(vault, "completed")
    if not jobs:
        print("\nNo completed jobs found. Give it some time and try again
 later.")
        return

    retrieval_job = None
```

```
    for job in jobs:
        if job.action == "ArchiveRetrieval" and job.status_code == "Succeeded":
            retrieval_job = job
            break
    if retrieval_job is None:
        print(
            "\nNo ArchiveRetrieval jobs found. Give it some time and try again "
            "later."
        )
        return

    print(f"\nGetting output from job {retrieval_job.id}.")
    archive_bytes = glacier.get_job_output(retrieval_job)
    archive_str = archive_bytes.decode("utf-8")
    print("\nGot archive data. Printing the first 10 lines.")
    print(os.linesep.join(archive_str.split(os.linesep)[:10]))

    print(f"\nDeleting the archive from {vault.name}.")
    archive = glacier.glacier_resource.Archive(
        "-", vault.name, retrieval_job.archive_id
    )
    glacier.delete_archive(archive)

    print(f"\nDeleting {vault.name}.")
    glacier.delete_vault(vault)
```

- 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。
  - DeleteArchive
  - DeleteVault
  - GetJobOutput
  - ListJobs

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK での S3 Glacier の使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

Amazon S3 Glacier でのセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティの影響を受けやすい組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、 AWS とユーザーの間で共有される責任です。責任共有モデルでは、この責任がクラウドのセキュリティおよびクラウド内のセキュリティとして説明されています。

- クラウドのセキュリティ – は、 で AWS サービスを実行するインフラストラクチャを保護する責任 AWS を担います AWS クラウド。 AWS また、 は、お客様が安全に使用できるサービスも提供します。セキュリティの有効性は、AWS コンプライアンスプログラムの一環として、サードパーティーの審査機関によって定期的にテストおよび検証されています。Amazon S3 Glacier (S3 Glacier) に適用するコンプライアンスプログラムの詳細については、AWS 「コンプライアンスプログラムによる 対象範囲内のサービス」を参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、お客様のデータの機密性、組織の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

このドキュメントは、S3 Glacier を使用するときに、共有責任モデルを適用する方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するために S3 Glacier を設定する方法を示します。また、S3 Glacier リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- Amazon S3 Glacier におけるデータ保護
- Amazon S3 Glacier の ID とアクセス管理
- Amazon S3 Glacier でのログ記録とモニタリング
- Amazon S3 Glacier のコンプライアンス検証
- Amazon S3 Glacier の耐障害性
- Amazon S3 Glacier のインフラストラクチャセキュリティ

Amazon S3 Glacier におけるデータ保護

Amazon S3 Glacier (S3 Glacier) は、データのアーカイブと長期バックアップ用の堅牢なクラウドストレージです。S3 Glacier は、99.999999999 パーセントの耐久性を実現するよう設計されており、厳格な規制要件を満たすよう包括的なセキュリティとコンプライアンス機能を備えています。S3 Glacier は、複数の AWS アベイラビリティーゾーン (AZ) と各 AZ 内の複数のデバイスにデータを冗長的に保存します。耐久性を高めるために、S3 Glacier は、アップロードが正常に実行されたことを確認する前に、複数の AZ 全体にデータを同期的に保存します。

AWS グローバルクラウドインフラストラクチャの詳細については、「グローバルインフラストラクチャ」を参照してください。

データ保護の目的で、 AWS アカウント 認証情報を保護し、個々のユーザー、グループ、またはロールに職務を果たすために必要なアクセス許可のみを付与することをお勧めします。

コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

トピック

- データ暗号化
- キーの管理
- インターネットトラフィックのプライバシー

データ暗号化

データ保護とは、転送中 (Amazon S3 Glacier との間で送受信されるデータ) および保管中 (AWS データセンターに格納されるデータ) のデータを保護することです。Secure Sockets Layer (SSL) またはクライアント側の暗号化を使用して、S3 Glacier に直接アップロードされる転送中のデータを保護することができます。

Amazon S3 から S3 Glacier にアクセスすることもできます。Amazon S3 はAmazon S3 バケットのライフサイクル設定をサポートしています。これにより、オブジェクトをアーカイブのために S3 Glacier ストレージクラスに移行させることができます。ライフサイクルポリシーを使用した Amazon S3 および S3 Glacier 間の送信データは、SSL を使用して暗号化されます。

S3 Glacier へ保存される静止データは、自動的に AWSの管理するキーを利用し 256 ビットの Advanced Encryption Standard (AES-256) を使用してサーバー側で暗号化されます。独自のキーを

管理する場合は、S3 Glacier にデータを格納する前にクライアント側の暗号化を使用することもできます。Amazon S3 のデフォルトの暗号化機能の詳細については、Amazon Simple Storage Service ユーザーガイドの「S3 バケットの Amazon S3 デフォルト暗号化」を参照してください。

キーの管理

サーバー側の暗号化は、保管中のデータ暗号化に関するものです。つまり、Amazon S3 Glacier は、データセンターの書き込み時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。リクエストが認証され、お客様がアクセス許可を持っている限りは、オブジェクトが暗号化されているかどうかに関係なく同じ方法でアクセスできます。

S3 Glacier に格納される保管データは、 AWSが管理するキーを使用して、サーバー側で自動的に AES-256 によって暗号化されます。追加の保護として、 は定期的にローテーションするルートキーを使用してキー自体を AWS 暗号化します。

インターネットトラフィックのプライバシー

ネットワーク経由で Amazon S3 Glacier にアクセスするには、 AWS 公開APIs を使用します。クライアントは Transport Layer Security (TLS) 1.2 をサポートしている必要があります。TLS 1.3 以降が推奨されます。クライアントは、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) などの Perfect Forward Secrecy (PFS) を備えた暗号スイートもサポートする必要があります。モードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。また、リクエストには、IAM プリンシパルに関連付けられたアクセスキー ID およびシークレットアクセスキーによる署名が必要です。または、リクエストへの署名のために一時的にセキュリティ認証情報を生成する AWS Security Token Service (AWS STS) を使用することもできます。

VPC エンドポイント

Virtual Private Cloud (VPC) エンドポイントを使用すると、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続を必要とせずに、サポートされている AWS サービスや AWS PrivateLink を搭載した VPC エンドポイントサービスに VPC をプライベートに接続できます。S3 Glacier は VPC エンドポイントに直接対応していませんが、Amazon S3 と統合したストレージの S3 Glacier にアクセスする場合、Amazon Simple Storage Service (Amazon S3) VPC エンドポイントを利用することができます。

Amazon S3 ライフサイクル設定と GLACIER ストレージクラスへのオブジェクト移行の詳細については、Amazon Simple Storage Service ユーザーガイドの「オブジェクトのライフサイクル管理」と「オブジェクトの移行」を参照してください。VPC エンドポイント の詳細については、Amazon VPC ユーザーガイドの「VPC Endpoints」を参照してください。

Amazon S3 Glacier の ID とアクセス管理

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、S3 Glacier リソースの使用を認証 (サインイン)、認可 (アクセス許可を持つ) できるユーザーを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- Amazon S3 Glacier と IAM が連携する方法
- Amazon S3 Glacier の ID ベースポリシーの例
- Amazon S3 Glacier のリソースベースポリシーの例
- Amazon S3 Glacier ID とアクセスのトラブルシューティング
- API の権限リファレンス

対象者

AWS Identity and Access Management (IAM) の使用方法は、S3 Glacier で行う作業によって異なります。

サービスユーザー – ジョブを実行するために S3 Glacier サービスを使用する場合は、管理者から必要な認証情報とアクセス許可が付与されます。さらに多くの S3 Glacier 機能を使用して作業を実行するとき、追加のアクセス許可が必要になる場合があります。アクセスの管理方法を理解すると、管理者に適切なアクセス許可をリクエストするのに役に立ちます。S3 Glacier で機能にアクセスできない場合は、「Amazon S3 Glacier ID とアクセスのトラブルシューティング」を参照してください。

サービス管理者 - 社内の S3 Glacier リソースを担当している場合は、おそらく、S3 Glacier に完全にアクセスすることができます。サービスを利用するユーザーがどの S3 Glacier 機能やリソースにアクセスできるかを決めるのは、管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。ご自分の会社で S3 Glacier で IAM を使用する方法の詳細については、「Amazon S3 Glacier と IAM が連携する方法」を参照してください。

IAM 管理者 – IAM 管理者は、S3 Glacier へのアクセスを管理するポリシーを作成する方法の詳細を確認する場合があります。IAM で使用できる S3 Glacier ID ベースのポリシーの例を表示するには、「Amazon S3 Glacier の ID ベースポリシーの例」を参照してください。

アイデンティティを使用した認証

認証は、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして、または IAM ロールを引き受けることによって認証 (にサインイン AWS) される必要があります。AWS アカウントのルートユーザー

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center （ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook 認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 AWS サインイン ユーザーガイド」の「 へのサインイン方法 AWS アカウント」を参照してください。

AWS プログラムで にアクセスする場合、 は、ソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストを暗号化して署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに自分で署名する推奨方法の使用については、「IAM ユーザーガイド」の「API リクエストに対するAWS Signature Version 4」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。たとえば、 では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させる AWS ことをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「IAM のAWS 多要素認証」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインイン ID から始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するとき

に使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、一時的な認証情報を使用して にアクセスするために ID プロバイダーとのフェデレーション AWS のサービス を使用することを要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリ、または ID ソースを介して提供された認証情報 AWS のサービス を使用して にアクセスするすべてのユーザーです。フェデレーティッド ID がアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成することも、独自の ID ソースのユーザーとグループのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用できるようにすることもできます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM Identity Center?」(IAM Identity Center とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、1 人のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内の ID です。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユー

ザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。で IAM ロールを一時的に引き受けるには AWS Management Console、ユーザーから IAM ロール (コンソール) に切り替えることができます。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールについては、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「Permission sets」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。

- 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用してアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストをリクエストする を使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除することができます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」を参照してください。
- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、 サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。EC2 インスタンスに AWS ロールを割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュ

メント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、IAM ユーザーガイドの JSON ポリシー概要を参照してください。

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プ

リンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、追加のあまり一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPsは、 の組織または組織単位 (OU) の最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、ビジネスが所有する複数の をグループ化して一元管理するためのサービス AWS アカウント です。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCP)」を参照してください。
- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定する

ために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースに対するアクセス許可を制限し、組織に属するかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID に対する有効なアクセス許可に影響を与える可能性があります。RCP AWS のサービス をサポートする のリストを含む Organizations と RCPs「リソースコントロールポリシー (RCPs」を参照してください。 AWS Organizations

- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関係する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM ユーザーガイドの「ポリシー評価ロジック」を参照してください。

Amazon S3 Glacier と IAM が連携する方法

IAM を使用して S3 Glacier へのアクセスを管理するときは、事前に、S3 Glacier で使用できる IAM の機能について理解しておきましょう。

Amazon S3 Glacier で使用できる IAM の機能

IAM 機能	S3 Glacier のサポート
アイデンティティベースポリシー	はい
リソースベースのポリシー	はい
ポリシーアクション	はい
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい

IAM 機能	S3 Glacier のサポート
ACL	いいえ
ABAC (ポリシー内のタグ)	いいえ
一時的な認証情報	はい
プリンシパル権限	いいえ
サービスロール	いいえ
サービスリンクロール	いいえ

S3 Glacier およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要については、IAM ユーザーガイドのAWS 「IAM と連携する のサービス」を参照してください。

S3 Glacier の ID ベースのポリシー

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。ID ベースのポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素について学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

S3 Glacier の ID ベースポリシーの例

S3 Glacier の ID ベースのポリシーの例を確認するには、「Amazon S3 Glacier の ID ベースポリシーの例」を参照してください。

S3 Glacier 内のリソースベースのポリシー

リソースベースのポリシーのサポート: あり

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、リソースにアクセスするためのアクセス許可をプリンシパルエンティティ (ユーザーまたはロール) に付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与する必要はありません。詳細については、「IAM ユーザーガイド」の「IAM でのクロスアカウントリソースアクセス」を参照してください。

S3 Glacier サービスは、ボールトポリシーと呼ばれるリソースベースのポリシーの 1 つのタイプのみをサポートし、それがボールトにアタッチされます。このポリシーは、ボールトでアクションを実行できるプリンシパルを定義します。

S3 Glacier ボールトポリシーは、以下の方法でアクセス許可を管理します。

- 複数の個々のユーザーポリシーではなく、単一のボールトポリシーを使用してアカウントでユーザーアクセス許可を管理します。
- IAM ロールを使用する代わりの方法として、クロスアカウント権限を管理します。

S3 Glacier 内のリソースベースのポリシーの例

S3 Glacier のリソースベースのポリシーの例を確認するには、「Amazon S3 Glacier のリソースベースポリシーの例」を参照してください。

S3 Glacier のポリシーアクション

ポリシーアクションのサポート:あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは依存アクションと呼ばれます。

このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

S3 Glacier アクションの一覧については、「サービス認可リファレンス」の「Amazon S3 Glacier で定義されるアクション」を参照してください。

S3 Glacier のポリシーアクションは、アクションの前に、次のプレフィックスを使用しています。

```
glacier
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
            "glacier:CreateVault",
            "glacier:DescribeVault",
            "glacier:ListVaults"
         ]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、`Describe` という単語で始まるすべてのアクションを指定するには次のアクションを含めます。

```
"Action": "glacier:GetVault*"
```

S3 Glacier の ID ベースのポリシーの例を確認するには、「Amazon S3 Glacier の ID ベースポリシーの例」を参照してください。

S3 Glacier のポリシーリソース

ポリシーリソースのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルが、どのリソースに対してどのような条件下でアクションを実行できるかということです。

`Resource` JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ステートメントには`Resource` または `NotResource` 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

S3 Glacier リソースのタイプとその ARN のリストを確認するには、「サービス認可リファレンス」の「Amazon S3 Glacier で定義されるリソース」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon S3 Glacier で定義されるアクション」を参照してください。

S3 Glacier では、プライマリリソースはボールトです。S3 Glacier では、ボールトレベルでのみポリシーをサポートしています。つまり、IAM ポリシーでは、指定する`Resource`値は、特定のボールトまたは特定の AWS リージョンのボールトのセットにすることができます。S3 Glacier では、アーカイブレベルのアクセス許可はサポートされていません。

すべての S3 Glacier アクションで、`Resource` はアクセス権限を付与するボールトを指定します。これらのリソースには、以下の表に示されているように、リソースに関連付けられる一意の Amazon リソースネーム (ARN) があり、ARN でワイルドカード文字 (*) を使用して同じプレフィックスで始まるボールト名に一致させることができます。

S3 Glacier には、S3 Glacier リソースを操作するための一連のオペレーションが用意されています。利用可能なオペレーションの詳細については、「Amazon S3 Glacier の API リファレンス」を参照してください。

複数のリソースをサポートする S3 Glacier API アクションもあります。例えば、`glacier:AddTagsToVault` は examplevault1 と examplevault2 にアクセスするため、プリンシパルには両方のリソースにアクセスする許可が必要です。複数リソースを単一ステートメントで指定するには、ARN をカンマで区切ります。

```
"Resource": [
""arn:aws:glacier:us-west-2:123456789012:vaults/examplevault1",",
""arn:aws:glacier:us-west-2:123456789012:vaults/examplevault2","
]
```

S3 Glacier のポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルが、どのリソースに対してどのような条件下でアクションを実行できるかということです。

`Condition` 要素 (または `Condition` ブロック) を使用すると、ステートメントが有効な条件を指定できます。`Condition` 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の `Condition` 要素を指定する場合、または 1 つの `Condition` 要素に複数のキーを指定する場合、 AWS では `AND` 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理`OR`オペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、IAM ユーザーガイドのAWS 「グローバル条件コンテキストキー」を参照してください。

S3 Glacier の条件キーのリストを確認するには、「サービス認可リファレンス」の「Amazon S3 Glacier の条件キー」を参照してください。どのアクションおよびリソースで条件キーを使用できるかについては、「Amazon S3 Glacier で定義されるアクション」を参照してください。

Glacier 固有の条件キーの使用例については、「ボールトロックポリシー」を参照してください。

S3 Glacier の ACL

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

S3 Glacier での ABAC

ABAC (ポリシー内のタグ) のサポート: なし

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。では AWS、これらの属性はタグと呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合にオペレーションを許可するように ABAC ポリシーをします。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

S3 Glacier で認証情報を一時的に使用する

一時的な認証情報のサポート: あり

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報 AWS のサービス を使用する方法などの詳細については、IAM ユーザーガイドAWS のサービス の「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。たとえば、会社のシングルサインオン (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ユーザーから IAM ロールに切り替える (コンソール)」を参照してください。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用してアクセスすることができます AWS。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

サービス間での S3 Glacier のプリンシパルのアクセス許可

転送アクセスセッション (FAS) のサポート: なし

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

S3 Glacier のサービスロール

サービスロールのサポート: なし

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」を参照してください。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、S3 Glacier の機能にエラーが生じる可能性があります。S3 Glacier が指示したとき以外は、サービスロールを編集しないでください。

S3 Glacier のサービスリンクロール

サービスにリンクされたロールのサポート: なし

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、 サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

サービスにリンクされたロールの作成または管理の詳細については、「IAM と提携するAWS のサービス」を参照してください。表の「サービスリンクロール」列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon S3 Glacier の ID ベースポリシーの例

デフォルトでは、ユーザーとロールには S3 Glacier リソースを作成または変更する許可がありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

これらのサンプル JSON ポリシードキュメントを使用して IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーを作成する (コンソール)」を参照してください。

S3 Glacier が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、「サービス認可リファレンス」の「Amazon S3 Glacier 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。

以下は、`us-west-2` AWS リージョン内のすべてのボールトを識別する Amazon リソースネーム (ARN`glacier:ListVaults`) を使用して`glacier:CreateVault`、リソースに対する 3 つの S3 Glacier ボールト関連アクション (、、) `glacier:DescribeVault` のアクセス許可を付与するポリシーの例です。ARNs AWS リソースを一意に識別します。S3 Glacier で使用する ARN の詳細については、「」を参照してください。S3 Glacier のポリシーリソース。

```
{
   "Version": "2012-10-17",
```

```
        "Statement": [
           {
              "Effect": "Allow",
              "Action": [
              "glacier:CreateVault",
              "glacier:DescribeVault",
              "glacier:ListVaults"
              ],
              "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/*"
           }
        ]
     }
```

ポリシーは、us-west-2 リージョンでのボールトの作成、一覧表示、説明の取得を行う権限を付与します。ARN の末尾のワイルドカード文字 (*) は、このステートメントはどのボールト名とも一致する可能性があることを意味します。

> **Important**
>
> glacier:CreateVault オペレーションを使用してボールトを作成する権限を付与するときは、ボールトを作成するまでボールト名がわからないため、ワイルドカード文字 (*) を指定する必要があります。

トピック

- ポリシーに関するベストプラクティス
- S3 Glacier のコンソールを使用する
- 自分の権限の表示をユーザーに許可する
- お客様が管理するポリシーの例

ポリシーに関するベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰が S3 Glacier リソースを作成し、これにアクセスし、これを削除できるかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行 – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理

ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「ジョブ機能の AWS マネージドポリシー」を参照してください。

- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、 などの特定の を通じてサービスアクションが使用される場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素:条件」を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer でポリシーを検証する」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、セキュリティを強化するために MFA を有効にします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA を使用した安全な API アクセス」を参照してください。

IAM でのベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

S3 Glacier のコンソールを使用する

Amazon S3 Glacier コンソールにアクセスするには、アクセス許可の最小限のセットが必要です。これらのアクセス許可により、 AWS アカウントにある S3 Glacier リソースの詳細を一覧で表示できます。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、その

ポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

S3 Glacier コンソールは、S3 Glacier ボールトの作成および管理のための統合された環境を提供します。次の例に示すように、作成した最小限の IAM ID には、S3 Glacier コンソールを表示するための `glacier:ListVaults` アクションのアクセス許可を付与する必要があります。

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

AWS は、 によって作成および管理されるスタンドアロン IAM ポリシーを提供することで、多くの一般的なユースケースに対処します AWS。マネージドポリシーは、一般的ユースケースに必要な許可を付与することで、どの許可が必要なのかをユーザーが調査する必要をなくすることができます。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

アカウントのユーザーにアタッチできる以下の AWS マネージドポリシーは、S3 Glacier に固有のものです。

- AmazonGlacierReadOnlyAccess – を介して S3 Glacier への読み取り専用アクセスを許可します AWS Management Console。
- AmazonGlacierFullAccess – を通じて S3 Glacier へのフルアクセスを許可します AWS Management Console。

独自のカスタム IAM ポリシーを作成して、S3 Glacier API アクションとリソースのためのアクセス権限を許可することもできます。これらのカスタムポリシーは、S3 Glacier ボールト用に作成したカスタム IAM ロールにアタッチできます。

次のセクションで説明する S3 Glacier AWS 管理ポリシーの両方が、 のアクセス許可を付与します`glacier:ListVaults`。

詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
```

```
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

お客様が管理するポリシーの例

このセクションでは、さまざまな S3 Glacier アクションのアクセス権限を付与するユーザー ポリシー例を示しています。これらのポリシーは、S3 Glacier REST API、Amazon SDKs、、 AWS CLI または該当する場合は S3 Glacier マネジメントコンソールを使用している場合に機能します。

> **Note**
> 例はすべて、米国西部 (オレゴン) リージョン (us-west-2) を使用し、架空のアカウント ID を使用しています。

例

- 例 1: ボールトからのアーカイブのダウンロードをユーザーに許可する
- 例 2: ボールトの作成と通知設定をユーザーに許可する
- 例 3: ユーザーに特定のボールトへのアーカイブのアップロードを許可する
- 例 4: ユーザーに特定のボールトのフルアクセス権限を許可する

例 1: ボールトからのアーカイブのダウンロードをユーザーに許可する

アーカイブをダウンロードするには、まずアーカイブを取得するジョブを開始します。取得ジョブが完了したら、データをダウンロードできます。次のポリシー例では、ジョブを開始する `glacier:InitiateJob` アクションと、取得したデータをダウンロードするアクションのアクセス権限 (ユーザーにボールトからのアーカイブまたはボールトインベントリの取得を許可する) を付与して、`glacier:GetJobOutput` アクションに対するアクセス権限を付与して、取得したデータをダウンロードします。このポリシーは、ユーザーがジョブのステータスを確認できるように、`glacier:DescribeJob` アクションを実行できる権限も付与します。詳細については、「ジョブの開始 (ジョブの POST)」を参照してください。

ポリシーは、`examplevault` という名前のボールトにこれらのアクセス権限を付与します。Amazon S3 Glacier コンソールから、またはプログラムで ボールトの説明 (GET vault)、または

ボールトのリスト (GET vaults) API アクションを呼び出すことで、ボールトの ARN を取得できます。

```
{
                "Version":"2012-10-17",
                "Statement": [
                   {
                      "Effect": "Allow",
                      "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/
examplevault",
                      "Action":["glacier:InitiateJob",
                               "glacier:GetJobOutput",
                               "glacier:DescribeJob"]
                   }
                ]
                }
```

例 2: ボールトの作成と通知設定をユーザーに許可する

次のポリシー例では、Resource 要素で指定されているように、us-west-2 リージョンにボールトを作成するアクセス許可を付与し、通知設定を行います。通知の操作方法の詳細については、「Amazon S3 Glacier でのボールト通知の設定」を参照してください。このポリシーは、 AWS リージョンのボールトを一覧表示し、特定のボールトの説明を取得するアクセス許可も付与します。

> **Important**
>
> glacier:CreateVault オペレーションを使用してボールトを作成する権限を付与するときは、ボールトを作成するまでボールト名がわからないため、Resource 値でワイルドカード文字 (*) を指定する必要があります。

```
{
                "Version":"2012-10-17",
                "Statement": [
                   {
                      "Effect": "Allow",
                      "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/*",
                      "Action":["glacier:CreateVault",
                               "glacier:SetVaultNotifications",
                               "glacier:GetVaultNotifications",
                               "glacier:DeleteVaultNotifications",
```

```
                                "glacier:DescribeVault",
                                "glacier:ListVaults"]
                   }
                ]
                }
```

例 3: ユーザーに特定のボールトへのアーカイブのアップロードを許可する

次のポリシー例では、us-west-2 リージョンの特定のボールトにアーカイブをアップロードする権限を付与します。これらの権限は、ユーザーが アーカイブのアップロード (POST archive) API オペレーションを使用し、または部分的に マルチパートアップロードの開始 (POST multipart-uploads) API オペレーションを使用して、すべてのアーカイブをアップロードすることを許可します。

```
{
                "Version":"2012-10-17",
                "Statement": [
                   {
                      "Effect": "Allow",
                      "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/
examplevault",
                      "Action":["glacier:UploadArchive",
                                "glacier:InitiateMultipartUpload",
                                "glacier:UploadMultipartPart",
                                "glacier:ListParts",
                                "glacier:ListMultipartUploads",
                                "glacier:CompleteMultipartUpload"]
                   }
                ]
                }
```

例 4: ユーザーに特定のボールトのフルアクセス権限を許可する

次のポリシー例では、 という名前のボールトで、すべての S3 Glacier アクションのアクセス許可を付与しますexamplevault。

```
{
            "Version":"2012-10-17",
            "Statement": [
               {
                  "Effect": "Allow",
                  "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/
examplevault",
```

```
                "Action":["glacier:*"]
            }
        ]
        }
```

Amazon S3 Glacier のリソースベースポリシーの例

S3 Glacier ボールトは、1つのボールト アクセスポリシーとそれに関連付けられている ボールトロック ポリシー を持つことができます。Amazon S3 Glacier ボールトアクセスポリシーは、ボールトに対するアクセス許可を管理するのに使用できるリソースベースのポリシーです。ボールトロックポリシーは、ロック可能なボールトアクセスポリシーです。ボールトロックポリシーをロックすると、ポリシーは変更できなくなります。コンプライアンス制御を適用しているボールトロックのポリシーを使用できます。

トピック

- ボールトアクセスポリシー
- ボールトロックポリシー

ボールトアクセスポリシー

Amazon S3 Glacier ボールトアクセスポリシーは、ボールトに対する権限を管理するのに使用できるリソースベースのポリシーです。

各ボールトに対して 1 つのボールトアクセスポリシーを作成してアクセス権限を管理できます。ボールトアクセスポリシーのアクセス許可は、いつでも変更できます。S3 Glacier では、各ボールトでのボールトロックポリシーもサポートしています。ボールトロックポリシーは、ロック後に変更できません。ボールトロックポリシーの操作の詳細については、「ボールトロックポリシー」を参照してください。

例

- 例 1: 特定の Amazon S3 Glacier アクションのクロスアカウント権限の付与
- 例 2: MFA 削除オペレーションのクロスアカウント権限の付与

例 1: 特定の Amazon S3 Glacier アクションのクロスアカウント権限の付与

次のポリシー例では、`examplevault` というボールトの一連の S3 Glacier オペレーションに対する 2つの AWS アカウント に、クロスアカウント権限を付与します。

> **Note**
> ボールトを所有するアカウントには、ボールトに関連するすべての料金が課金されます。許可された外部アカウントによって行われたすべてのリクエスト、データ転送、および取得のコストは、ボールトを所有するアカウントに課金されます。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"cross-account-upload",
         "Principal": {
            "AWS": [
               "arn:aws:iam::123456789012:root",
               "arn:aws:iam::444455556666:root"
            ]
         },
         "Effect":"Allow",
         "Action": [
            "glacier:UploadArchive",
            "glacier:InitiateMultipartUpload",
            "glacier:AbortMultipartUpload",
            "glacier:CompleteMultipartUpload"
         ],
         "Resource": [
            "arn:aws:glacier:us-west-2:999999999999:vaults/examplevault"

         ]
      }
   ]
}
```

例 2: MFA 削除オペレーションのクロスアカウント権限の付与

Multi-Factor Authentication (MFA) を使用して、S3 Glacier リソースを保護できます。セキュリティのレベルをさらに強化するために、MFA は、ユーザーに有効な MFA コードを入力させて MFA デバイスの物理的所有を証明することを要求します。MFA アクセスの設定の詳細については、「」を参照してください。MFA 保護 API アクセスの設定のIAM ユーザーガイド。

このポリシー例では、リクエストが MFA デバイスで認証されている場合、examplevault という名前のボールトからアーカイブを削除する権限を AWS アカウント 一時的な認証情報で に付与します。ポリシーは `aws:MultiFactorAuthPresent` 条件キーを使用して、この追加要件を指定します。詳細については、IAM ユーザーガイドの一部のサービスに使用可能なキーを参照してください。

```
                    {
                       "Version": "2012-10-17",
                       "Statement": [
                          {
                             "Sid": "add-mfa-delete-requirement",
                             "Principal": {
                                "AWS": [
                                   "arn:aws:iam::123456789012:root"
                                ]
                             },
                             "Effect": "Allow",
                             "Action": [
                                "glacier:Delete*"
                             ],
                             "Resource": [
                                "arn:aws:glacier:us-west-2:999999999999:vaults/
examplevault"
                             ],
                             "Condition": {
                                "Bool": {
                                   "aws:MultiFactorAuthPresent": true
                                }
                             }
                          }
                       ]
                    }
```

ボールトロックポリシー

Amazon S3 Glacier (S3 Glacier) ボールトでは、リソースベースのボールトアクセスポリシーを 1 つ持つことが可能で、それに 1 つのボールトロックポリシーを割り当てることができます。ボールトロックポリシーは、ユーザーがロック可能なボールトアクセスポリシーです。ボールトロックを使用すると、規制要件およびコンプライアンス要件を適用するのに役立てることができます。Amazon S3 Glacier には、ボールトロックポリシーの管理に使用する一連の API オペレーションが用意されています。「」S3 Glacier API を使用したボールトのロック。

ボールトロックポリシーの例として、ポリシーを削除する前に 1 年間そのアーカイブを保持するように指定されていると仮定します。この条件を導入するには、アーカイブが 1 年経過するまで、ユーザーにそのアーカイブを削除する権限を拒否するス許可を拒否するボールトロックポリシーを作成します。ポリシーをロックする前に、このポリシーをテストできます。ポリシーをロックすると、ポリシーは変更不可能になります。ボールトロック処理の詳細については、「ボールトロックポリシー」を参照してください。変更可能な他のユーザー権限を管理する場合は、ボールトアクセスポリシーを使用できます (「ボールトアクセスポリシー」を参照)。

S3 Glacier API、Amazon SDKs AWS CLI、または S3 Glacier コンソールを使用して、ボールトロックポリシーを作成および管理できます。ボールトリソースベースのポリシーに対して許可される S3 Glacier アクションのリストについては、「API の権限リファレンス」を参照してください。

例

- 例 1: 365 日経過していないアーカイブの削除権限を拒否する
- 例 2: タグに基づいて削除の権限を拒否します。

例 1: 365 日経過していないアーカイブの削除権限を拒否する

アーカイブが削除可能になる前に 1 年間保持しなければならない規制要件があるとします。次のボールトロックのポリシーを導入すると、その要件を適用できます。削除しようとしているアーカイブが 1 年経過していない場合は、ポリシーによって `glacier:DeleteArchive` アクションが拒否されます。ポリシーは、S3 Glacier-specific 固有の条件 キー `ArchiveAgeInDays` を使用して、1年間の保持要件を適用します。

```
{
                "Version":"2012-10-17",
                "Statement":[
                   {
                      "Sid": "deny-based-on-archive-age",
```

```
                "Principal": "*",
                "Effect": "Deny",
                "Action": "glacier:DeleteArchive",
                "Resource": [
                   "arn:aws:glacier:us-west-2:123456789012:vaults/examplevault"
                ],
                "Condition": {
                   "NumericLessThan" : {
                         "glacier:ArchiveAgeInDays" : "365"
                         }
                      }
                   }
                ]
             }
```

例 2: タグに基づいて削除の権限を拒否します。

1 年未満のアーカイブを削除可能にする時間ベースの保持ルールがあるとします。同時に、法的な調査が行われている間は、削除や変更を防ぐため、アーカイブを無期限にリーガルホールドの対象としなければならない場合があるとします。この場合、リーガルホールドはボールトロックポリシーで指定されている時間ベースの保持ルールよりも優先されます。

これら 2 つのルールを配置するために、次のポリシーの例では 2 つのステートメントが含まれています。

- 最初のステートメントは、全員の削除権限を拒否し、ボールトをロックします。このロックは `LegalHold` タグを使用して行われます。
- 2 番目のステートメントは、アーカイブが 365 日経過していない場合に削除の権限を付与します。ただし、アーカイブが 365 日経過していない場合でも、最初のステートメントの条件が満たされていれば、誰もアーカイブを削除することはできません。

```
             {
             "Version":"2012-10-17",
             "Statement":[
                {
                   "Sid": "lock-vault",
                   "Principal": "*",
                   "Effect": "Deny",
                   "Action": [
```

```
                "glacier:DeleteArchive"
            ],
            "Resource": [
                "arn:aws:glacier:us-west-2:123456789012:vaults/examplevault"
            ],
            "Condition": {
                "StringLike": {
                    "glacier:ResourceTag/LegalHold": [
                    "true",
                    ""
                    ]
                }
            }
            },
            {
            "Sid": "you-can-delete-archive-less-than-1-year-old",
            "Principal": {
                    "AWS": "arn:aws:iam::123456789012:root"
                },
            "Effect": "Allow",
            "Action": [
                "glacier:DeleteArchive"
            ],
            "Resource": [
                "arn:aws:glacier:us-west-2:123456789012:vaults/examplevault"
            ],
            "Condition": {
                "NumericLessThan": {
                    "glacier:ArchiveAgeInDays": "365"
                }
            }
            }
        ]
    }
```

Amazon S3 Glacier ID とアクセスのトラブルシューティング

次の情報は、S3 Glacier と IAM の使用に伴い発生する可能性のある、一般的な問題の診断や修復に役立ちます。

トピック

- S3 Glacier でアクションを実行するための権限がない

- iam:PassRole を実行する権限がありません
- 自分の 以外のユーザーに S3 Glacier リソース AWS アカウント へのアクセスを許可したい

S3 Glacier でアクションを実行するための権限がない

アクションを実行する権限がないというエラーが表示された場合は、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要なglacier:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 glacier:GetWidget on resource: my-example-widget
```

この場合、glacier:*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して S3 Glacier にロールを渡せるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、既存のロールをそのサービスに渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

marymajor という IAM ユーザーがコンソールを使用して S3 Glacier でアクションを実行しようすると、次のエラー例が発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

自分の 以外のユーザーに S3 Glacier リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- S3 Glacier でこれらの機能がサポートされるかどうかを確認するには、「Amazon S3 Glacier と IAM が連携する方法」を参照してください。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

API の権限リファレンス

Amazon S3 Glacier と IAM が連携する方法、および IAM アイデンティティ (アイデンティティベースのポリシー) またはリソース (リソースベースのポリシー) にアタッチできる書き込み権限ポリシーをセットアップしている場合は、以下の表をリファレンスとして使用できます。この一覧表示されます。 S3 AWS

ポリシーの `Action` 要素でアクションを指定し、ポリシーの `Resource` 要素でリソースの値を指定します。また、IAM ポリシー言語 `Condition` 要素を使用して、ポリシーをいつ適用するかを指定できます。

アクションを指定するには、API オペレーション名 (`glacier:CreateVault` など) の前に `glacier:` プレフィックスを使用します。ほとんどの S3 Glacier アクションで、`Resource` はアクセス許可を付与したいボールトです。ボールト ARN を使用して、`Resource` 値としてボールトを指定します。条件を表すには、あらかじめ定義された条件キーを使用します。詳細については、「S3 Glacier 内のリソースベースのポリシー」を参照してください。

次の表で、アイデンティティベースのポリシーとリソースベースのポリシーで使用できるアクションを一覧しています。

> **Note**
>
> アクションによっては、アイデンティティベースのポリシーでしか使用できないものもあります。そのようなアクションには、最初の列の API オペレーション名の後にアスタリスク (*) マークが付いています。

S3 Glacier API およびアクションで必要なアクセス権限

マルチパートアップロードの中止 (DELETE uploadID)

必要なアクセス権限 (API アクション) : `glacier:AbortMultipartUpload`

リソース: `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/vault-name`、`arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/example*`、`arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/*`

S3 Glacier の条件キー:

ボールトロックの中止 (ロックポリシーの DELETE)

必要なアクセス権限 (API アクション) : `glacier:AbortVaultLock`

リソース:

S3 Glacier の条件キー:

ボールトにタグを追加する (POST タグの追加)

必要なアクセス許可 (API アクション): `glacier:AddTagsToVault`

リソース: `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/vault-name`、`arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/example*`、`arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/*`

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

マルチパートアップロードの完了 (POST uploadID)

必要なアクセス許可 (API アクション): `glacier:CompleteMultipartUpload`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

ボールトロックの完了 (ロック ID の POST)

必要なアクセス許可 (API アクション): `glacier:CompleteVaultLock`

リソース:

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

ボールトの作成 (PUT vault) *

必要なアクセス許可 (API アクション): `glacier:CreateVault`

リソース:

S3 Glacier の条件キー:

アーカイブの削除 (DELETE archive)

必要なアクセス許可 (API アクション): `glacier:DeleteArchive`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ArchiveAgeInDays,glacier:ResourceTag/TagKey`

ボールトの削除 (DELETE vault)

必要なアクセス許可 (API アクション): `glacier:DeleteVault`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

ボールトアクセスポリシーの削除 (DELETE access-policy)

必要なアクセス許可 (API アクション): `glacier:DeleteVaultAccessPolicy`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

ボールト通知の削除 (通知設定の削除)

必要なアクセス許可 (API アクション): `glacier:DeleteVaultNotifications`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

ジョブの説明 (GET JobID)

必要なアクセス許可 (API アクション): `glacier:DescribeJob`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー:

ボールトの説明 (GET vault)

必要なアクセス許可 (API アクション): `glacier:DescribeVault`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー:

データ取り出しポリシーの取得 (ポリシーの GET) *

必要なアクセス許可 (API アクション): `glacier:GetDataRetrievalPolicy`

リソース: `arn:aws:glacier:region:account-id:policies/retrieval-limit-policy`

S3 Glacier の条件キー:

ジョブの出力の取得 (GET output)

必要なアクセス許可 (API アクション): `glacier:GetJobOutput`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー:

ボールトアクセスポリシー (GET access-policy) の取得

必要なアクセス許可 (API アクション): `glacier:GetVaultAccessPolicy`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー:

ボールトロックの取得 (ロックポリシーの GET)

必要なアクセス許可 (API アクション): `glacier:GetVaultLock`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー:

ボールト通知の取得 (GET notification-configuration)

必要なアクセス許可 (API アクション): `glacier:GetVaultNotifications`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー:

ジョブの開始 (ジョブの POST)

必要なアクセス許可 (API アクション): `glacier:InitiateJob`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ArchiveAgeInDays`,`glacier:ResourceTag/TagKey`

マルチパートアップロードの開始 (POST multipart-uploads)

必要なアクセス許可 (API アクション): `glacier:InitiateMultipartUpload`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

ボールトロックの開始 (ロックポリシーの POST)

必要なアクセス許可 (API アクション): `glacier:InitiateVaultLock`

リソース:

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

ジョブのリスト表示 (GET jobs)

必要なアクセス許可 (API アクション): `glacier:ListJobs`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー:

マルチパートアップロードのリスト (GET multipart-uploads)

必要なアクセス許可 (API アクション): `glacier:ListMultipartUploads`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー:

パートのリスト (GET uploadID)

必要なアクセス許可 (API アクション): `glacier:ListParts`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー:

ボールトのタグの一覧表示 (GET タグ)

必要なアクセス許可 (API アクション): `glacier:ListTagsForVault`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー:

ボールトのリスト (GET vaults)

必要なアクセス許可 (API アクション): `glacier:ListVaults`

リソース:

S3 Glacier の条件キー:

ボールトからタグを削除する (POST タグの削除)

必要なアクセス許可 (API アクション): `glacier:RemoveTagsFromVault`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

データ取り出しポリシーの設定 (ポリシーの PUT) *

必要なアクセス許可 (API アクション): `glacier:SetDataRetrievalPolicy`

リソース: `arn:aws:glacier:region:account-id:policies/retrieval-limit-policy`

S3 Glacier の条件キー:

ボールトアクセスポリシー (PUT access-policy) の設定

必要なアクセス許可 (API アクション): `glacier:SetVaultAccessPolicy`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

ボールトの通知設定の指定 (PUT notification-configuration)

必要なアクセス許可 (API アクション): `glacier:SetVaultNotifications`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

アーカイブのアップロード (POST archive)

必要なアクセス許可 (API アクション): `glacier:UploadArchive`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

パートのアップロード (PUT uploadID)

必要なアクセス許可 (API アクション): `glacier:UploadMultipartPart`

リソース: `arn:aws:glacier:region:account-id:vaults/vault-name`、`arn:aws:glacier:region:account-id:vaults/example*`、`arn:aws:glacier:region:account-id:vaults/*`

S3 Glacier の条件キー: `glacier:ResourceTag/TagKey`

Amazon S3 Glacier でのログ記録とモニタリング

モニタリングは、Amazon S3 Glacier (S3 Glacier) と AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。障害の原因が発生した場合は、その原因をより簡単に特定してデバッグできるように、 AWS ソリューションのすべての部分からモニタリングデータを収集する必要があります。 には、S3 Glacier リソースをモニタリングし、潜在的なインシデントに対応するための以下のツール AWS が用意されています。

Amazon CloudWatch アラーム

Amazon S3 を介して S3 Glacier を利用する場合、Amazon CloudWatch アラームを使用し、指定する期間中絶えず単一のメトリクスを監視できます。メトリクスが特定の閾値を超えると、Amazon SNS トピックまたは AWS Auto Scaling ポリシーに通知が送信されます。CloudWatch アラームは、特定の状態にあるという理由ではアクションを呼び出しません。状態が変わり、それが指定した期間だけ維持される必要があります。詳細については、「Amazon CloudWatch によるモニタリング」を参照してください。

AWS CloudTrail ログ

CloudTrail は、S3 Glacier のユーザー、ロール、または AWS のサービスによって実行されたアクションの記録を提供します。CloudTrail は、S3 Glacier コンソールからの呼び出しや S3 Glacier API へのコード呼び出しを含む、S3 Glacier のすべての API コールをイベントとしてキャプチャします。詳細については、「を使用した Amazon S3 Glacier API コールのログ記録 AWS CloudTrail」を参照してください。

AWS Trusted Advisor

Trusted Advisor は、数十万の AWS 顧客へのサービス提供から学んだベストプラクティスを活用します。 は AWS 、環境 Trusted Advisor を検査し、コスト削減、システムの可用性とパフォーマンスの向上、セキュリティギャップの解消に役立つ機会があれば、レコメンデーションを行います。すべての AWS お客様は 5 つの Trusted Advisor チェックにアクセスできます。ビジネスまたはエンタープライズサポートプランをご利用のお客様は、すべての Trusted Advisor チェックを表示できます。

詳細については、「サポート ユーザーガイド」の 「AWS Trusted Advisor」を参照してください。

Amazon S3 Glacier のコンプライアンス検証

Amazon S3 Glacier (S3 Glacier) のセキュリティとコンプライアンスは、以下を含む複数の AWS コンプライアンスプログラムの一環として、サードパーティーの監査者によって評価されます。

- System and Organization Controls (SOC)
- Payment Card Industry Data Security Standard (PCI DSS)
- Federal Risk and Authorization Management Program (FedRAMP)
- Health Insurance Portability and Accountability Act (HIPAA)

AWS は、コンプライアンスプログラムによる対象範囲内の AWS のサービスで、特定のコンプライアンスAWS プログラムの範囲内の のサービスのリストを頻繁に更新します。

サードパーティーの監査レポートは、 を使用してダウンロードできます AWS Artifact。詳細については、「 AWS Artifact ユーザーガイド」の AWS 「アーティファクトでのレポートのダウンロード」を参照してください。

AWS コンプライアンスプログラムの詳細については、AWS 「コンプライアンスプログラム」を参照してください。

S3 Glacier を使用する際のお客様のコンプライアンス責任は、組織のデータの機密性や組織のコンプライアンス目的、適用可能な法律、規制によって決定されます。S3 Glacier の使用が HIPAA、PCI、FedRAMP などの標準に準拠していることを前提としている場合、 AWS は以下を支援するリソースを提供します。

- S3 Glacier ボールトロック では、ボールトロックポリシーを使用して、S3 Glacier の各ボールトに対するコンプライアンス管理を簡単にデプロイして適用することができます。ボールトロックポリシーで「write once read many」(WORM) などのコントロールを指定して、ポリシーをロックし、今後編集できないようにします。ロックされると、そのポリシーは変更できなくなります。ボールトロックのポリシーは、SEC17a-4 や HIPAA などの規制フレームワークに準拠するのに役立ちます。
- セキュリティとコンプライアンスのクイックスタートガイドでは、セキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイするためのアーキテクチャ上の考慮事項と手順について説明します AWS。
- HIPAA セキュリティとコンプライアンスのアーキテクチャでは、企業が AWS を使用して HIPAA 要件を満たす方法の概要を説明します。

- AWS Well-Architected Tool (AWS WA Tool) は、 AWS ベストプラクティスを使用してアーキテクチャを確認および測定するための一貫したプロセスを提供するクラウド内のサービスです。 AWS WA Tool は、ワークロードの信頼性、安全性、効率、コスト効率を向上させるための推奨事項を提供します。
- AWS コンプライアンスリソースには、お客様の業界や地域に適用される可能性のあるいくつかのワークブックやガイドが用意されています。
- AWS Config を使用すると、社内プラクティス、業界ガイドライン、および規制に対するリソースの設定の準拠状態を評価できます。
- AWS Security Hub は、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。

Amazon S3 Glacier の耐障害性

AWS グローバルインフラストラクチャは、リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンでは、複数の物理的に独立し隔離されたアベイラビリティーゾーンが提供されており、それらは低レイテンシー、高スループット、高冗長性のネットワークにより接続されています。これらのアベイラビリティーゾーンを利用すると、アプリケーションとデータベースを効率的に設計して運用できます。アベイラビリティーゾーンは、従来の単一データセンターのインフラストラクチャや複数データセンターのインフラストラクチャよりも可用性、耐障害性、および拡張性が優れています。S3 Glacier は、少なくとも 3 つのアベイラビリティーゾーンにまたがる複数のデバイスに冗長的にデータを保存します。耐久性を高めるために、S3 Glacier は、アップロードが正常に実行されたことを確認する前に、複数の AZ 全体にデータを同期的に保存します。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

Amazon S3 Glacier のインフラストラクチャセキュリティ

マネージドサービスである Amazon S3 Glacier (S3 Glacier) は、「Amazon Web Services: セキュリティプロセスの概要」で説明されている AWS グローバルネットワークセキュリティ手順で保護されています。

ネットワーク経由で S3 Glacier にアクセスするには、 AWS 公開APIs を使用します。クライアントは Transport Layer Security (TLS) 1.2 をサポートしている必要があります。TLS 1.3 以降が推奨されます。また、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもサポートしている必要があります。これらのモードはJava 7 以降など、ほとんどの最新システムでサポートされています。また、リクエストには、IAM プリンシパルに関連付けられたアクセスキー ID およびシークレットアクセスキーによる署名が必要です。または、リクエストへの署名のために一時的にセキュリティ認証情報を生成する AWS Security Token Service (AWS STS) を使用することもできます。

VPC エンドポイント

Virtual Private Cloud (VPC) エンドポイントを使用すると、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続を必要とせずに、サポートされている AWS サービスや AWS PrivateLink を搭載した VPC エンドポイントサービスに VPC をプライベートに接続できます。S3 Glacier は VPC エンドポイントに直接対応していませんが、Amazon S3 と統合したストレージの S3 Glacier にアクセスする場合、Amazon S3 VPC エンドポイントを利用することができます。

Amazon S3ライフサイクル設定と S3 Glacier ストレージクラスへのオブジェクト移行の詳細については、Amazon Simple Storage Service User Guide の「オブジェクトのライフサイクル管理」と「オブジェクトの移行」を参照してください。VPC エンドポイント の詳細については、Amazon VPC ユーザーガイドの「VPC Endpoints」を参照してください。

S3 Glacier データ取得ポリシー

Amazon S3 Glacier データ取り出しポリシーを使用すると、データ取り出しクォータを簡単に設定し、各 AWS アカウント の 全体でデータ取り出しアクティビティを管理できます AWS リージョン。S3 Glacier のデータ取り出し料金の詳細については、「S3 Glacier 料金表」を参照してください。

> **Important**
>
> データ取り出しポリシーは、標準取り出しにのみ適用され、S3 Glacier に直接送信される取り出しリクエストを管理します。
> S3 Glacier 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「オブジェクトのアーカイブに適したストレージクラス」と「オブジェクトの移行」を参照してください。

トピック

- S3 Glacier データ取り出しポリシーの選択
- S3 Glacier コンソールを使用したデータ取り出しポリシーの設定
- Amazon S3 Glacier API を使用したデータ取り出しポリシーの設定

S3 Glacier データ取り出しポリシーの選択

データ取り出しポリシーは、[No Retrieval Limit]、[Free Tier Only]、[Max Retrieval Rate] の 3 つのタイプから選択できます。

[No Retrieval Limit] は取り出しに使用されるデフォルトのデータ取り出しポリシーです。[No Retrieval Limit] ポリシーを使用する場合、取り出しクォータが設定されず、あらゆる有効なデータ取り出しリクエストが許可されます。

無料利用枠のみのポリシーを使用すると、取得を 1 日あたりの AWS 無料利用枠の範囲内に維持でき、データ取得コストは発生しません。 AWS 無料利用枠の許容量よりも多くのデータを取得する場合は、最大取得レートポリシーを使用して、1 bytes-per-hour数のクォータを設定できます。 AWS リージョン でアカウント全体に [Max Retrieval Rate] ポリシーを設定すると、そのリージョン内のすべての取り出しジョブの最大取り出しレートが、設定した 1時間あたりの最大バイト数クォータを超えることはありません。

[Free Tier Only] と [Max Retrieval Rate] の両方のポリシーを使用する場合、指定した取り出しクォータを超えるデータ取り出しリクエストは拒否されます。[Free Tier Only] ポリシーを使用する場合、AWS 無料利用枠上限を超える取り出しリクエストは S3 Glacier により同時に拒否されます。[Max Retrieval Rate] ポリシーを使用する場合、進行中ジョブの最大取り出しレートが、ポリシーで設定した 1 時間あたりの最大バイト数のクォータを超える取り出しリクエストは S3 Glacier により拒否されます。これらのポリシーを使用することで、データ取り出しコストの管理を簡素化できます。

以下にデータ取り出しポリシーの使用時に役立つヒントをいくつか示します。

- データ取り出しポリシーの設定は、標準取り出しを使用した S3 Glacier からのデータの取り出しにかかる 3〜5 時間の期間に影響を与えません。
- 新しいデータ取り出しポリシーの設定は、その前に許可されてすでに進行中だった取り出しジョブに影響を与えません。
- 取り出しジョブリクエストがデータ取り出しポリシーによって拒否された場合、そのジョブまたはリクエストについて料金を請求されることはありません。
- それぞれに 1 つのデータ取り出しポリシーを設定できます。これにより AWS リージョン、 アカウントの AWS リージョン 内のすべてのデータ取り出しアクティビティが管理されます。データ取り出しポリシーは、データ取り出しコストが異なる AWS リージョン ため、特定の に固有です AWS リージョン。詳細については、「Amazon S3 Glacierの料金」を参照してください。

[Free Tier Only] ポリシー

データ取り出しポリシーを無料利用枠のみに設定して、データ取り出し料金が発生しないように、取り出しが常に AWS 無料利用枠の範囲内に収まるようにすることができます。取り出しリクエストが拒否された場合は、現在のデータ取り出しポリシーによってリクエストが拒否されたことを示すエラーメッセージが表示されます。

データ取り出しポリシーをリージョンごとに [Free Tier Only] に設定できます。このポリシーが設定されると、1 日に、その AWS リージョンの AWS 無料利用枠 (日割り計算) の取り出し許容量を超えてデータを取り出せなくなります。データ取り出し料金も発生しません。

データ取り出し料金が発生した後、同じ月に [Free Tier Only] ポリシーに切り替えることもできます。その場合、[Free Tier Only] ポリシーは新しい取り出しリクエストに対して適用されますが、過去のリクエストには影響しません。切り替え前に発生した料金は請求されます。

[Max Retrieval Rate] ポリシー

データ取り出しポリシーを [Max Retrieval Rate] に設定して、データ取り出しクォータを 1 時間あたりの最大バイト数として指定することで、最大取り出しレートを制御できます。データ取り出しポリシーを [Max Retrieval Rate] に設定すると、進行中ジョブの最大取り出しレートが、ポリシーで設定した 1 時間あたりの最大バイト数のクォータを超える取り出しリクエストは拒否されます。取り出しジョブリクエストが拒否された場合は、現在のデータ取り出しポリシーによってリクエストが拒否されたことを示すエラーメッセージが表示されます。

データ取り出しポリシーを最大取り出しレートポリシーに設定すると、1 日に使用できる AWS 無料利用枠の許容量に影響する可能性があります。たとえば、[Max Retrieval Rate] を 1 時間あたり 1 MB に設定したとします。これは、 AWS 無料利用枠ポリシーのレートよりも低くなります。毎日の AWS 無料利用枠枠を最大限に活用するには、まずポリシーを無料利用枠のみに設定し、必要に応じて後で最大取得レートポリシーに切り替えることができます。取り出し許容量の計算方法の詳細については、「Amazon S3 Glacier に関するよくある質問」を参照してください。

[No Retrieval Limit] ポリシー

データ取り出しポリシーを [No Retrieval Limit] ポリシーに設定した場合、すべての有効なデータ取り出しリクエストは許可されます。そのため、データ取り出しコストは使用状況によって変わります。

S3 Glacier コンソールを使用したデータ取り出しポリシーの設定

Amazon S3 Glacier コンソールを使用したデータ取り出しポリシーの作成方法

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/glacier/home で S3 Glacier コンソールを開きます。
2. 「リージョンの選択」で、ドロップダウンメニュー AWS リージョン から を選択します。それぞれにデータ取り出しポリシーを設定できます AWS リージョン。
3. 左側のナビゲーションペインで、[データ取り出し設定] を選択します。
4. [編集] を選択します。[データ取得ポリシーを編集] ページが表示されます。
5. [データ取得ポリシー] で、ポリシーを選択します。

   データ取り出しポリシーは、[取得制限なし]、[無料利用枠のみ]、[最大取得率を指定します] の 3 つのタイプから選択できます。

   - [取得制限なし] を選択すると、有効なデータ取り出し要求はすべて受け入れられます。

- 無料利用枠のみを選択した場合、 AWS 無料利用枠を超えるデータ取り出しリクエストは受け入れられません。
- [最大取得率を指定します] を選択した場合、進行中ジョブの最大取り出しレートが、指定した最大取り出しレートを超えるデータ取り出しリクエストは [最大取得率を指定します] を選択した時点で拒否されるようになります。[最大取得率] の [GB/時間] ボックスに 1 時間あたりのギガバイト (GB) の値を指定する必要があります。[GB/時間] に値を入力すると、コンソールで推定コストが計算されます。

6. [変更の保存] をクリックします。

Amazon S3 Glacier API を使用したデータ取り出しポリシーの設定

Amazon S3 Glacier REST API または AWS SDK を使用して、データ取り出しポリシーを表示したり設定したりできます。

Amazon S3 Glacier REST API を使用したデータ取り出しポリシーの設定

Amazon S3 Glacier REST API を使用して、データ取り出しポリシーを表示したり設定したりできます。データ取り出しポリシーの取得 (ポリシーの GET) オペレーションを使用して、既存のデータ取り出しポリシーを表示できます。データ取り出しポリシーの設定 (ポリシーの PUT) オペレーションを使用して、データ取り出しポリシーを設定します。

PUT ポリシーオペレーションを使用するときは、JSON `Strategy` フィールド値を、`BytesPerHour`、`FreeTier`、または `None` に設定することで、データ取り出しポリシーのタイプを選択します。`BytesPerHour` は、コンソールで [最大取得率を指定します] を選択すること、`FreeTier` は [無料利用枠のみ] を選択すること、`None` は [取得制限なし] を選択することに相当します。

ジョブの開始 (ジョブの POST) オペレーションを使用して、データ取り出しポリシーで設定した最大取り出しレートを超えるデータ取り出しジョブを開始すると、`Initiate Job` オペレーションは中止され、例外がスローされます。

AWS SDKs を使用したデータ取得ポリシーの設定

AWS にはSDKs が用意されています。 Amazon S3 これらの SDK では、基本となる REST API に対応するライブラリが提供され、簡単にリクエストを作成したりレスポンスを処理したりできるオブジェクトが提供されます。詳細については、「Amazon S3 Glacier での AWS SDKs の使用 Amazon S3 」を参照してください。

Amazon S3 Glacier リソースのタグ付け

タグは、 AWS リソースに割り当てるラベルです。タグはそれぞれ、1 つのキーと 1 つの値で構成されており、どちらもお客様側が定義します。定義するタグを、Amazon S3 Glacier (S3 Glacier) ボールトリソースに割り当てることができます。タグの使用は、請求データを含む AWS リソースを管理し、データを整理するためのシンプルで強力な方法です。

トピック

- タグ付けの基本
- タグの制限
- タグ付けを使用したコストの追跡
- タグ付けによるアクセス制御の管理
- 関連するセクション

タグ付けの基本

S3 Glacier コンソール、 AWS Command Line Interface （ AWS CLI ）、または S3 Glacier API を使用して、次のタスクを完了します。

- ボールトにタグを追加します
- ボールトのタグを一覧表示します
- ボールトからタグを削除します

タグを追加、一覧表示、削除する方法の詳細については、「S3 Glacier ボールトにタグを付ける」を参照してください。

タグを使用すると、ボールトを分類できます。たとえば、目的、所有者、環境などに基づいてボールトを分類できます。タグごとにキーと値を定義するため、特定のニーズを満たすためのカテゴリのカスタムセットを作成できます。たとえば、所有者と、ボールトの目的に基づいてボールトを追跡するのに役立つタグのセットを定義できます。以下に、タグのいくつかの例を示します。

- 所有者: 名前
- 目的: 動画のアーカイブ
- 環境:本稼働

タグの制限

基本的なタグの制限は次のとおりです。

- リソース (ボールト) のタグの最大数は 50 です。
- タグのキーと値は大文字と小文字が区別されます。

タグのキー制約は次のとおりです。

- ボールトの一連のタグ内で、各タグのキーは一意である必要があります。既に使用されているキーを含むタグを追加すると、新しいタグで、既存のキーと値のペアが上書きされます。
- タグキーは で始めることはできません。`aws:`このプレフィックスは で使用するために予約されています AWS。 AWS はユーザーに代わってこのプレフィックスで始まるタグを作成できますが、編集または削除することはできません。
- タグキーの長さは 1 ~ 128 文字 (Unicode) にする必要があります。
- タグ キーは、次の文字で構成する必要があります。Unicode 文字、数字、空白、特殊文字: (`_ . / = + - @`)。

タグ値の制約は次のとおりです。

- タグの長さは 0 ~ 255 文字 (Unicode) にする必要があります。
- タグ値は空白にすることができます。空白にしない場合は、次の文字で構成する必要があります。Unicode 文字、数字、空白、特殊文字: (`_ . / = + - @`)。

タグ付けを使用したコストの追跡

タグを使用して、 AWS コストを分類および追跡できます。ボールトを含む AWS リソースにタグを適用すると、 AWS コスト配分レポートにはタグ別に集計された使用量とコストが含まれます。自社のカテゴリ (例えばコストセンター、アプリケーション名、所有者) を表すタグを適用すると、複数のサービスにわたってコストを分類することができます。詳細については、AWS Billing ユーザーガイドのコスト配分タグを使用したカスタム請求レポートを参照してください。

タグ付けによるアクセス制御の管理

アクセスポリシーステートメントでは、タグを条件として使用できます。たとえば、リーガルホールドタグを設定して、「リーガルホールドタグの値が `True` に設定されている場合はアーカイブの削除

を拒否する」という条件としてデータ保持ポリシーに含めることができます。このデータ保持ポリシーをデプロイし、通常の条件としてリーガルホールドタグを `False` に設定できます。調査のためにデータをリーガルホールドの対象にする必要がある場合は、リーガルホールドタグの値を `True` に設定することによって簡単にリーガルホールドを有効にすることができます。その後にリーガルホールドの対象から外す場合も、同様の方法で行えます。詳細については、 IAM ユーザーガイドの「タグを使用したアクセス制御」を参照してください。

関連するセクション

- S3 Glacier ボールトにタグを付ける

を使用した Amazon S3 Glacier API コールのログ記録 AWS CloudTrail

Amazon S3 Glacier (S3 Glacier) は AWS CloudTrail、S3 Glacier のユーザー、ロール、またはのサービスによって実行されたアクションを記録する AWS サービスである と統合されています。CloudTrail は、S3 Glacier コンソールからの呼び出しや S3 Glacier API へのコード呼び出しを含む、S3 Glacier のすべての API コールをイベントとしてキャプチャします。証跡を作成する場合は、S3 Glacier のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、S3 Glacier に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

CloudTrail 内の Amazon S3 Glacier 情報

CloudTrail は、アカウントの作成 AWS アカウント 時に で有効になります。S3 Glacier でアクティビティが発生すると、そのアクティビティは CloudTrail イベントとイベント履歴の他の AWS サービスイベントに記録されます。で最近のイベントを表示、検索、ダウンロードできます AWS アカウント。詳細については、「CloudTrailイベント履歴でのイベントの表示」を参照してください。

S3 Glacier のイベントなど AWS アカウント、 のイベントの継続的な記録については、証跡を作成します。追跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、証跡はすべての AWS リージョンに適用されます。証跡は、 AWS パーティション内のすべての AWS リージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように、他の AWS サービスを設定できます。詳細については、以下を参照してください。

- 証跡を作成するための概要
- CloudTrail がサポートするサービスと統合
- CloudTrail 用 Amazon SNS 通知の構成
- 「複数のリージョンからCloudTrailログファイルを受け取る」および「複数のアカウントからCloudTrailログファイルを受け取る」

すべての S3 Glacier アクションは CloudTrail によってログに記録され、Amazon S3 Glacier の API リファレンス に記録されます。例えば、ボールトの作成 (PUT vault)、ボールトの削除 (DELETE vault)、ボールトのリスト (GET vaults) の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストが、ルートユーザーと他の認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、CloudTrail userIdentity 要素を参照してください。

Amazon S3 Glacier ログファイルエントリの概要

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントはあらゆるソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどの情報が含まれます。CloudTrail ログファイルは、公開 API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次は、ボールトの作成 (PUT vault)、ボールトの削除 (DELETE vault)、ボールトのリスト (GET vaults)および ボールトの説明 (GET vault) のアクションを示す CloudTrail ログエントリの例です。

```
{
    "Records": [
        {
            "awsRegion": "us-east-1",
            "eventID": "52f8c821-002e-4549-857f-8193a15246fa",
            "eventName": "CreateVault",
            "eventSource": "glacier.amazonaws.com",
            "eventTime": "2014-12-10T19:05:15Z",
            "eventType": "AwsApiCall",
            "eventVersion": "1.02",
            "recipientAccountId": "999999999999",
            "requestID": "HJiLgvfXCY88QJAC6rRoexS9ThvI21Q1Nqukfly02hcUPPo",
            "requestParameters": {
                "accountId": "-",
```

```
                "vaultName": "myVaultName"
            },
            "responseElements": {
                "location": "/999999999999/vaults/myVaultName"
            },
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/1.9.6 Mac_OS_X/10.9.5 Java_HotSpot(TM)_64-
Bit_Server_VM/25.25-b02/1.8.0_25",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "999999999999",
                "arn": "arn:aws:iam::999999999999:user/myUserName",
                "principalId": "A1B2C3D4E5F6G7EXAMPLE",
                "type": "IAMUser",
                "userName": "myUserName"
            }
        },
        {
            "awsRegion": "us-east-1",
            "eventID": "cdd33060-4758-416a-b7b9-dafd3afcec90",
            "eventName": "DeleteVault",
            "eventSource": "glacier.amazonaws.com",
            "eventTime": "2014-12-10T19:05:15Z",
            "eventType": "AwsApiCall",
            "eventVersion": "1.02",
            "recipientAccountId": "999999999999",
            "requestID": "GGdw-VfhVfLCFwAM6iVUvMQ6-fMwSqSO9FmRd0eRSa_Fc7c",
            "requestParameters": {
                "accountId": "-",
                "vaultName": "myVaultName"
            },
            "responseElements": null,
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/1.9.6 Mac_OS_X/10.9.5 Java_HotSpot(TM)_64-
Bit_Server_VM/25.25-b02/1.8.0_25",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "999999999999",
                "arn": "arn:aws:iam::999999999999:user/myUserName",
                "principalId": "A1B2C3D4E5F6G7EXAMPLE",
                "type": "IAMUser",
                "userName": "myUserName"
            }
        },
```

```
        {
            "awsRegion": "us-east-1",
            "eventID": "355750b4-e8b0-46be-9676-e786b1442470",
            "eventName": "ListVaults",
            "eventSource": "glacier.amazonaws.com",
            "eventTime": "2014-12-10T19:05:15Z",
            "eventType": "AwsApiCall",
            "eventVersion": "1.02",
            "recipientAccountId": "999999999999",
            "requestID": "yPTs22ghTsWprFivb-2u30FAaDALIZP17t4jM_xL9QJQyVA",
            "requestParameters": {
                "accountId": "-"
            },
            "responseElements": null,
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/1.9.6 Mac_OS_X/10.9.5 Java_HotSpot(TM)_64-
Bit_Server_VM/25.25-b02/1.8.0_25",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "999999999999",
                "arn": "arn:aws:iam::999999999999:user/myUserName",
                "principalId": "A1B2C3D4E5F6G7EXAMPLE",
                "type": "IAMUser",
                "userName": "myUserName"
            }
        },
        {
            "awsRegion": "us-east-1",
            "eventID": "569e830e-b075-4444-a826-aa8b0acad6c7",
            "eventName": "DescribeVault",
            "eventSource": "glacier.amazonaws.com",
            "eventTime": "2014-12-10T19:05:15Z",
            "eventType": "AwsApiCall",
            "eventVersion": "1.02",
            "recipientAccountId": "999999999999",
            "requestID": "QRt1ZdFLGn0TCm784HmKafBmcB2lVaV81UU3fsOR3PtoIiM",
            "requestParameters": {
                "accountId": "-",
                "vaultName": "myVaultName"
            },
            "responseElements": null,
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/1.9.6 Mac_OS_X/10.9.5 Java_HotSpot(TM)_64-
Bit_Server_VM/25.25-b02/1.8.0_25",
```

```
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "999999999999",
                "arn": "arn:aws:iam::999999999999:user/myUserName",
                "principalId": "A1B2C3D4E5F6G7EXAMPLE",
                "type": "IAMUser",
                "userName": "myUserName"
            }
        }
    ]
}
```

Amazon S3 Glacier の API リファレンス

Amazon S3 Glacier では、サービスを操作できる一連のオペレーション 特に、一連の RESTful API コール - をサポートしています。

HTTP リクエストを送信できる任意のプログラミングライブラリを使用して、REST リクエストを S3 Glacier に送信できます。REST リクエストを送信する場合、S3 Glacier では、リクエストに署名することで、すべてのリクエストを認証する必要があります。さらに、アーカイブをアップロードする場合は、ペイロードのチェックサムを計算し、リクエストに含める必要もあります。詳細については、「リクエストへの署名」を参照してください。

エラーが発生した場合は、エラーを処理できるように、S3 Glacier から送信されたエラーレスポンスの内容を把握する必要があります。このセクションでは、REST API 呼び出しを直接実行できるように、REST オペレーションの詳細に加え、こうした情報をすべて提供します。

REST API 呼び出しを直接使用することや、ラッパーライブラリを提供する Amazon SDK を使用することで、コーディングタスクを簡略化できます。これらのライブラリでは、送信する各リクエストに署名し、リクエストのペイロードのチェックサムを計算します。そのため、Amazon SDK を使用すると、コーディングタスクが簡略化されます。この開発者ガイドでは、 AWS SDK for Java と .NET を使用した基本的な S3 Glacier オペレーションの実例を示します。詳細については、「」を参照してくださいAmazon S3 Glacier での AWS SDKs の使用 Amazon S3 。

トピック

- 一般的なリクエストヘッダー
- 共通のレスポンスヘッダー
- リクエストへの署名
- チェックサムの計算
- エラーレスポンス
- ボールトオペレーション
- アーカイブオペレーション
- マルチパートアップロードオペレーション
- ジョブのオペレーション
- ジョブオペレーションで使用されるデータ型
- データ取り出しオペレーション

一般的なリクエストヘッダー

Amazon S3 Glacier (S3 Glacier) REST リクエストには、リクエストに関する基本的な情報を含むヘッダーが含まれています。次の表では、すべての S3 Glacier 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	必須
`Authorization`	リクエストに署名するために必要なヘッダー。S3 Glacier には、署名 バージョン 4 が必要です。詳細については、「リクエストへの署名」を参照してください。 タイプ: 文字列	はい
`Content-Length`	リクエストボディの長さ (ヘッダー以外)。 タイプ: 文字列 条件:アーカイブのアップロード (POST archive) API の場合にのみ必須です。	条件付き
`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。 Date ヘッダーを署名に使用しない場合は、RFC 2616 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。たとえば、`Wed, 10 Feb 2017 12:00:00 GMT` という日付/時間は、S3 Glacier で使用できる有効な日付/時間ヘッダーです。	条件付き

ヘッダー名	説明	必須
	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。 タイプ: 文字列 条件: 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、「署名バージョン 4 の日付の処理」(アマゾン ウェブ サービス用語集) を参照してください。	
Host	このヘッダーには、リクエストの送信先となるサービスエンドポイントを指定します。値はglacier.*region*.amazonaws.com 「」の形式である必要があります。*#####*は などの AWS リージョン指定に置き換えられますus-west-2 。 タイプ: 文字列	はい

ヘッダー名	説明	必須
x-amz-content-sha256	アーカイブのアップロード (POST archive)またはパートのアップロード (PUT uploadID)でアップロードされるペイロード全体の計算された SHA256 チェックサム。このヘッダーは x-amz-sha256-tree-hash ヘッダーと同じではありませんが、一部の小さいペイロードについては値が同じになります。x-amz-content-sha256 が必要な場合は、x-amz-content-sha256 と x-amz-sha256-tree-hash の両方を指定する必要があります。 タイプ: 文字列 条件: ストリーミング API (アーカイブのアップロード (POST archive)およびパートのアップロード (PUT uploadID)) の場合は必須です。	条件付き
x-amz-date	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。たとえば、20170210T120000Z という日付/時間は、S3 Glacier で使用できる有効な x-amz-date です。 タイプ: 文字列 条件: 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、「署名バージョン 4 の日付の処理」(アマゾンウェブ サービス用語集) を参照してください。	条件付き

ヘッダー名	説明	必須
x-amz-glacier-version	使用する S3 Glacier API バージョン。現在のバージョンは 2012-06-01 です。 タイプ: 文字列	はい
x-amz-sha256-tree-hash	アップロードしたアーカイブ (アーカイブのアップロード (POST archive)) またはアーカイブのパート (パートのアップロード (PUT uploadID)) の計算された SHA256 木構造ハッシュのチェックサム。このチェックサムの計算の詳細については、「チェックサムの計算」を参照してください。 タイプ: 文字列 デフォルト: なし 条件: アーカイブのアップロード (POST archive)およびパートのアップロード (PUT uploadID)の場合は必須です。	条件付き

共通のレスポンスヘッダー

以下の表は、ほとんどの API 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
Content-Length	レスポンス本文の長さ (バイト単位) 。 タイプ: 文字列
Date	Amazon S3 Glacier (S3 Glacier) が応答した日時。たとえば、Wed, 10 Feb 2017 12:00:00 GMT。日付の形式は、RFC 2616 のセクション 3.3 で指定されている完全な日付の形式のいずれかであることが必要です。返される Date は、他の日付とわずかに異なるものになることがあるため、たとえば、アーカイブのアップロード (POST archive) リクエストから返される日

名前	説明
	付は、ボールトのインベントリリストにあるアーカイブに表示される日付と一致しないことがあります。 タイプ: 文字列
`x-amzn-RequestId`	S3 Glacier 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。S3 Glacier に問題がある場合、 はこの値を使用して問題のトラブルシューティング AWS を行うことができます。この値は、記録しておくことをお勧めします。 タイプ: 文字列
`x-amz-sha256-tree-hash`	アーカイブまたはインベントリ本文の SHA256 木構造ハッシュのチェックサム。このチェックサムの計算の詳細については、「チェックサムの計算」を参照してください。 タイプ: 文字列

リクエストへの署名

S3 Glacier では、リクエストに署名することで、送信するすべてのリクエストを認証する必要があります。リクエストに署名するには、暗号化ハッシュ関数を使用してデジタル署名を計算します。暗号化ハッシュは、入力データから一意のハッシュ値生成して返す関数です。ハッシュ関数に渡される入力データとしては、リクエストのテキスト、およびシークレットアクセスキーが該当します。ハッシュ関数から返されるハッシュ値をリクエストに署名として含めます。署名は、リクエストの `Authorization` ヘッダーの一部です。

S3 Glacier または は、リクエストを受け取ると、リクエストの署名に使用されたものと同じハッシュ関数と入力を使用して署名を再計算します。再計算された署名とリクエスト内の署名が一致した場合、S3 Glacier はリクエストを処理します。それ以外の場合、リクエストは拒否されます。

S3 Glacier は、AWS 署名バージョン 4 を使用した認証をサポートします。署名の計算プロセスは 3 つのタスクに分けることができます。

- タスク 1: 正規リクエストを作成する

HTTP リクエストを正規形式に変換します。S3 Glacier では、送信された署名と比較するために署名を再計算するときに正規形式が使用されるので、同じ正規形式を使用する必要があります。

- タスク 2: 署名文字列を作成する

  暗号化ハッシュ関数への入力値の 1 つとして使用する文字列を作成します。署名文字列と呼ばれる文字列は、ハッシュアルゴリズムの名前、要求日付、認証情報スコープの文字列、および前のタスクで正規化されたリクエストを結合したものです。認証情報スコープ文字列自体は、日付、AWS リージョン、およびサービス情報の連結です。

- タスク 3: 署名を作成する

  2 つの入力文字列 (署名文字列と派生キー) を受け付ける暗号化ハッシュ関数を使用して、リクエストの署名を作成します。シークレットアクセスキーから開始し、認証情報スコープの文字列を使用して一連のハッシュベースのメッセージ認証コード (HMAC) を作成することで、派生キーが計算されます。この署名手順で使用するハッシュ関数は、データのアップロードのための S3 Glacier API で使用する木構造ハッシュアルゴリズムではありません。

トピック

- 署名の計算例
- ストリーミングオペレーションの署名の計算

署名の計算例

次の例で、ボールトの作成 (PUT vault) の署名を作成する詳細な手順を示します。実際の署名計算方法を確認するときに、この例を参考にしてください。詳細については、IAM ユーザーガイドの AWS API リクエストの署名を参照してください。

例では、次のように想定しています。

- リクエストのタイムスタンプは `Fri, 25 May 2012 00:24:53 GMT`
- エンドポイントは米国東部 (バージニア北部) リージョン `us-east-1`

リクエストの一般的な構文 (JSON の本体を含む) は次のとおりです。

```
PUT /-/vaults/examplevault HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
Date: Fri, 25 May 2012 00:24:53 GMT
```

```
Authorization: SignatureToBeCalculated
x-amz-glacier-version: 2012-06-01
```

「タスク 1: 正規リクエストを作成する」で計算されるリクエストの正規形式は次のとおりです。

```
PUT
/-/vaults/examplevault

host:glacier.us-east-1.amazonaws.com
x-amz-date:20120525T002453Z
x-amz-glacier-version:2012-06-01

host;x-amz-date;x-amz-glacier-version
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

正規リクエストの最後の行はリクエストボディのハッシュです。また、正規リクエストの 3 行目が空であることに注意してください。これは、この API のクエリパラメータがないためです。

「タスク 2: 署名する文字列を作成する」で署名する文字列は次のとおりです。

```
AWS4-HMAC-SHA256
20120525T002453Z
20120525/us-east-1/glacier/aws4_request
5f1da1a2d0feb614dd03d71e87928b8e449ac87614479332aced3a701f916743
```

署名する文字列の最初の行はアルゴリズム、2 行目はタイムスタンプ、3 行目は認証情報スコープ、最後の行は「タスク 1: 正規リクエストを作成する」で作成した正規リクエストのハッシュです。認証情報スコープで使用するサービス名は `glacier` です。

「タスク 3: 署名を作成する」について、派生キーは次のように表されます。

```
derived key = HMAC(HMAC(HMAC(HMAC("AWS4" + YourSecretAccessKey,"20120525"),"us-
east-1"),"glacier"),"aws4_request")
```

シークレットアクセスキー `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY` を使用する場合、計算された署名は次のようになります。

```
3ce5b2f2fffac9262b4da9256f8d086b4aaf42eba5f111c21681a65a127b7c2a
```

最後のステップは、Authorization ヘッダーの構築です。デモンストレーションのアクセスキー AKIAIOSFODNN7EXAMPLE の場合、ヘッダーは次のとおりです (読みやすいように改行しています)。

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-east-1/
glacier/aws4_request,
SignedHeaders=host;x-amz-date;x-amz-glacier-version,
Signature=3ce5b2f2fffac9262b4da9256f8d086b4aaf42eba5f111c21681a65a127b7c2a
```

ストリーミングオペレーションの署名の計算

アーカイブのアップロード (POST archive) と パートのアップロード (PUT uploadID) の 2 つのストリーミングオペレーションでは、リクエストに署名して送信する際に、追加のヘッダーとして x-amz-content-sha256 を含める必要があります。ストリーミングオペレーションの署名手順は、ストリーミング用のヘッダーを追加することを除いて、他のオペレーションとまったく同じです。

ストリーミングヘッダー x-amz-content-sha256 の計算は、アップロードされる全コンテンツ (ペイロード) の SHA256 ハッシュに基づいています。この計算は、SHA256 木構造ハッシュとは異なりますのでご注意ください (チェックサムの計算)。ペイロードデータの SHA 256 ハッシュ値は、基本的にはペイロードデータの SHA256 木構造ハッシュと異なるものになります。

ペイロードデータがバイトの配列として指定されている場合には、以下の Java コードスニペットを使用して SHA256 ハッシュを計算できます。

```
public static byte[] computePayloadSHA256Hash2(byte[] payload) throws
 NoSuchAlgorithmException, IOException {
    BufferedInputStream bis =
       new BufferedInputStream(new ByteArrayInputStream(payload));
    MessageDigest messageDigest = MessageDigest.getInstance("SHA-256");
    byte[] buffer = new byte[4096];
    int bytesRead = -1;
    while ( (bytesRead = bis.read(buffer, 0, buffer.length)) != -1 ) {
        messageDigest.update(buffer, 0, bytesRead);
    }
    return messageDigest.digest();
}
```

これと同じく、C# では、以下のコードスニペットに示すように、ペイロードデータの SHA256 ハッシュを計算できます。

```
public static byte[] CalculateSHA256Hash(byte[] payload)
{
    SHA256 sha256 = System.Security.Cryptography.SHA256.Create();
    byte[] hash = sha256.ComputeHash(payload);

    return hash;
}
```

ストリーミング API の署名の計算例

以下の例は、2 つある S3 Glacier のストリーミング API の 1 つ、アーカイブのアップロード (POST archive) の署名の作成に関する詳細を示したものです。例では、次のように想定しています。

- リクエストのタイムスタンプは `Mon, 07 May 2012 00:00:00 GMT`
- エンドポイントは米国東部 (バージニア北部) リージョン us-east-1 です。
- コンテンツのペイロードは、文字列「Welcome to S3 Glacier」です。

(JSON 本文も含めた) 一般的なリクエストの構文を以下の例に示します。 `x-amz-content-sha256` ヘッダーが含まれていることに注意してください。この単純な例では、`x-amz-sha256-tree-hash` と `x-amz-content-sha256` は同じ値となっています。ただし、このことは、1 MB を超えるアーカイブをアップロードする場合には必ずしも当てはまりません。

```
POST /-/vaults/examplevault HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
Date: Mon, 07 May 2012 00:00:00 GMT
x-amz-archive-description: my archive
x-amz-sha256-tree-hash: SHA256 tree hash
x-amz-content-sha256: SHA256 payload hash
Authorization: SignatureToBeCalculated
x-amz-glacier-version: 2012-06-01
```

「タスク 1: 正規リクエストを作成する」で計算されるリクエストの正規形式は、以下のとおりです。ストリーミングヘッダー `x-amz-content-sha256` が値に含まれていることにご注意ください。これは、ペイロードを読み取って SHA256 ハッシュを計算してから、署名を計算する必要があることを示しています。

```
POST
/-/vaults/examplevault

host:glacier.us-east-1.amazonaws.com
x-amz-content-sha256:726e392cb4d09924dbad1cc0ba3b00c3643d03d14cb4b823e2f041cff612a628
x-amz-date:20120507T000000Z
x-amz-glacier-version:2012-06-01

host;x-amz-content-sha256;x-amz-date;x-amz-glacier-version
726e392cb4d09924dbad1cc0ba3b00c3643d03d14cb4b823e2f041cff612a628
```

署名の計算手順の残りの部分については、「署名の計算例」で説明しています。Authorization ヘッダーはシークレットアクセスキー wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY を使用しています。アクセスキー AKIAIOSFODNN7EXAMPLE は、以下に示すとおりです (読みやすくなるよう、改行を追加しています)。

```
Authorization=AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20120507/us-east-1/glacier/aws4_request,
SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-amz-glacier-version,
Signature=b092397439375d59119072764a1e9a144677c43d9906fd98a5742c57a2855de6
```

チェックサムの計算

アーカイブをアップロードする場合は、x-amz-sha256-tree-hash ヘッダーと x-amz-content-sha256 ヘッダーを両方とも含める必要があります。x-amz-sha256-tree-hash ヘッダーは、リクエストボディのペイロードのチェックサムです。このトピックでは、x-amz-sha256-tree-hash ヘッダーを計算する方法について説明します。x-amz-content-sha256 ヘッダーはペイロード全体のハッシュであり、認可に必要です。詳細については、「ストリーミング API の署名の計算例」を参照してください。

リクエストのペイロードは以下のようになります。

- アーカイブ全体 - アーカイブのアップロード API を使用して単一のリクエストでアーカイブをアップロードする場合は、リクエストボディでアーカイブ全体を送信します。この場合は、アーカイブ全体のチェックサムを含める必要があります。

- アーカイブのパート - マルチパートアップロード API を使用してアーカイブをパート単位でアップロードする場合は、リクエストボディでアーカイブのパートを 1 つのみ送信します。この場合は、アーカイブのパートのチェックサムを含めます。すべてのパートをアップロードしたら、マルチパートアップロードの完了リクエストを送信します。これにはアーカイブ全体のチェックサムを含める必要があります。

ペイロードのチェックサムは、SHA-256 木構造ハッシュです。チェックサムの計算中に SHA-256 ハッシュ値の木構造を計算することから、木構造ハッシュと呼ばれます。ルートのハッシュ値はアーカイブ全体のチェックサムです。

> **Note**
>
> このセクションでは、SHA-256 木構造ハッシュを計算する方法を説明します。ただし、同じ結果になる限り、任意の方法を使用できます。

次のように、SHA-256 木構造ハッシュを計算します。

1. ペイロードデータの 1 MB のチャンクごとに、SHA-256 ハッシュを計算します。データの最後のチャンクは 1 MB を下回ることがあります。たとえば、3.2 MB のアーカイブをアップロードする場合、データの最初の 3 個の 1 MB のチャンクごとに SHA-256 ハッシュ値を計算してから、残りの 0.2 MB のデータの SHA-256 ハッシュを計算します。これらのハッシュ値は木構造の葉ノードを構成します。
2. 木構造の次のレベルを作成します。
   a. 2 つの連続した子ノードのハッシュ値を連結し、連結したハッシュ値の SHA-256 ハッシュを計算します。この連結と SHA-256 ハッシュの生成により、2 個の子ノードの親ノードが作成されます。
   b. 子ノードが 1 個だけ残った場合は、そのハッシュ値を木構造の次のレベルに昇格させます。
3. 結果の木構造にルートが含まれるまで、ステップ 2 を繰り返します。木構造のルートではアーカイブ全体のハッシュが提供され、サブツリーのルートではマルチパートアップロードの対応するパートのハッシュが提供されます。

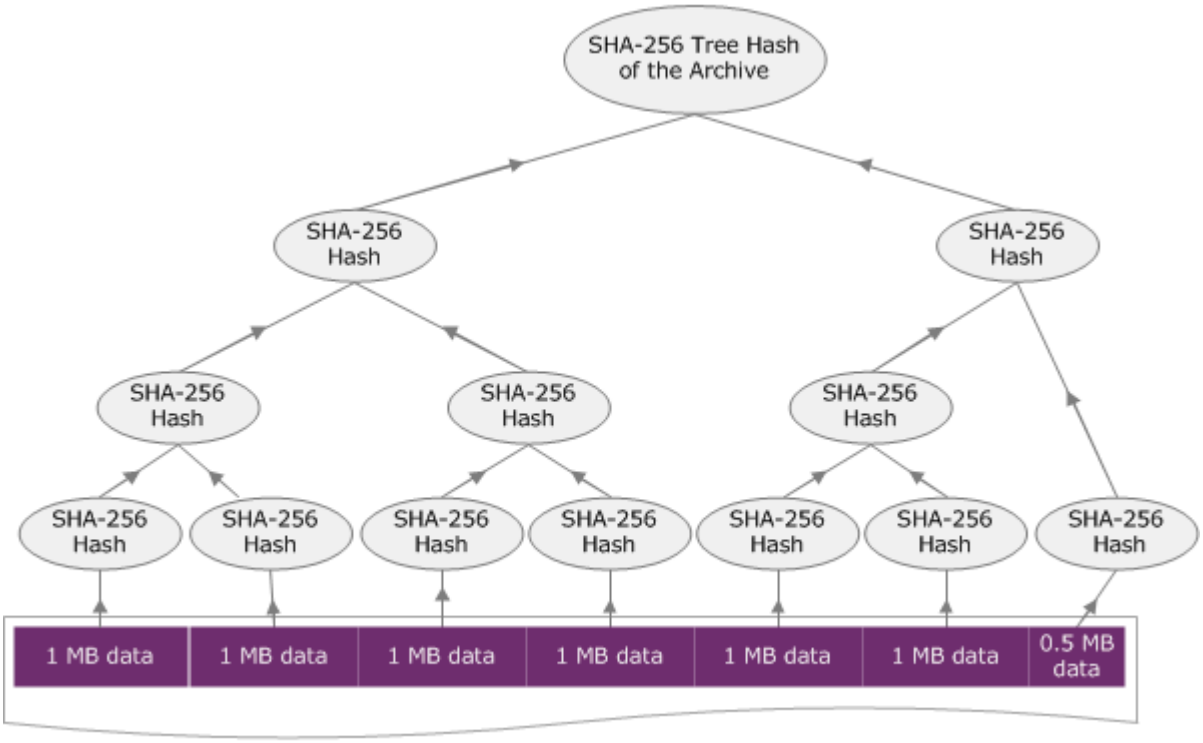
トピック

- 木構造ハッシュの例 1: 単一のリクエストでのアーカイブのアップロード

- 木構造ハッシュの例 2: マルチパートアップロードを使用したアーカイブのアップロード
- ファイルの木構造ハッシュの計算
- データをダウンロードするときのチェックサムの受信

木構造ハッシュの例 1: 単一のリクエストでのアーカイブのアップロード

アーカイブのアップロード API を使用して単一のリクエストでアーカイブをアップロードする場合 (「アーカイブのアップロード (POST archive)」を参照)、リクエストのペイロードにはアーカイブ全体が含まれます。このため、アーカイブ全体の木構造ハッシュを `x-amz-sha256-tree-hash` リクエストヘッダーに含める必要があります。6.5 MB のアーカイブをアップロードするとします。次の図は、アーカイブの SHA-256 ハッシュを作成するプロセスを示しています。アーカイブを読み取り、1 MB のチャンクそれぞれの SHA-256 ハッシュを計算します。残りの 0.5 MB のデータのハッシュも計算し、前の手順で説明したように木構造を作成します。

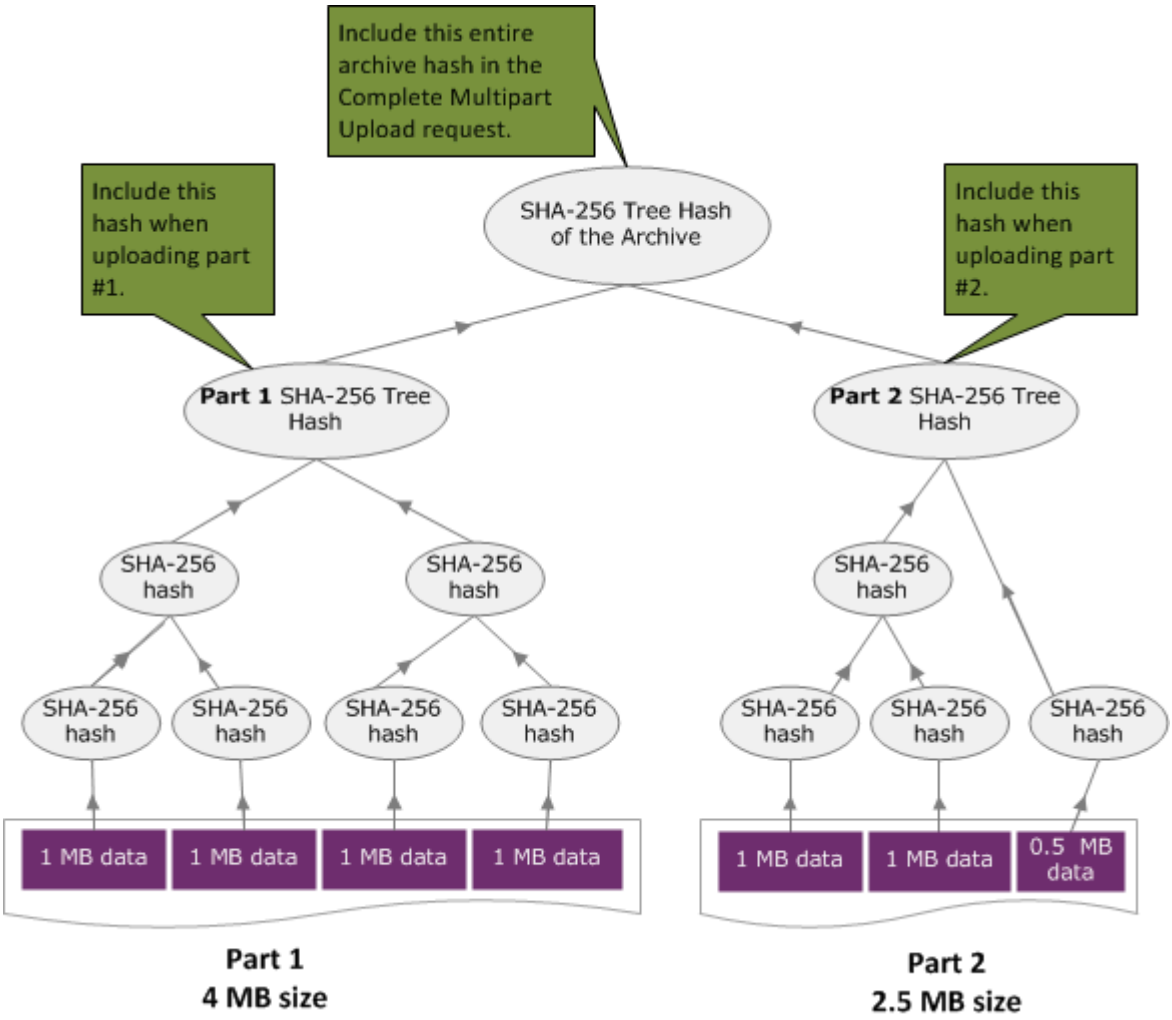


6.5 MB Archive

木構造ハッシュの例 2: マルチパートアップロードを使用したアーカイブのアップロード

マルチパートアップロードでアーカイブをアップロードする場合の木構造ハッシュの計算のプロセスは、単一のリクエストでアーカイブをアップロードする場合と同じです。唯一の違いは、 (パートの

アップロード (PUT uploadID) API を使用して) 各リクエストでアーカイブのパートを 1 つのみアップロードする点です。したがって、そのパートのチェックサムのみを `x-amz-sha256-tree-hash` リクエストヘッダーに含めます。ただし、すべてのパートをアップロードした後で、 マルチパートアップロードの完了 (POST uploadID)リクエストヘッダーにアーカイブ全体の木構造ハッシュを含めたマルチパートアップロードの完了 (「`x-amz-sha256-tree-hash`」を参照) リクエストを送信する必要があります。



ファイルの木構造ハッシュの計算

以下に示すアルゴリズムは、デモンストレーションのために選択したものです。実装シナリオでは、必要に応じてコードを最適化できます。Amazon S3 Glacier (S3 Glacier) で Amazon SDK を使用してプログラミングする場合は、木構造ハッシュの計算が自動的に行われるため、必要な作業はファイルの参照を指定することのみです。

Example 1: Java の例

以下の例は、Java を使用してファイルの SHA256 木構造ハッシュを計算する方法を示しています。この例は、ファイルの場所を引数として指定するか、コードから直接 `TreeHashExample.computeSHA256TreeHash` メソッドを使用することで実行できます。

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class TreeHashExample {

static final int ONE_MB = 1024 * 1024;

    /**
     * Compute the Hex representation of the SHA-256 tree hash for the specified
     * File
     *
     * @param args
     *            args[0]: a file to compute a SHA-256 tree hash for
     */
    public static void main(String[] args) {

        if (args.length < 1) {
            System.err.println("Missing required filename argument");
            System.exit(-1);
        }

        File inputFile = new File(args[0]);
        try {

            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 Tree Hash = %s\n", toHex(treeHash));

        } catch (IOException ioe) {
            System.err.format("Exception when reading from file %s: %s", inputFile,
                    ioe.getMessage());
            System.exit(-1);

        } catch (NoSuchAlgorithmException nsae) {
            System.err.format("Cannot locate MessageDigest algorithm for SHA-256: %s",
```

```
                    nsae.getMessage());
            System.exit(-1);
        }
    }

    /**
     * Computes the SHA-256 tree hash for the given file
     *
     * @param inputFile
     *            a File to compute the SHA-256 tree hash for
     * @return a byte[] containing the SHA-256 tree hash
     * @throws IOException
     *             Thrown if there's an issue reading the input file
     * @throws NoSuchAlgorithmException
     */
    public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
            NoSuchAlgorithmException {

        byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
        return computeSHA256TreeHash(chunkSHA256Hashes);
    }

    /**
     * Computes a SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk even if it is smaller than 1 MB.
     *
     * @param file
     *            A file to compute checksums on
     * @return a byte[][] containing the checksums of each 1 MB chunk
     * @throws IOException
     *             Thrown if there's an IOException when reading the file
     * @throws NoSuchAlgorithmException
     *             Thrown if SHA-256 MessageDigest can't be found
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");

        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }
```

```
        if (numChunks == 0) {
            return new byte[][] { md.digest() };
        }

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

        try {
            fileStream = new FileInputStream(file);
            byte[] buff = new byte[ONE_MB];

            int bytesRead;
            int idx = 0;
            int offset = 0;

            while ((bytesRead = fileStream.read(buff, offset, ONE_MB)) > 0) {
                md.reset();
                md.update(buff, 0, bytesRead);
                chunkSHA256Hashes[idx++] = md.digest();
                offset += bytesRead;
            }

            return chunkSHA256Hashes;

        } finally {
            if (fileStream != null) {
                try {
                    fileStream.close();
                } catch (IOException ioe) {
                    System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                            ioe.getMessage());
                }
            }
        }
    }

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     *
     * This method uses a pair of arrays to iteratively compute the tree hash
     * level by level. Each iteration takes two adjacent elements from the
     * previous level source array, computes the SHA-256 hash on their
```

```
     * concatenated value and places the result in the next level's destination
     * array. At the end of an iteration, the destination array becomes the
     * source array for the next level.
     *
     * @param chunkSHA256Hashes
     *            An array of SHA-256 checksums
     * @return A byte[] containing the SHA-256 tree hash for the input chunks
     * @throws NoSuchAlgorithmException
     *             Thrown if SHA-256 MessageDigest can't be found
     */
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");

        byte[][] prevLvlHashes = chunkSHA256Hashes;

        while (prevLvlHashes.length > 1) {

            int len = prevLvlHashes.length / 2;
            if (prevLvlHashes.length % 2 != 0) {
                len++;
            }

            byte[][] currLvlHashes = new byte[len][];

            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

                // If there are at least two elements remaining
                if (prevLvlHashes.length - i > 1) {

                    // Calculate a digest of the concatenated nodes
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();

                } else { // Take care of remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }

            prevLvlHashes = currLvlHashes;
```

```
        }

        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     *
     * @param data
     *            a byte[] to convert to Hex characters
     * @return A String containing Hex characters
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);

        for (int i = 0; i < data.length; i++) {
            String hex = Integer.toHexString(data[i] & 0xFF);

            if (hex.length() == 1) {
                // Append leading zero.
                sb.append("0");
            }
            sb.append(hex);
        }
        return sb.toString().toLowerCase();
    }
}
```

Example 2: C# .NET の例

以下の例は、ファイルの SHA256 木構造ハッシュを計算する方法を示しています。この例は、ファイルの場所を引数として指定して実行できます。

```
using System;
using System.IO;

using System.Security.Cryptography;

namespace ExampleTreeHash
{
    class Program
    {
        static int ONE_MB = 1024 * 1024;
```

```
        /**
        * Compute the Hex representation of the SHA-256 tree hash for the
        * specified file
        *
        * @param args
        *            args[0]: a file to compute a SHA-256 tree hash for
        */
        public static void Main(string[] args)
        {
            if (args.Length < 1)
            {
                Console.WriteLine("Missing required filename argument");
                Environment.Exit(-1);
            }
            FileStream inputFile = File.Open(args[0], FileMode.Open, FileAccess.Read);
            try
            {
                byte[] treeHash = ComputeSHA256TreeHash(inputFile);
                Console.WriteLine("SHA-256 Tree Hash = {0}",
BitConverter.ToString(treeHash).Replace("-", "").ToLower());
                Console.ReadLine();
                Environment.Exit(-1);
            }
            catch (IOException ioe)
            {
                Console.WriteLine("Exception when reading from file {0}: {1}",
                    inputFile, ioe.Message);
                Console.ReadLine();
                Environment.Exit(-1);
            }
            catch (Exception e)
            {
                Console.WriteLine("Cannot locate MessageDigest algorithm for SHA-256:
{0}",
                    e.Message);
                Console.WriteLine(e.GetType());
                Console.ReadLine();
                Environment.Exit(-1);
            }
            Console.ReadLine();
        }
```

```
        /**
         * Computes the SHA-256 tree hash for the given file
         *
         * @param inputFile
         *            A file to compute the SHA-256 tree hash for
         * @return a byte[] containing the SHA-256 tree hash
         */
        public static byte[] ComputeSHA256TreeHash(FileStream inputFile)
        {
            byte[][] chunkSHA256Hashes = GetChunkSHA256Hashes(inputFile);
            return ComputeSHA256TreeHash(chunkSHA256Hashes);
        }


        /**
         * Computes a SHA256 checksum for each 1 MB chunk of the input file. This
         * includes the checksum for the last chunk even if it is smaller than 1 MB.
         *
         * @param file
         *            A file to compute checksums on
         * @return a byte[][] containing the checksums of each 1MB chunk
         */
        public static byte[][] GetChunkSHA256Hashes(FileStream file)
        {
            long numChunks = file.Length / ONE_MB;
            if (file.Length % ONE_MB > 0)
            {
                numChunks++;
            }

            if (numChunks == 0)
            {
                return new byte[][] { CalculateSHA256Hash(null, 0) };
            }
            byte[][] chunkSHA256Hashes = new byte[(int)numChunks][];

            try
            {
                byte[] buff = new byte[ONE_MB];

                int bytesRead;
                int idx = 0;

                while ((bytesRead = file.Read(buff, 0, ONE_MB)) > 0)
```

```
                {
                    chunkSHA256Hashes[idx++] = CalculateSHA256Hash(buff, bytesRead);
                }
                return chunkSHA256Hashes;
            }
            finally
            {
                if (file != null)
                {
                    try
                    {
                        file.Close();
                    }
                    catch (IOException ioe)
                    {
                        throw ioe;
                    }
                }
            }

        }

        /**
         * Computes the SHA-256 tree hash for the passed array of 1MB chunk
         * checksums.
         *
         * This method uses a pair of arrays to iteratively compute the tree hash
         * level by level. Each iteration takes two adjacent elements from the
         * previous level source array, computes the SHA-256 hash on their
         * concatenated value and places the result in the next level's destination
         * array. At the end of an iteration, the destination array becomes the
         * source array for the next level.
         *
         * @param chunkSHA256Hashes
         *            An array of SHA-256 checksums
         * @return A byte[] containing the SHA-256 tree hash for the input chunks
         */
        public static byte[] ComputeSHA256TreeHash(byte[][] chunkSHA256Hashes)
        {
            byte[][] prevLvlHashes = chunkSHA256Hashes;
            while (prevLvlHashes.GetLength(0) > 1)
            {

                int len = prevLvlHashes.GetLength(0) / 2;
```

```
                if (prevLvlHashes.GetLength(0) % 2 != 0)
                {
                    len++;
                }

                byte[][] currLvlHashes = new byte[len][];

                int j = 0;
                for (int i = 0; i < prevLvlHashes.GetLength(0); i = i + 2, j++)
                {

                    // If there are at least two elements remaining
                    if (prevLvlHashes.GetLength(0) - i > 1)
                    {

                        // Calculate a digest of the concatenated nodes
                        byte[] firstPart = prevLvlHashes[i];
                        byte[] secondPart = prevLvlHashes[i + 1];
                        byte[] concatenation = new byte[firstPart.Length +
secondPart.Length];
                        System.Buffer.BlockCopy(firstPart, 0, concatenation, 0,
firstPart.Length);
                        System.Buffer.BlockCopy(secondPart, 0, concatenation,
firstPart.Length, secondPart.Length);

                        currLvlHashes[j] = CalculateSHA256Hash(concatenation,
concatenation.Length);

                    }
                    else
                    { // Take care of remaining odd chunk
                        currLvlHashes[j] = prevLvlHashes[i];
                    }
                }

                prevLvlHashes = currLvlHashes;
            }

            return prevLvlHashes[0];
        }

        public static byte[] CalculateSHA256Hash(byte[] inputBytes, int count)
        {
            SHA256 sha256 = System.Security.Cryptography.SHA256.Create();
```

```
            byte[] hash = sha256.ComputeHash(inputBytes, 0, count);
            return hash;
        }
    }
}
```

データをダウンロードするときのチェックサムの受信

ジョブの開始 API を使用してアーカイブを取得する場合は (「ジョブの開始 (ジョブの POST)」を参照)、オプションでアーカイブの取得範囲を指定できます。同様に、ジョブの出力の取得 API を使用してデータをダウンロードする場合は (「ジョブの出力の取得 (GET output)」を参照)、オプションでダウンロードするデータの範囲を指定できます。これらの範囲には、アーカイブのデータを取得およびダウンロードする際に理解していることが重要な 2 つの特性があります。取得する範囲は、アーカイブに対してメガバイト単位に調整する必要があります。データをダウンロードしたときにチェックサム値を受け取るには、取得する範囲とダウンロードする範囲が両方とも木構造ハッシュ可能である必要があります。この 2 つのタイプの範囲の調整は、次のように定義されています。

- メガバイト整列-範囲 [StartByte,EndBytes] は、メガバイト (1024*1024) が揃うときStartBytes1MBで割り切れるしEndBytesプラス 1 は 1 MB で割り切れるか、指定されたアーカイブの終わり (アーカイブのバイトサイズから 1 を引いたもの) と等しくなります。ジョブの開始 API で使用する範囲 (指定した場合) は、メガバイト単位に調整する必要があります。
- 木構造ハッシュ可能 - 範囲 [StartBytes, EndBytes] は、その範囲で構築された木構造ハッシュのルートがアーカイブ全体の木構造ハッシュ内のノードに相当する場合のみ、アーカイブに対して木構造ハッシュ可能です。ダウンロードしたデータのチェックサム値を受け取るには、取得する範囲とダウンロードする範囲が両方とも木構造ハッシュ可能である必要があります。範囲の例およびアーカイブ木構造ハッシュとの関係については、「木構造ハッシュの例: 木構造ハッシュ可能なアーカイブの範囲を取得する」を参照してください。

  木構造ハッシュ可能な範囲は、メガバイト単位にも調整できることに注意してください。ただし、メガバイト単位に調整された範囲が木構造ハッシュ可能であるとは限りません。

以下は、アーカイブデータをダウンロードしたときにチェックサムを受け取る場合を示しています。

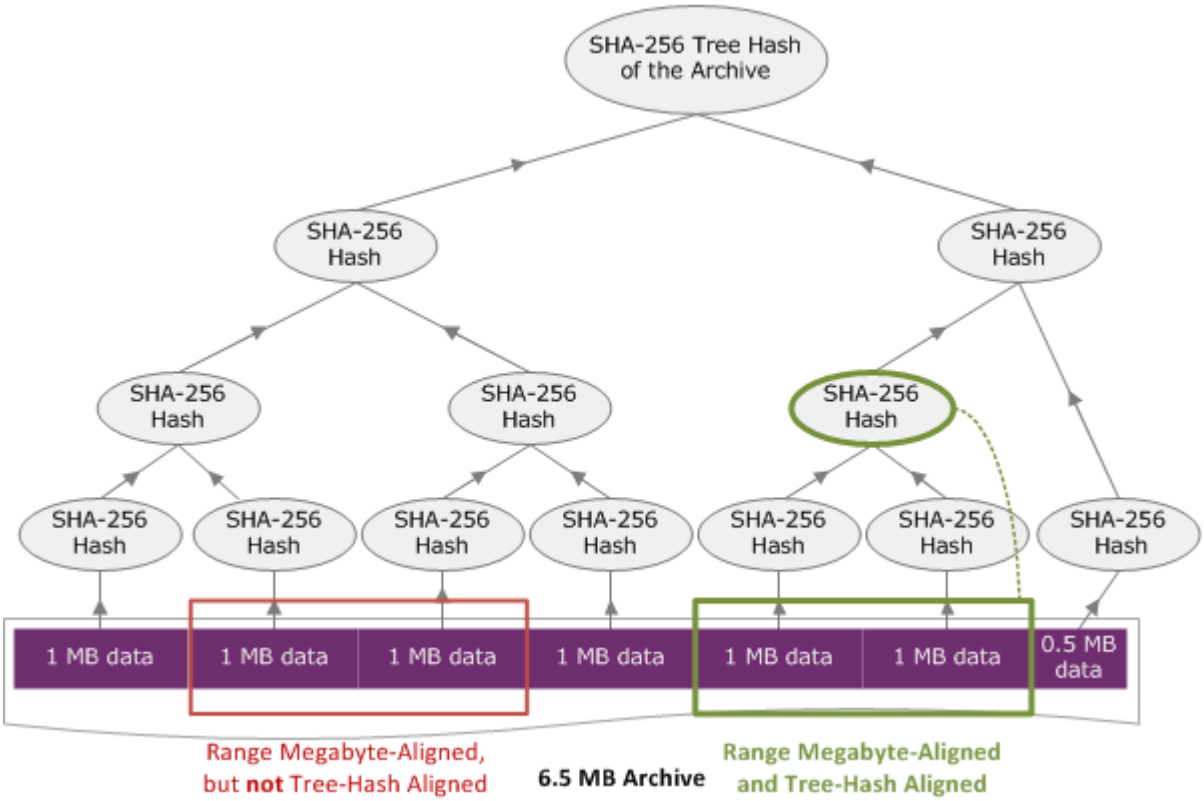
- ジョブの開始リクエストで取得する範囲を指定せず、ジョブの取得リクエストでアーカイブ全体をダウンロードした場合。
- ジョブの開始リクエストで取得する範囲を指定せず、ジョブの取得リクエストでダウンロードする木構造ハッシュ可能な範囲を指定した場合。

- ジョブの開始リクエストで取得する木構造ハッシュ可能な範囲を指定し、ジョブの取得リクエストでその範囲全体をダウンロードした場合。
- ジョブの開始リクエストで取得する木構造ハッシュ可能な範囲を指定し、ジョブの取得リクエストでダウンロードする木構造ハッシュ可能な範囲を指定した場合。

ジョブの開始リクエストで取得する範囲を指定し、その範囲が木構造ハッシュ可能ではない場合は、ジョブの取得リクエストでデータをダウンロードしたときにアーカイブデータを取得できますが、チェックサム値は返されません。

木構造ハッシュの例: 木構造ハッシュ可能なアーカイブの範囲を取得する

ボールト内に 6.5 MB のアーカイブがあり、アーカイブの 2 MB 分を取得するとします。ジョブの開始リクエストで 2 MB の範囲を指定する方法によって、データのダウンロード時にデータチェックサム値を受け取るかどうかが決定されます。次の図は、6.5 MB のアーカイブに対してダウンロードできる 2 つの 2 MB の範囲を示しています。両方ともメガバイト単位に調整されていますが、木構造ハッシュ可能な範囲は 1 つのみです。



木構造ハッシュ可能な範囲の指定

このセクションでは、木構造ハッシュ可能な範囲の正確な指定について説明します。木構造ハッシュ可能な範囲は、アーカイブの一部分をダウンロードするときに、取得する範囲のデータと、取得したデータからダウンロードする範囲を指定する場合に重要です。これらの範囲が両方とも木構造ハッシュ可能である場合は、データをダウンロードしたときにチェックサムデータを受け取ります。

範囲 [A,B] は、新しい木構造ハッシュが[A,B]の上に構築されるとき、その範囲の木ハッシュのルートがアーカイブ全体の木構造ハッシュ内のノードに相当する場合のみ、アーカイブに関してアラインされています。これについては、「木構造ハッシュの例: 木構造ハッシュ可能なアーカイブの範囲を取得する」の図に示されています。このセクションでは、木構造ハッシュ可能な範囲の指定について説明します。

[P, Q) を、N メガバイト (MB) のアーカイブの範囲クエリとします。P および Q は 1 MB の倍数です。実際に含まれる範囲は [P MB,Q MB 1 バイト] ですが、単純化のために、[P, Q) と表しています。これらの前提に立つと、次のようになります。

- P が奇数の場合、木構造ハッシュ可能な範囲は 1 つのみ、つまり [P, P + 1 MB) です。
- もしP.偶数を指定しますk.は、最大数です。P.2.k.*Xとすると、最大でも存在しているk.木構造ハッシュ可能な範囲P.。Xは、0 より大きい整数です。木構造ハッシュ可能な範囲は、次のカテゴリに含まれます。
  - それぞれの i に対して、(0 <= i <= k) で、$P + 2^i < N$ の場合、$[P, Q + 2^i)$ は木構造ハッシュ可能な範囲です。
  - P = 0 は、A = 2[lgN]*0 である特殊なケースです。

エラーレスポンス

API は、エラーが発生すると以下の例外のいずれか 1 つを返します。

コード	説明	HTTP ステータスコード	タイプ
`AccessDeniedException`	AWS Identity and Access Management (IAM) ポリシーで許可されていないリソースにアクセスしようとした場合、またはリク	`403 Forbidden`	クライアント

コード	説明	HTTP ステータスコード	タイプ
	エスト URI で間違った AWS アカウント ID が使用された場合に返されます。詳細については、「Amazon S3 Glacier の ID とアクセス管理」を参照してください。		
BadRequest	リクエストを処理することができない場合に返されます。	400 Bad Request	クライアント
ExpiredTokenException	リクエストで使用しているセキュリティトークンが期限切れになっている場合に返されます。	403 Forbidden	クライアント
InsufficientCapacityException	迅速リクエストを処理する容量が足りない場合に返されます。このエラーは、迅速取り出しにのみ該当し、標準取り出しまたは大容量取り出しには該当しません。	503 Service Unavailable	[Server] (サーバー)
InvalidParameterValueException	リクエストのパラメータの指定が不正である場合に返されます。	400 Bad Request	クライアント
InvalidSignatureException	リクエストの署名が無効である場合に返されます。	403 Forbidden	クライアント
LimitExceededException	リクエストが、ボールトの制限、タグの制限、またはプロビジョニングされた容量の制限のいずれかを超えている場合に返されます。	400 Bad Request	クライアント

コード	説明	HTTP ステータスコード	タイプ
MissingAuthenticationTokenException	認証データがリクエストにない場合に返されます。	400 Bad Request	クライアント
MissingParameterValueException	必須のヘッダーまたはパラメータがリクエストにない場合に返されます。	400 Bad Request	クライアント
PolicyEnforcedException	取り出しジョブが現在のデータポリシーの取り出しレート制限を超えた場合に返されます。データ取り出しポリシーの詳細については、「S3 Glacier データ取得ポリシー」を参照してください。	400 Bad Request	クライアント
ResourceNotFoundException	ボールト、アップロード ID、ジョブ ID など、指定されたリソースがない場合に返されます。	404 Not Found	クライアント
RequestTimeoutException	アーカイブをアップロードし、Amazon S3 Glacier (S3 Glacier) がアップロードを受信する際にタイムアウトとなった場合に返されます。	408 Request Timeout	クライアント
SerializationException	リクエストの本文が無効になっている場合に返されます。JSON ペイロードが含まれる場合には、正しい形式になっていることを確認します。	400 Bad Request	クライアント
ServiceUnavailableException	サービスがリクエストを完了できない場合に返されます。	500 Internal Server Error	[Server] (サーバー)

コード	説明	HTTP ステータスコード	タイプ
ThrottlingException	S3 Glacier に対するリクエストのレートを減らす必要がある場合に返されます。	400 Bad Request	クライアント
UnrecognizedClientException	アクセスキー ID またはセキュリティトークンが無効である場合に返されます。	400 Bad Request	クライアント

さまざまな S3 Glacier API によって、同じ例外が返されるものの、例外メッセージにはさまざまなものが用意されているため、発生した特定のエラーのトラブルシューティングに役立てることができます。

S3 Glacier は、レスポンス本文でエラーに関する情報を返します。以下の例では、エラーレスポンスの例をいくつか挙げて説明します。

例 1: 存在しないジョブ ID を使用したジョブリクエストに関する説明

存在しないジョブについて ジョブの説明 (GET JobID) リクエストを送信したとします。つまり、存在していないジョブ ID を指定したとします。

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVEXAMPLEbadJobID HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスでは、S3 Glacier によって以下のエラーレスポンスが返されます。

```
HTTP/1.1 404 Not Found
x-amzn-RequestId: AAABaZ9N92Iiyv4N7sru3ABEpSQkuFtmH3NP6aAC51ixfjg
Content-Type: application/json
Content-Length: 185
Date: Wed, 10 Feb 2017 12:00:00 GMT
{
```

```
  "code": "ResourceNotFoundException",
  "message": "The job ID was not found: HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVEXAMPLEbadJobID",
  "type": "Client"
  }
```

コードの説明は以下のとおりです。

コード

一般的な例外の 1 つです。

型: 文字列

メッセージ

エラーを返した API 特有のエラー発生条件の一般的な説明です。

型: 文字列

Type

エラーの原因です。このフィールドの値は、`Client`、`Server`、`Unknown` のいずれかになります。

型: 文字列

前述のレスポンスでは次の点に注意してください。

- このエラーレスポンスでは、S3 Glacier によりステータスコードのうち、`4xx` と`5xx` の値が返されます。この例では、ステータスコードは `404 Not Found` です。
- `Content-Type` ヘッダーの値 `application/json` は、本文の JSON を示しています。
- 本文の JSON には、エラーに関する情報が表示されます。

前述のリクエストで、誤ったジョブ ID ではなく、存在しないボールトを指定したとします。すると、レスポンスには別のメッセージが返されます。

```
HTTP/1.1 404 Not Found
x-amzn-RequestId: AAABBeC9Zw0rp_5D0L8VfB3FA_WlTupqTKAUehMcPhdgni0
Content-Type: application/json
Content-Length: 154
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

```
{
  "code": "ResourceNotFoundException",
  "message": "Vault not found for ARN: arn:aws:glacier:us-west-2:012345678901:vaults/
examplevault",
  "type": "Client"
}
```

例 2: リクエストパラメータに無効な値が設定されたジョブリクエストの一覧表示

この例では、ジョブのリスト表示 (GET jobs) リクエストを送信して、ボールトジョブと特定の statuscode を取得します。このとき、statuscode の値として許容される値 finished、InProgress、または Succeeded のいずれでもなく、誤って Failed を指定したとします。

```
GET /-/vaults/examplevault/jobs?statuscode=finished HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

S3 Glacier によって、適切なメッセージと併せて InvalidParameterValueException が返されます。

```
HTTP/1.1 400 Bad Request
x-amzn-RequestId: AAABaZ9N92Iiyv4N7sru3ABEpSQkuFtmH3NP6aAC51ixfjg
Content-Type: application/json
Content-Length: 141
Date: Wed, 10 Feb 2017 12:00:00 GMT
{
  "code": "InvalidParameterValueException",
  "message": "The job status code is not valid: finished",
  "type: "Client"
}
```

ボールトオペレーション

以下は、S3 Glacier で使用できるボールトオペレーションです。

トピック

- ボールトロックの中止 (ロックポリシーの DELETE)
- ボールトにタグを追加する (POST タグの追加)
- ボールトの作成 (PUT vault)
- ボールトロックの完了 (ロック ID の POST)
- ボールトの削除 (DELETE vault)
- ボールトアクセスポリシーの削除 (DELETE access-policy)
- ボールト通知の削除 (通知設定の削除)
- ボールトの説明 (GET vault)
- ボールトアクセスポリシー (GET access-policy) の取得
- ボールトロックの取得 (ロックポリシーの GET)
- ボールト通知の取得 (GET notification-configuration)
- ボールトロックの開始 (ロックポリシーの POST)
- ボールトのタグの一覧表示 (GET タグ)
- ボールトのリスト (GET vaults)
- ボールトからタグを削除する (POST タグの削除)
- ボールトアクセスポリシー (PUT access-policy) の設定
- ボールトの通知設定の指定 (PUT notification-configuration)

ボールトロックの中止 (ロックポリシーの DELETE)

説明

このオペレーションでは、ボールトロックが `Locked` 状態でない場合にボールトロック処理を中止します。ボールトロックが `Locked` 状態のときにこのオペレーションを要求した場合は、`AccessDeniedException` エラーが返されます。ボールトロック処理を中止すると、指定したボールトからボールトロックポリシーが削除されます。

ボールトロックは、`InProgress`を要求することによって ボールトロックの開始 (ロックポリシーの POST) 状態になります。ボールトロックは、`Locked`を要求することによって ボールトロックの完了 (ロック ID の POST) 状態になります。ボールトロックの状態は、ボールトロックの取得 (ロックポリシーの GET)を要求することによって取得できます。ボールトロック処理の詳細については、「S3 Glacier ボールトロック」を参照してください。ボールトロックポリシーの詳細については、「ボールトロックポリシー」を参照してください。

このオペレーションはべき等です。ボールトロックが InProgress 状態の場合やボールトにポリシーが関連付けられていない場合は、このオペレーションを複数回正常に実行することができます。

リクエスト

ボールトロックポリシーを削除するには、ボールトの DELETE サブリソースの URI に HTTP lock-policy リクエストを送信します。

構文

```
DELETE /AccountId/vaults/vaultName/lock-policy HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

ポリシーが正常に削除された場合、S3 Glacier は HTTP 204 No Content レスポンスを返します。

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

次の例は、ボールトロック処理を中止する方法を示しています。

リクエストの例

この例では、DELETE リクエストが **examplevault** というボールトの `lock-policy` サブリソースに送信されます。

```
DELETE /-/vaults/examplevault/lock-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
x-amz-glacier-version: 2012-06-01
```

レスポンスの例

ポリシーが正常に削除された場合、次の例に示すように、S3 Glacier は `HTTP 204 No Content` レスポンスを返します。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

関連するセクション

- ボールトロックの完了 (ロック ID の POST)
- ボールトロックの取得 (ロックポリシーの GET)
- ボールトロックの開始 (ロックポリシーの POST)

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトにタグを追加する (POST タグの追加)

このオペレーションでは、指定したタグをボールトに追加します。各タグはキーと値で構成されます。各ボールトは、最大 50 個のタグを持つことができます。リクエストによりボールトのタグの制限を超える場合、オペレーションは `LimitExceededException` エラーをスローします。

ボールトで、指定したキーの下にタグがすでに存在する場合、既存のキーの値は上書きされます。タグの詳細については、Amazon S3 Glacier リソースのタグ付けを参照してください。

リクエストの構文

ボールトにタグを追加するには、次の構文例に示すように、タグの URI に HTTP POST リクエストを送信します。

```
POST /AccountId/vaults/vaultName/tags?operation=add HTTP/1.1
Host: glacier.Region.amazonaws.com
```

```
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01

{
   "Tags":
      {
         "string": "string",
         "string": "string"
      }
}
```

> **Note**
>
> AccountId 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

リクエストパラメータ

名前	説明	必須
operation=add	値 operation を持つ 1 つのクエリ文字パラメータ add が、ボールトからタグを削除する (POST タグの削除) からこれを区別します。	はい

リクエストヘッダー

このオペレーションは、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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

リクエストボディ には、次の JSON フィールドが含まれます。

[タグ]

ボールトに追加するタグ。各タグはキーと値で構成されます。値は空の文字列とすることができます。

タイプ: 文字列から文字列へのマッピング

長さの制限: 最小長は 1 です。最大長は 10 です。

必須: はい

レスポンス

オペレーションリクエストが成功した場合、サービスは HTTP 応答 `204 No Content` を返します。

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

次の例では、タグとともに HTTP POST リクエストを送信し、ボールトに追加します。

```
POST /-/vaults/examplevault/tags?operation=add HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
Content-Length: length
x-amz-glacier-version: 2012-06-01

{
  "Tags":
    {
       "examplekey1": "examplevalue1",
       "examplekey2": "examplevalue2"
    }
}
```

レスポンスの例

リクエストが成功した場合、次の例に示しているように、S3 Glacier は `HTTP 204 No Content` を返します。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

関連するセクション

- ボールトのタグの一覧表示 (GET タグ)
- ボールトからタグを削除する (POST タグの削除)

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトの作成 (PUT vault)

説明

このオペレーションでは、指定した名前の新しいボールトを作成します。 ボールトの名前は、 の AWS リージョン内で一意である必要があります AWS アカウント。アカウントにつき最大 1,000 個のボールトを作成できます。ボールトの追加作成の詳細については、 Amazon S3 Glacier の製品詳細ページを参照してください。

ボールトの名前を指定する際は、以下のガイドラインに従う必要があります。

- 名前は 1 ~ 255 文字の長さです。
- 使用できる文字は、a-z、A-Z、0-9、'_' (アンダースコア)、'-' (ハイフン)、'.' (ピリオド) です。

このオペレーションはべき等です。同じリクエストを何度も送信できますが、指定したボールトが最初に Amazon S3 Glacier (S3 Glacier) によって作成された後は、リクエストがあってもそれ以上項目への影響はありません。

リクエスト

構文

ボールトを作成するには、作成するボールトの URI に HTTP PUT リクエストを送信します。

```
PUT /AccountId/vaults/VaultName HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

このオペレーションのリクエストボディを空（0 バイト）にする必要があります。

レスポンス

構文

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
```

レスポンスヘッダー

成功したレスポンスには、すべての操作に共通のレスポンスヘッダーに加えて、次のレスポンスヘッダーが含まれます。共通のレスポンスヘッダーの詳細については、「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
Location	作成されたボールトの相対 URI パス。 タイプ: 文字列

レスポンス本文

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

次の例では、HTTP PUT リクエストを送信して、`examplevault` というボールトを作成します。

```
PUT /-/vaults/examplevault HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Content-Length: 0
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

S3 Glacier によりボールトが作成され、`Location` ヘッダーにボールトの相対 URI パスが返されます。リクエストでアカウント ID を指定したか、ハイフン ("`Location`") を指定したかに関係なく、- ヘッダーにはアカウント ID が常に表示されます。

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
Location: /111122223333/vaults/examplevault
```

関連するセクション

- ボールトのリスト (GET vaults)
- ボールトの削除 (DELETE vault)
- Amazon S3 Glacier の ID とアクセス管理

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトロックの完了 (ロック ID の POST)

説明

このオペレーションでは、ボールトロックを InProgress 状態から Locked 状態にすることによってボールトロック処理を完了します。これにより、ボールトロックポリシーは変更できなくなります。ボールトロックは、InProgressを要求することによって ボールトロックの開始 (ロックポリシーの POST) 状態になります。ボールトロックの状態は、ボールトロックの取得 (ロックポリシーの GET)を要求することによって取得できます。ボールトロック処理の詳細については、「S3 Glacier ボールトロック」を参照してください。

このオペレーションはべき等です。このリクエストは、ボールトロックが Locked 状態にあり、指定したロック ID がボールトのロックに使用されたロック ID と一致する場合は常に成功します。

ボールトロックが Locked 状態のときにリクエストで無効なロック ID が渡された場合は、AccessDeniedException エラーが返されます。ボールトロックが InProgress 状態のときにリクエストで無効なロック ID が渡された場合は、InvalidParameter エラーがスローされます。

リクエスト

ボールトロック処理を完了するには、ボールトの POST サブリソースの URI に有効なロック ID を指定した HTTP lock-policy リクエストを送信します。

構文

```
POST /AccountId/vaults/vaultName/lock-policy/lockId HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier は

リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

`lockId` の値は、ボールトロックの開始 (ロックポリシーの POST)リクエストによって取得したロック ID です。

リクエストパラメータ

リクエストヘッダー

このオペレーションは、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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

オペレーションリクエストが成功した場合、サービスは HTTP 応答 `204 No Content` を返します。

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

次の例では、ボールトロック処理を完了するためにロック ID を指定した HTTP POST リクエストを送信します。

```
POST /-/vaults/examplevault/lock-policy/AE863rKkWZU53SLW5be4DUcW HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
Content-Length: length
x-amz-glacier-version: 2012-06-01
```

レスポンスの例

リクエストが成功した場合、次の例に示すように、Amazon S3 Glacier (S3 Glacier) は `HTTP 204 No Content` レスポンスを返します。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

関連するセクション

- ボールトロックの中止 (ロックポリシーの DELETE)
- ボールトロックの取得 (ロックポリシーの GET)
- ボールトロックの開始 (ロックポリシーの POST)

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトの削除 (DELETE vault)

説明

この操作は、ボールトを削除します。このオペレーションは、ボールトを削除するものです。Amazon S3 Glacier (S3 Glacier) では、最後にインベントリを作成した時点でボールト内にアーカイブがなく、また、最後にインベントリを作成してからボールトへの書き込みがない場合にのみ、ボールトを削除できます。ここに挙げた条件のいずれかが満たされない場合、ボールトの削除は失敗し (つまり、ボールトは削除されず)、S3 Glacier によってエラーが返されます。

ボールト内のアーカイブの数など、ボールトに関する情報を取得するには、ボールトの説明 (GET vault) オペレーションを使用できます。ただし、その情報は S3 Glacier によって最後に生成されたボールトインベントリに基づきます。

このオペレーションはべき等です。

> **Note**
>
> ボールトを削除すると、ボールトにアタッチされているボールトアクセスポリシーも削除されます。ボールトアクセスポリシーの詳細については、「ボールトアクセスポリシー」を参照してください。

リクエスト

ボールトを削除するには、ボールトリソース URI に DELETE リクエストを送信します。

構文

```
DELETE /AccountId/vaults/VaultName HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

次の例では、examplevault というボールトを削除します。例のリクエストは、削除するリソース (ボールト) の URI に対する DELETE リクエストです。

```
DELETE /-/vaults/examplevault HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

関連するセクション

- ボールトの作成 (PUT vault)
- ボールトのリスト (GET vaults)
- ジョブの開始 (ジョブの POST)
- Amazon S3 Glacier の ID とアクセス管理

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトアクセスポリシーの削除 (DELETE access-policy)

説明

このオペレーションでは、指定されたボールトに関連付けられたアクセスポリシーを削除します。このオペレーションは、最終的には一貫性のある結果になります。ただし、Amazon S3 Glacier (S3 Glacier) の側でアクセスポリシーを完全に削除するまでには、多少時間がかかります。このため、削除リクエストを送信してからも少しの間は、ポリシーの効果が継続することがあります。

このオペレーションはべき等です。ボールトに関連付けられたポリシーがない場合でも、複数回 delete を呼び出すことができます。ボールトアクセスポリシーの詳細については、「ボールトアクセスポリシー」を参照してください。

リクエスト

現在のボールトアクセスポリシーを削除するには、ボールトの `access-policy` サブリソースの URI に HTTP `DELETE` リクエストを送信します。

構文

```
DELETE /AccountId/vaults/vaultName/access-policy HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

レスポンスでは、ポリシーが正常に削除された場合、S3 Glacier は `204 No Content` を返します。

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

以下の例では、ボールトアクセスポリシーの削除方法を示しています。

リクエストの例

この例では、`DELETE` リクエストが **`examplevault`** というボールトの `access-policy` サブリソースに送信されます。

```
DELETE /-/vaults/examplevault/access-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
```

```
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
x-amz-glacier-version: 2012-06-01
```

レスポンスの例

レスポンスでは、ポリシーが正常に削除された場合、次の例に示しているように、S3 Glacier は204 No Content を返します。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

関連するセクション

- ボールトアクセスポリシー (GET access-policy) の取得
- ボールトアクセスポリシー (PUT access-policy) の設定

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールト通知の削除 (通知設定の削除)

説明

このオペレーションは、ボールトに設定されている通知設定を削除するものですボールトの通知設定の指定 (PUT notification-configuration)。このオペレーションは、最終的には一貫性のある結果になります。ただし、Amazon S3 Glacier (S3 Glacier) が通知を完全に無効にするまでには、多少時間がかかります。このため、削除リクエストを送信してからも少しの間は、一部の通知が継続することがあります。

リクエスト

ボールトの通知設定を削除するには、ボールトの DELETE サブリソースに notification-configuration リクエストを送信します。

構文

```
DELETE /AccountId/vaults/VaultName/notification-configuration HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

以下の例は、ボールトの通知設定を削除する方法を示したものです。

リクエストの例

この例では、DELETE リクエストが notification-configuration というボールトの examplevault サブリソースに送信されます。

```
DELETE /111122223333/vaults/examplevault/notification-configuration HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

関連するセクション

- ボールト通知の取得 (GET notification-configuration)
- ボールトの通知設定の指定 (PUT notification-configuration)
- Amazon S3 Glacier の ID とアクセス管理

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトの説明 (GET vault)

説明

このオペレーションでは、ボールトの Amazon リソースネーム (ARN)、ボールトが作成された日、ボールト内に含まれているアーカイブの数、ボールト内のすべてのアーカイブの合計サイズなど、ボールトに関する情報が返されます。アーカイブの数と合計サイズは、Amazon S3 Glacier (S3 Glacier) が最後に生成されたボールトインベントリの時点のデータになります (Amazon S3 Glacier のボールトの操作を参照してください。)。S3 Glacier は、ほぼ毎日ボールトインベントリを生成します。つまり、ボールトのアーカイブを追加または削除し、すぐにボールトの説明リクエストを送信すると、レスポンスに変更が反映されない場合があります。

リクエスト

ボールトに関する情報を取得するには、特定のボールトリソースの URI に GET リクエストを送信します。

構文

```
GET /AccountId/vaults/VaultName HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: Length

{
  "CreationDate" : String,
  "LastInventoryDate" : String,
  "NumberOfArchives" : Number,
  "SizeInBytes" : Number,
  "VaultARN" : String,
  "VaultName" : String
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

レスポンス本文には次の JSON フィールドが含まれています。

CreationDate

ボールトが作成された日付（UTC）。

タイプ: ISO 8601 の日付形式の文字列表現。たとえば`2013-03-20T17:03:43.221Z`。

LastInventoryDate

S3 Glacier が最後にボールトインベントリを完了した日付 (UTC)。ボールトのインベントリの開始の詳細については、「ジョブの開始 (ジョブの POST)」を参照してください。

タイプ: ISO 8601 の日付形式の文字列表現。たとえば`2013-03-20T17:03:43.221Z`。

NumberOfArchives

最後にボールトインベントリが生成された時点のボールトのアーカイブ数。ボールトを作成したばかりのときなど、ボールトでインベントリが実行されていない場合、このフィールドには null が返されます。

タイプ: 数値

SizeInBytes

最後にインベントリを生成した日付における、アーカイブごとのオーバーヘッドを含む、ボールトのアーカイブの合計サイズ (バイト単位)。ボールトを作成したばかりのときなど、ボールトでインベントリが実行されていない場合、このフィールドには `null` が返されます。

タイプ: 数値

VaultARN

ボールトの Amazon リソースネーム (ARN)。

型: 文字列

VaultName

作成時に指定されたボールト名。ボールト名は、ボールトの ARN にも含まれます。

型: 文字列

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

以下の例は、examplevault というボールトに関する情報を取得する方法を示しています。

```
GET /-/vaults/examplevault HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
Content-Type: application/json
Content-Length: 260

{
  "CreationDate" : "2012-02-20T17:01:45.198Z",
  "LastInventoryDate" : "2012-03-20T17:03:43.221Z",
  "NumberOfArchives" : 192,
  "SizeInBytes" : 78088912,
  "VaultARN" : "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault",
  "VaultName" : "examplevault"
}
```

関連するセクション

- ボールトの作成 (PUT vault)
- ボールトのリスト (GET vaults)
- ボールトの削除 (DELETE vault)
- ジョブの開始 (ジョブの POST)
- Amazon S3 Glacier の ID とアクセス管理

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトアクセスポリシー (GET access-policy) の取得

説明

このオペレーションでは、ボールトの `access-policy` サブリソースセットを取得します。このサブリソースの設定の詳細については、「ボールトアクセスポリシー (PUT access-policy) の設定」を参照してください。ボールトにアクセスポリシーセットがない場合、オペレーションは `404 Not found` エラーを返します。ボールトアクセスポリシーの詳細については、「ボールトアクセスポリシー」を参照してください。

リクエスト

現在のボールトアクセスポリシーを取得するには、ボールトの `GET` サブリソースの URI に HTTP `access-policy` リクエストを送信します。

構文

```
GET /AccountId/vaults/vaultName/access-policy HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

レスポンスでは、Amazon S3 Glacier (S3 Glacier) はレスポンス本文で JSON 形式のボールトアクセスポリシーを返します。

構文

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length

{
  "Policy": "string"
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

レスポンス本文には次の JSON フィールドが含まれています。

Policy

JSON 文字列としてのボールトアクセスポリシー (エスケープ文字として "\" を使用) 。

タイプ: 文字列

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

以下の例では、ボールトアクセスポリシーの取得方法を示しています。

リクエストの例

この例では、ボールトの GET サブリソースの URI に access-policy リクエストを送信します。

```
GET /-/vaults/examplevault/access-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

リクエストが成功した場合、S3 Glacier はレスポンス本文で JSON 文字列としてボールトアクセスポリシーを返します。返される JSON 文字列では、「ボールトアクセスポリシー (PUT access-policy) の設定」の例に示すようにエスケープ文字として "\" を使用します。ただし、次の例では読みやすくするため、エスケープ文字を使用せずに、返される JSON 文字列を示します。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: length

{
  "Policy": "
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "allow-time-based-deletes",
          "Principal": {
            "AWS": "999999999999"
```

```
          },
          "Effect": "Allow",
          "Action": "glacier:Delete*",
          "Resource": [
            "arn:aws:glacier:us-west-2:999999999999:vaults/examplevault"
          ],
          "Condition": {
            "DateGreaterThan": {
              "aws:CurrentTime": "2018-12-31T00:00:00Z"
            }
          }
        }
      ]
    }
  "
}
```

関連するセクション

- ボールトアクセスポリシーの削除 (DELETE access-policy)
- ボールトアクセスポリシー (PUT access-policy) の設定

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトロックの取得 (ロックポリシーの GET)

説明

このオペレーションでは、指定したボールトの `lock-policy` サブリソースの以下の属性を取得します。

- ボールトに設定されているボールトロックポリシー。

- ボールトロックの状態。InProgess または Locked。
- ロック ID の有効期限。ロック ID は、ボールトロック処理を完了するために使用します。
- ボールトロックが開始されて InProgress 状態になった時刻。

ボールトロックは、InProgressを要求することによって ボールトロックの開始 (ロックポリシーの POST) 状態になります。ボールトロックは、Lockedを要求することによって ボールトロックの完了 (ロック ID の POST) 状態になります。ボールトロック処理は、ボールトロックの中止 (ロックポリシーの DELETE)を要求することによって中止できます。ボールトロック処理の詳細については、「S3 Glacier ボールトロック」を参照してください。

ボールトにボールトロックポリシーが設定されていない場合は、404 Not found エラーが返されます。ボールトロックポリシーの詳細については、「ボールトロックポリシー」を参照してください。

リクエスト

現在のボールトロックポリシーとその他の属性を取得するには、次の構文の例に示すように、ボールトの GET サブリソースの URI に HTTP lock-policy リクエストを送信します。

構文

```
GET /AccountId/vaults/vaultName/lock-policy HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

レスポンスでは、Amazon S3 Glacier (S3 Glacier) はレスポンス本文で JSON 形式のボールトアクセスポリシーを返します。

構文

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length

{
  "Policy": "string",
  "State": "string",
  "ExpirationDate": "string",
  "CreationDate":"string"
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

レスポンス本文には次の JSON フィールドが含まれています。

Policy

JSON 文字列としてのボールトロックポリシー (エスケープ文字として "\" を使用) 。

タイプ: 文字列

状態

ボールトロックの状態。

タイプ: 文字列

有効な値:`InProgress` |`Locked`

ExpirationDate

ロック ID の有効期限の日時 (UTC) 。この値は、ボールトロックが `null` 状態の場合は `Locked` になります。

タイプ: ISO 8601 の日付形式の文字列表現。たとえば`2013-03-20T17:03:43.221Z`。

CreationDate

ボールトロックが `InProgress` 状態になった日時 (UTC)。

タイプ: ISO 8601 の日付形式の文字列表現。たとえば`2013-03-20T17:03:43.221Z`。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

次の例は、ボールトロックポリシーの取得方法を示しています。

リクエストの例

この例では、ボールトの `GET` サブリソースの URI に `lock-policy` リクエストを送信します。

```
GET /-/vaults/examplevault/lock-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

リクエストが成功した場合、S3 Glacier はレスポンス本文で JSON 文字列としてボールトアクセスポリシーを返します。返される JSON 文字列では、「ボールトロックの開始 (ロックポリシーの POST)」のリクエストの例に示すようにエスケープ文字として "\" を使用します。ただし、次の例では読みやすくするため、エスケープ文字を使用せずに、返される JSON 文字列を示します。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: length

{
  "Policy": "
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "Define-vault-lock",
          "Principal": {
            "AWS": "arn:aws:iam::999999999999:root"
          },
          "Effect": "Deny",
          "Action": "glacier:DeleteArchive",
          "Resource": [
            "arn:aws:glacier:us-west-2:999999999999:vaults/examplevault"
          ],
          "Condition": {
            "NumericLessThanEquals": {
              "glacier:ArchiveAgeInDays": "365"
            }
          }
        }
      ]
    }
  ",
  "State": "InProgress",
  "ExpirationDate": "exampledate",
  "CreationDate": "exampledate"
}
```

関連するセクション

- ボールトロックの中止 (ロックポリシーの DELETE)
- ボールトロックの完了 (ロック ID の POST)
- ボールトロックの開始 (ロックポリシーの POST)

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールト通知の取得 (GET notification-configuration)

説明

このオペレーションでは、ボールトに設定した `notification-configuration` サブリソース (「ボールトの通知設定の指定 (PUT notification-configuration)」を参照) を取得します。ボールトに通知設定が設定されていない場合は、`404 Not Found` エラーが返されます。ボールト通知の詳細については、「Amazon S3 Glacier でのボールト通知の設定」を参照してください。

リクエスト

通知の設定情報を取得するには、ボールトの `GET` サブリソースの URI に `notification-configuration` リクエストを送信します。

構文

```
GET /AccountId/vaults/VaultName/notification-configuration HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length
{
  "Events": [
    String,
    ...
  ],
  "SNSTopic": String
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

JSON レスポンス本文には次の JSON フィールドが含まれています。

イベント

Amazon S3 Glacier (S3 Glacier) から指定した Amazon SNS トピックに通知が送信される 1 つ以上のイベントのリスト。通知を発行するようにボールトを設定できるボールトイベントの詳細については、「ボールトの通知設定の指定 (PUT notification-configuration)」を参照してください。

型: 配列

SNSTopic

Amazon Simple Notification Service (Amazon SNS) トピックの Amazon Resource Name (ARN)。詳細については、Amazon Simple Notification Service 使用開始 ガイドの「Amazon SNS の使用開始」を参照してください。

型: 文字列

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

以下の例は、ボールトの通知設定を取得する方法を示したものです。

リクエストの例

この例では、ボールトの GET サブリソースに `notification-configuration` リクエストを送信します。

```
GET /-/vaults/examplevault/notification-configuration HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

正常なレスポンスでは、レスポンス本文に JSON 形式で監査ログ作成設定のドキュメントが示されます。この例では、2 種類のイベント (`ArchiveRetrievalCompleted` および `InventoryRetrievalCompleted`) の通知が Amazon SNS トピック () に送信される設定を示しています。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 150

{
  "Events": [
    "ArchiveRetrievalCompleted",
    "InventoryRetrievalCompleted"
  ],
  "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic"
}
```

関連するセクション

- ボールト通知の削除 (通知設定の削除)
- ボールトの通知設定の指定 (PUT notification-configuration)
- Amazon S3 Glacier の ID とアクセス管理

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトロックの開始 (ロックポリシーの POST)

説明

このオペレーションでは、以下の手順を実行することによってボールトロック処理を開始します。

- 指定したボールトへのボールトロックポリシーのインストール
- ボールトロックの InProgress 状態への設定
- ボールトロック処理を完了するために使用するロック ID の取得

ボールトごとに 1 つのボールトロックポリシーを設定でき、ポリシーのサイズは最大 20 KB とすることができます。ボールトロックポリシーの詳細については、「ボールトロックポリシー」を参照してください。

ボールトロック処理は、ボールトロックが InProgress 状態になってから 24 時間以内に完了する必要があります。24 時間が過ぎると、ロック ID の有効期限が切れ、ボールトの InProgress 状態が自動的に終了し、ボールトロックポリシーがボールトから削除されます。ボールトロック処理を完了するには、ボールトロックの完了 (ロック ID の POST)を要求してボールトロックを Locked 状態にします。

> **Note**
>
> ボールトロックが Locked 状態になった後は、ボールトに対して新しいボールトロックを開始することはできません。

ボールトロック処理は、ボールトロックの中止 (ロックポリシーの DELETE)を要求することによって中止できます。ボールトロックの状態は、ボールトロックの取得 (ロックポリシーの GET)を要求することによって取得できます。ボールトロック処理の詳細については、「S3 Glacier ボールトロック」を参照してください。

ボールトロックが InProgress 状態のときにこのオペレーションを要求した場合は、AccessDeniedException エラーが返されます。ボールトロックが InProgress 状態の場合は、新しいボールトロックポリシーを開始する前にボールトロックの中止 (ロックポリシーの DELETE)を要求する必要があります。

リクエスト

ボールトロック処理を開始するには、次の構文の例に示すように、ボールトの POST サブリソースの URI に HTTP lock-policy リクエストを送信します。

構文

```
POST /AccountId/vaults/vaultName/lock-policy HTTP/1.1
```

```
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01

{
  "Policy": "string"
}
```

> **Note**
>
> AccountId 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

リクエストボディ には、次の JSON フィールドが含まれます。

Policy

JSON 文字列としてのボールトロックポリシー (エスケープ文字として "\" を使用) 。

タイプ: 文字列

必須: はい

レスポンス

Amazon S3 Glacier (S3 Glacier) はHTTP 201 Createdポリシーが受け入れられた場合、レスポンスを返します。

構文

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-lock-id: lockId
```

レスポンスヘッダー

成功したレスポンスには、すべての操作に共通のレスポンスヘッダーに加えて、次のレスポンスヘッダーが含まれます。共通のレスポンスヘッダーの詳細については、「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-lock-id	ボールトロック処理を完了するために使用するロック ID。 タイプ: 文字列

レスポンス本文

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

次の例では、ボールトの PUT サブリソースの URI に HTTP lock-policy リクエストを送信します。Policy JSON 文字列では、エスケープ文字として "\" を使用します。

```
PUT /-/vaults/examplevault/lock-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
Content-Length: length
x-amz-glacier-version: 2012-06-01

{"Policy":"{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":\"Define-vault-
lock\",\"Effect\":\"Deny\",\"Principal\":{\"AWS\":\"arn:aws:iam::999999999999:root
\"},\"Action\":\"glacier:DeleteArchive\",\"Resource\":\"arn:aws:glacier:us-
west-2:999999999999:vaults/examplevault\",\"Condition\":{\"NumericLessThanEquals\":
{\"glacier:ArchiveAgeinDays\":\"365\"}}}]}"}
```

レスポンスの例

リクエストが成功した場合、次の例に示すように、S3 Glacier は `HTTP 201 Created` レスポンスを返します。

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
x-amz-lock-id: AE863rKkWZU53SLW5be4DUcW
```

関連するセクション

- ボールトロックの中止 (ロックポリシーの DELETE)
- ボールトロックの完了 (ロック ID の POST)
- ボールトロックの取得 (ロックポリシーの GET)

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトのタグの一覧表示 (GET タグ)

このオペレーションでは、ボールトにアタッチされているすべてのタグを一覧表示します。タグがない場合は、空のマッピングが返されます。タグの詳細については、Amazon S3 Glacier リソースのタグ付けを参照してください。

リクエストの構文

ボールトのタグを一覧表示するには、次の構文例に示すように、タグの URI に HTTP GET リクエストを送信します。

```
GET /AccountId/vaults/vaultName/tags HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

オペレーションが成功した場合、サービスは HTTP レスポンス `200 OK` を返します。

レスポンスの構文

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: Length
{
   "Tags":
      {
         "string" : "string",
         "string" : "string"
      }
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

レスポンス本文には次の JSON フィールドが含まれています。

[タグ]

ボールトにアタッチされるタグ。各タグはキーと値で構成されます。

タイプ: 文字列から文字列へのマッピング

必須: はい

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

例: ボールトのタグの一覧表示

次の例では、ボールトのタグを一覧表示します。

リクエストの例

この例では、指定したボールトからタグのリストを取得するため、GET リクエストが送信されます。

```
GET /-/vaults/examplevault/tags HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

リクエストが成功した場合、Amazon S3 Glacier (S3 Glacier) は次の例に示すように、ボールトのタグのリストとともに `HTTP 200 OK` を返します。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
Content-Type: application/json
Content-Length: length

{
   "Tags",
      {
         "examplekey1": "examplevalue1",
         "examplekey2": "examplevalue2"
      }
}
```

関連するセクション

- ボールトにタグを追加する (POST タグの追加)
- ボールトからタグを削除する (POST タグの削除)

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトのリスト (GET vaults)

説明

このオペレーションでは、呼び出し元のユーザーのアカウントによって所有されているすべてのボールトのリストを表示します。レスポンスとして返されるリストは、ボールト名で ASCII コード順にソートされます。

このオペレーションでは、デフォルトで最大でリクエストあたり 10 項目返されます。リストで表示した以上にボールトが存在する場合は、レスポンス本文の `marker` フィールドに、新しいボールトのリストリクエストでリストの続きを表示できるように、ボールトの Amazon リソースネーム (ARN) が含まれています。そうでない場合は、`marker` フィールドは `null` です。次回のボールトリストのリクエストでは、前回のボールトリストのリクエストに対するレスポンスとして Amazon S3 Glacier (S3 Glacier) によって返された値を `marker` パラメータに設定します。また、リクエストで `limit` パラメータを指定して、レスポンスで返されるボールトの数を制限することもできます。

リクエスト

ボールトのリストを取得するには、`GET` リクエストをボールトのリソースに送信します。

構文

```
GET /AccountId/vaults HTTP/1.1
Host: glacier.Region.amazonaws.com
```

```
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

リクエストパラメータ

このオペレーションでは、次のリクエストパラメーターを使用します。

名前	説明	必須
limit	返されるボールトの最大数。デフォルトの上限は 10 です。返されるボールトの数は、指定された上限を下回ることはあっても、上限を上回ることはありません。 タイプ: 文字列 制約: 最小の整数値は 1 です。最大の整数値は 10 です。	いいえ
marker	ページ分割に使用する文字列。marker には、ボールトのリストの開始点となるボールトの ARN を指定します。(返されるリストには、marker に指定したボールトは含まれません)。marker 値は、前回のボールトのリストのレスポンスから取得します。前回のボールドのリストのリクエストで開始された結果のページ分割を継続する場合のみ、marker を指定する必要があります。マーカーに空の値 ("") を指定すると、最初のボールトから始まるボールトのリストが返されます。 タイプ: 文字列	いいえ

名前	説明	必須
	制約: なし	

リクエストヘッダー

このオペレーションは、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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: Length

{
  "Marker": String
  "VaultList": [
   {
    "CreationDate": String,
    "LastInventoryDate": String,
    "NumberOfArchives": Number,
    "SizeInBytes": Number,
    "VaultARN": String,
    "VaultName": String
   },
   ...
  ]
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

レスポンス本文には次の JSON フィールドが含まれています。

CreationDate

ボールトが作成された、協定世界時 (UTC) による日付。

型:文字列 たとえば、ISO 8601 の日付形式の文字列表現。`2013-03-20T17:03:43.221Z`。

LastInventoryDate

ボールトのインベントリが最後に生成された、協定世界時 (UTC) による日付。ボールトを作成したばかりのときなど、ボールトでインベントリが実行されていない場合、このフィールドの値は null です。ボールトのインベントリの開始の詳細については、「ジョブの開始 (ジョブの POST)」を参照してください。

タイプ: ISO 8601 の日付形式の文字列表現。たとえば`2013-03-20T17:03:43.221Z`。

Marker

結果のページ分割をどこから継続するかを示す `vaultARN`。リストに含まれるボールトをさらに取得するには、ボールトのリストの新規リクエストで `marker` を使用します。ボールトがそれ以上存在しない場合、この値は `null` です。

型: 文字列

NumberOfArchives

最後にインベントリを生成した日付のボールトのアーカイブ数。

タイプ: 数値

SizeInBytes

最後にインベントリを生成した日付における、アーカイブごとのオーバーヘッドを含む、ボールトのすべてのアーカイブの合計サイズ (バイト単位)。

タイプ: 数値

VaultARN

ボールトの Amazon リソースネーム (ARN)。

型: 文字列

VaultList

オブジェクトの配列。各オブジェクトにはボールトの説明が含まれます。

型: 配列

VaultName

ボールト名。

型: 文字列

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

例: すべてのボールトのリスト

以下はボールトのリストを表示する例です。リクエストでは、`marker` および `limit` パラメータが指定されていないため、最大 10 個のボールトが返されます。

リクエストの例

```
GET /-/vaults HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

`Marker` が `null` であり、リストに表示するボールトがこれ以上存在しないことを示しています。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

```
Content-Type: application/json
Content-Length: 497

{
  "Marker": null,
  "VaultList": [
   {
    "CreationDate": "2012-03-16T22:22:47.214Z",
    "LastInventoryDate": "2012-03-21T22:06:51.218Z",
    "NumberOfArchives": 2,
    "SizeInBytes": 12334,
    "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault1",
    "VaultName": "examplevault1"
   },
   {
    "CreationDate": "2012-03-19T22:06:51.218Z",
    "LastInventoryDate": "2012-03-21T22:06:51.218Z",
    "NumberOfArchives": 0,
    "SizeInBytes": 0,
    "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault2",
    "VaultName": "examplevault2"
   },
   {
    "CreationDate": "2012-03-19T22:06:51.218Z",
    "LastInventoryDate": "2012-03-25T12:14:31.121Z",
    "NumberOfArchives": 0,
    "SizeInBytes": 0,
    "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault3",
    "VaultName": "examplevault3"
   }
  ]
}
```

例: ボールトの部分的なリスト

以下は、marker に指定したボールトから開始して 2 個のボールトを返す例です。

リクエストの例

```
GET /-/vaults?limit=2&marker=arn:aws:glacier:us-west-2:012345678901:vaults/
examplevault1 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
```

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

2 個のボールトがリストに返されます。次回のボールトのリストのリクエストでページ分割を継続できるように、`Marker` にボールト ARN が含まれています。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
Content-Type: application/json
Content-Length: 497

{
  "Marker": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault3",
  "VaultList": [
   {
    "CreationDate": "2012-03-16T22:22:47.214Z",
    "LastInventoryDate": "2012-03-21T22:06:51.218Z",
    "NumberOfArchives": 2,
    "SizeInBytes": 12334,
    "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault1",
    "VaultName": "examplevault1"
   },
   {
    "CreationDate": "2012-03-19T22:06:51.218Z",
    "LastInventoryDate": "2012-03-21T22:06:51.218Z",
    "NumberOfArchives": 0,
    "SizeInBytes": 0,
    "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault2",
    "VaultName": "examplevault2"
   }
  ]
}
```

関連するセクション

- ボールトの作成 (PUT vault)
- ボールトの削除 (DELETE vault)

- ジョブの開始 (ジョブの POST)
- Amazon S3 Glacier の ID とアクセス管理

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトからタグを削除する (POST タグの削除)

このオペレーションでは、ボールトにアタッチされたタグのセットから 1 つ以上のタグを削除します。タグの詳細については、Amazon S3 Glacier リソースのタグ付けを参照してください。

このオペレーションはべき等です。オペレーションは、ボールトにアタッチされたタグがない場合でも成功します。

リクエストの構文

ボールトからタグを削除するには、次の構文例に示すように、タグの URI に HTTP POST リクエストを送信します。

```
POST /AccountId/vaults/vaultName/tags?operation=remove HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01
{
   "TagKeys": [
      "string",
      "string"
   ]
}
```

> **Note**
>
> `AccountId` 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

リクエストパラメータ

名前	説明	必須
`operation=remove`	値 `operation` を持つ 1 つのクエリ文字パラメータ `remove` が、ボールトにタグを追加する (POST タグの追加) からこれを区別します。	はい

リクエストヘッダー

このオペレーションは、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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

リクエストボディ には、次のJSONフィールドが含まれます。

TagKeys

タグキーのリスト。対応する各タグがボールトから削除されます。

タイプ: 文字列の配列

長さの制限: リストに 1 つ以上の項目があること。リストの項目は最大 10 個。

必須: はい

レスポンス

アクションが成功した場合、サービスは空の HTTP 本文を持つ HTTP 204 No Content 応答を送信します。

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

次の例では、指定したタグを削除する HTTP POST リクエストを送信します。

```
POST /-/vaults/examplevault/tags?operation=remove HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
Content-Length: length
x-amz-glacier-version: 2012-06-01

{
```

```
   "TagsKeys": [
      "examplekey1",
      "examplekey2"
   ]
}
```

レスポンスの例

リクエストが成功した場合、次の例に示しているように、Amazon S3 Glacier (S3 Glacier) は HTTP `204 No Content` を返します。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

関連するセクション

- ボールトにタグを追加する (POST タグの追加)
- ボールトのタグの一覧表示 (GET タグ)

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトアクセスポリシー (PUT access-policy) の設定

説明

このオペレーションでは、ボールト用のアクセスポリシーを設定し、既存のポリシーを上書きします。ボールトアクセスポリシーを設定するには、ボールトの PUT サブリソースに `access-policy` リクエストを送信します。ボールトごとに 1 つのアクセスポリシーを設定でき、ポリシーのサイズは最大 20 KB とすることができます。ボールトアクセスポリシーの詳細については、「ボールトアクセスポリシー」を参照してください。

リクエスト

構文

ボールトアクセスポリシーを設定するには、次の構文の例に示すように、ボールトの access-policy サブリソースの URI に HTTP PUT リクエストを送信します。

```
PUT /AccountId/vaults/vaultName/access-policy HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01

{
  "Policy": "string"
}
```

> **Note**
>
> AccountId 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

リクエストボディ には、次の JSON フィールドが含まれます。

Policy

JSON 文字列としてのボールトアクセスポリシー (エスケープ文字として "\" を使用) 。

タイプ: 文字列

必須: はい

レスポンス

レスポンスでは、ポリシーが受け入れられた場合、S3 Glacier によって `204 No Content` が返されます。

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

次の例では、ボールトの PUT サブリソースの URI に HTTP `access-policy` リクエストを送信します。`Policy` JSON 文字列では、エスケープ文字として "\" を使用します。

```
PUT /-/vaults/examplevault/access-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

```
Content-Length: length
x-amz-glacier-version: 2012-06-01

{"Policy":"{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":\"Define-owner-access-
rights\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"arn:aws:iam::999999999999:root
\"},\"Action\":\"glacier:DeleteArchive\",\"Resource\":\"arn:aws:glacier:us-
west-2:999999999999:vaults/examplevault\"}]}"}
```

レスポンスの例

リクエストが成功した場合、次の例に示しているように、Amazon S3 Glacier (S3 Glacier) は HTTP `204 No Content` を返します。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

関連するセクション

- ボールトアクセスポリシーの削除 (DELETE access-policy)
- ボールトアクセスポリシー (GET access-policy) の取得

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

ボールトの通知設定の指定 (PUT notification-configuration)

説明

Amazon S3 Glacier (S3 Glacier) では、アーカイブおよびボールトインベントリの取得は非同期オペレーションです。そのため、ジョブ出力をダウンロードするには、まずジョブを開始し、ジョブが完了するまで待機する必要があります。そのため、こうしたジョブが完了したときに Amazon Simple

Notification Service (Amazon SNS) トピックにメッセージを投稿するようにボールトを設定できます。このオペレーションを使用すると、ボールトに通知設定を指定できます。詳細については、「Amazon S3 Glacier でのボールト通知の設定」を参照してください。

ボールトの通知を設定するには、ボールトの `notification-configuration` サブリソースに PUT リクエストを送信します。通知設定はボールトに固有です。そのため、ボールトサブリソースとも呼ばれます。リクエストには、Amazon Simple Notification Service (Amazon SNS)トピックと S3 Glacier がトピックに通知を送信するイベントを指定した JSON ドキュメントを含める必要があります。

次のボールトイベントに対して通知を発行するようにボールトを設定できます。

- **`ArchiveRetrievalCompleted`**-このイベントは、アーカイブを取得するために開始されたジョブ (ジョブの開始 (ジョブの POST)) が完了したときに発生します。完了したジョブのステータスは、`Succeeded` または `Failed` になります。SNS トピックに送信される通知は、ジョブの説明 (GET JobID) から返される出力と同じ出力です。
- **`InventoryRetrievalCompleted`**- このイベントは、インベントリを取得するために開始されたジョブ (ジョブの開始 (ジョブの POST)) が完了したときに発生します。完了したジョブのステータスは、`Succeeded` または `Failed` になります。SNS トピックに送信される通知は、ジョブの説明 (GET JobID) から返される出力と同じ出力です。

Amazon SNS トピックでは、トピックに通知を発行できるようにボールトにアクセス権限を付与する必要があります。

リクエスト

ボールトに通知設定を指定するには、ボールトの `notification-configuration` サブリソースの URI に PUT リクエストを送信します。リクエストボディに設定を指定します。設定には、Amazon SNS トピック名と、各トピックへの通知をトリガーするイベントの配列を含めます。

構文

```
PUT /AccountId/vaults/VaultName/notification-configuration HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01

{
```

```
    "SNSTopic": String,
    "Events":[String, ...]
}
```

> **Note**
>
> `AccountId` 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

リクエスト本文の JSON には、次のフィールドが含まれます。

イベント

S3 Glacier が通知を送信する 1 つ以上のイベントの配列。

有効な値: `ArchiveRetrievalCompleted` | `InventoryRetrievalCompleted`

必須: はい

型: 配列

SNSTopic

Amazon SNS トピック ARN。詳細については、Amazon Simple Notification Service 使用開始 ガイドの「Amazon SNS の使用開始」を参照してください。

必須: はい

型: 文字列

レスポンス

レスポンスでは、通知設定が承認された場合、Amazon S3 Glacier (S3 Glacier) によって `204 No Content` が返されます。

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

このオペレーションは、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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

レスポンス本文

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

以下の例は、ボールトの通知を設定する方法を示したものです。

リクエストの例

次のリクエストでは、`examplevault` の通知が設定されるため、2 つのイベント (`ArchiveRetrievalCompleted` と `InventoryRetrievalCompleted`) に対する通知が Amazon SNS トピック に送信されます。

```
PUT /-/vaults/examplevault/notification-policy HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
```

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
   "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"],
   "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic"
}
```

レスポンスの例

正常なレスポンスでは、`204 No Content` が返されます。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

関連するセクション

- ボールト通知の取得 (GET notification-configuration)
- ボールト通知の削除 (通知設定の削除)
- Amazon S3 Glacier の ID とアクセス管理

以下の資料も参照してください。

言語固有の Amazon SDK のいずれかでこの API を使用する方法の詳細については、次を参照してください。

- AWS Command Line Interface

アーカイブオペレーション

以下は、S3 Glacier で使用できるアーカイブ オペレーションです。

トピック

- アーカイブの削除 (DELETE archive)
- アーカイブのアップロード (POST archive)

アーカイブの削除 (DELETE archive)

説明

このオペレーションは、ボールトからアーカイブを削除します。ボールトから削除できるアーカイブは一度に 1 つです。アーカイブを削除するには、削除リクエストにアーカイブ ID を指定する必要があります。アーカイブ ID は、そのアーカイブを含むボールトのボールトインベントリをダウンロードすることで取得できます。ボールトインベントリのダウンロードの詳細については、「Amazon S3 Glacier でボールトインベントリをダウンロードする」を参照してください。

アーカイブを削除した後でも、削除したアーカイブの取得ジョブを開始することはリクエストできますが、アーカイブの取得ジョブ自体は失敗します。

アーカイブを削除する際に、該当するアーカイブ ID のアーカイブが取得中であった場合、取得は以下のシナリオに応じて成功する場合と成功しない場合があります。

- Amazon S3 Glacier (S3 Glacier) がアーカイブの削除リクエストを受け取ったときに、アーカイブの取得ジョブがダウンロード用のデータを準備している最中であった場合は、アーカイブの取得オペレーションが失敗することがあります。
- S3 Glacier がアーカイブの削除リクエストを受け取ったときに、アーカイブの取得ジョブがダウンロード対象のアーカイブの準備を完了していた場合は、出力をダウンロードできます。

アーカイブの取得に関する詳細については、「S3 Glacier でのアーカイブのダウンロード」を参照してください。

このオペレーションはべき等です。既に削除されたアーカイブを削除しようとした場合には、エラーは発生しません。

リクエスト

アーカイブを削除するには、アーカイブのリソース URI に DELETE リクエストを送信します。

構文

```
DELETE /AccountId/vaults/VaultName/archives/ArchiveID HTTP/1.1
Host: glacier.Region.amazonaws.com
x-amz-Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

以下の例は、examplevault というボールトをアーカイブから削除する方法を示しています。

リクエストの例

削除されるアーカイブの ID は、archives のサブリソースとして指定します。

```
DELETE /-/vaults/examplevault/archives/NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

リクエストが成功した場合、S3 Glacier は 204 No Content で応答し、アーカイブが削除されたことを示します。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

関連するセクション

- マルチパートアップロードの開始 (POST multipart-uploads)
- アーカイブのアップロード (POST archive)
- Amazon S3 Glacier の ID とアクセス管理

アーカイブのアップロード (POST archive)

説明

このオペレーションでは、ボールトにアーカイブを追加します。正常にアップロードされると、データは永続的に保持されます。Amazon S3 Glacier (S3 Glacier) により、x-amz-archive-idレスポ

ンスのヘッダーにアーカイブ IDが返されます。後でアーカイブにアクセスできるように、返されたアーカイブ ID を保管しておく必要があります。

アップロードするデータの SHA256 木構造ハッシュを指定する必要があります。SHA256 木構造ハッシュの計算の詳細については、「チェックサムの計算」を参照してください。

> **Note**
>
> SHA256 木構造ハッシュは、API を使用する場合のアーカイブのアップロード (POST アーカイブ) アクションにのみ必要です。を使用する場合は必要ありません AWS CLI。

アーカイブのアップロード時に、オプションでアーカイブの説明を最大 1,024 文字の印刷可能な ASCII 文字で指定できます。S3 Glacier は、アーカイブを取得するか、ボールトインベントリを取得するときに、アーカイブの説明を返します。S3 Glacier はその説明を一切解釈しません。アーカイブの説明は、一意である必要はありません。説明を使用して、アーカイブのリストを取得することや、ソートすることはできません。

オプションのアーカイブの説明以外に、S3 Glacier ではアーカイブに対してどのような追加のメタデータもサポートしていません。アーカイブ ID は、アーカイブに関するどのような情報も推察することができないように、意味のない文字列になっています。そのため、クライアント側でアーカイブに関するメタデータを管理することもできます。詳細については、「Amazon S3 Glacier でのアーカイブの操作」を参照してください。

アーカイブは変更不可能です。アーカイブをアップロードした後で、アーカイブやアーカイブの説明を編集することはできません。

リクエスト

アーカイブをアップロードするには、HTTP POST メソッドを使用し、アーカイブの保存先となるボールトの `archives` サブリソースをリクエストの範囲として指定します。リクエストには、アーカイブのペイロードサイズ、チェックサム (SHA256 木構造ハッシュ) を含める必要があります。オプションでアーカイブの説明を含めることができます。

構文

```
POST /AccountId/vaults/VaultName/archives
Host: glacier.Region.amazonaws.com
x-amz-glacier-version: 2012-06-01
Date: Date
Authorization: SignatureValue
```

```
x-amz-archive-description: Description
x-amz-sha256-tree-hash: SHA256 tree hash
x-amz-content-sha256: SHA256 linear hash
Content-Length: Length

<Request body.>
```

> **Note**
>
> AccountId 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

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

この操作では、すべての操作で共通のリクエストヘッダーに加えて、次のリクエストヘッダーを使用します。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

名前	説明	必須
Content-Length	オブジェクトのサイズ (バイト単位) 。詳細については、http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.13 を参照してください。 タイプ: 数値 デフォルト: なし 制約: なし	はい
x-amz-archive-description		いいえ

名前	説明	必須
	アップロードするアーカイブのオプションの説明。わかりやすい説明や、割り当てる何らかの識別子を自分で選択して指定できます。説明は、すべてのアーカイブで一意である必要はありません。ボールトインベントリを取得した場合 (「ジョブの開始 (ジョブの POST)」を参照)、レスポンスで返されるアーカイブごとにこの説明が含まれます。 タイプ: 文字列 デフォルト: なし 制約: 説明は 1,024 文字以下である必要があります。使用可能な文字は、制御コードを除く 7 ビット ASCII コードです。具体的には、32—126 (10 進) または 0x20—0x7E (16 進) の ASCII 値です。	
`x-amz-content-sha256`	ペイロードの SHA256 チェックサム (線形ハッシュ)。この値は、`x-amz-sha256-tree-hash` ヘッダーで指定する値とは異なります。 タイプ: 文字列 デフォルト: なし 制約: なし	はい
`x-amz-sha256-tree-hash`	ユーザーが計算したペイロードのチェックサム (SHA256 木構造ハッシュ) 。SHA256 木構造ハッシュの計算については、「チェックサムの計算」を参照してください。S3 Glacier によって計算されたペイロードのチェックサムが異なる場合、リクエストは拒否されます。 タイプ: 文字列 デフォルト: なし 制約: なし	はい

リクエスト本文

リクエストボディには、アップロードするデータを含めます。

レスポンス

レスポンスでは、S3 Glacier により、アーカイブが永続的に保存され、アーカイブ ID の URI パスが返されます。

構文

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-sha256-tree-hash: ChecksumComputedByAmazonGlacier
Location: Location
x-amz-archive-id: ArchiveId
```

レスポンスヘッダー

成功したレスポンスには、すべての操作に共通のレスポンスヘッダーに加えて、次のレスポンスヘッダーが含まれます。共通のレスポンスヘッダーの詳細については、「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
`Location`	新しく追加されたアーカイブリソースの相対 URI パス。 タイプ: 文字列
`x-amz-archive-id`	アーカイブの ID。この値も `Location` ヘッダーの一部として含まれます。 タイプ: 文字列
`x-amz-sha256-tree-hash`	によって計算されたアーカイブのチェックサム。 タイプ: 文字列

レスポンス本文

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

次の例に、アーカイブをアップロードするリクエストを示します。

```
POST /-/vaults/examplevault/archives HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-sha256-tree-hash:
 beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60
x-amz-content-sha256: 7f2fe580edb35154041fa3d4b41dd6d3adaef0c85d2ff6309f1d4b520eeecda3
Content-Length: 2097152
x-amz-glacier-version: 2012-06-01
Authorization: Authorization=AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-
amz-glacier-
version,Signature=16b9a9e220a37e32f2e7be196b4ebb87120ca7974038210199ac5982e792cace

<Request body (2097152 bytes).>
```

レスポンスの例

以下の正常なレスポンスには、`Location` ヘッダーが含まれており、S3 Glacier がアーカイブに割り当てた ID を取得できます。

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
x-amz-sha256-tree-hash:
 beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60
Location: /111122223333/vaults/examplevault/archives/
NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
```

```
x-amz-archive-id: NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
```

関連するセクション

- Amazon S3 Glacier でのアーカイブの操作
- パート単位での大きなアーカイブのアップロード (マルチパートアップロード)
- アーカイブの削除 (DELETE archive)
- Amazon S3 Glacier の ID とアクセス管理

マルチパートアップロードオペレーション

以下は、S3 Glacier で使用できるマルチパートアップロードオペレーションです。

トピック

- マルチパートアップロードの中止 (DELETE uploadID)
- マルチパートアップロードの完了 (POST uploadID)
- マルチパートアップロードの開始 (POST multipart-uploads)
- パートのリスト (GET uploadID)
- マルチパートアップロードのリスト (GET multipart-uploads)
- パートのアップロード (PUT uploadID)

マルチパートアップロードの中止 (DELETE uploadID)

説明

このマルチパートアップロードのオペレーションのコマンドは、アップロード ID によって識別されるマルチパートアップロードを中止します。

マルチパートアップロードの中止リクエストが成功すると、そのアップロード ID を使用して、それ以上パートをアップロードすることや、その他のオペレーションを実行することができなくなります。完了済みのマルチパートアップロードを中止した場合は、失敗します。ただし、既に中止したアップロードを中止した場合は、少しの間成功します。

このオペレーションはべき等です。

マルチパートアップロードの詳細については、「パート単位での大きなアーカイブのアップロード (マルチパートアップロード)」を参照してください。

リクエスト

マルチパートアップロードを中止するには、ボールトの DELETE サブリソースの URI に HTTP multipart-uploads リクエストを送信し、URI の一部として特定のマルチパートアップロード ID を指定します。

構文

```
DELETE /AccountId/vaults/VaultName/multipart-uploads/uploadID HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

以下に、マルチパートアップロード ID リソースの URI に DELETE リクエストを送信する例を示します。

```
DELETE /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE  HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

```
HTTP/1.1 204 No Content
```

```
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

関連するセクション

- マルチパートアップロードの開始 (POST multipart-uploads)
- パートのアップロード (PUT uploadID)
- マルチパートアップロードの完了 (POST uploadID)
- マルチパートアップロードのリスト (GET multipart-uploads)
- パートのリスト (GET uploadID)
- パート単位での大きなアーカイブのアップロード (マルチパートアップロード)
- Amazon S3 Glacier の ID とアクセス管理

マルチパートアップロードの完了 (POST uploadID)

説明

アーカイブのすべてのパーツがアップロードされ、S3 Glacier がアップロードされたパーツからアーカイブをアセンブルする準備ができたことを Amazon S3 Glacier (S3 Glacier) に通知するには、このマルチパートアップロードオペレーションを呼び出します。

マルチパートアップロードの詳細については、「パート単位での大きなアーカイブのアップロード (マルチパートアップロード)」を参照してください。

S3 Glacier では、アーカイブをアセンブルして、ボールトに保存した後、新しく作成されたアーカイブリソースのアーカイブ ID を返します。アーカイブをアップロードしたら、後でアーカイブを取得するために、返されたアーカイブ ID を保管しておく必要があります。

アップロードしたアーカイブ全体の SHA256 木構造ハッシュを計算してリクエストに含める必要があります。SHA256 木構造ハッシュの計算の詳細については、「チェックサムの計算」を参照してください。サーバー側でも、S3 Glacier によりアセンブルされたアーカイブの SHA256 木構造ハッシュが作成されます。値が一致した場合は、S3 Glacier によってボールトにアーカイブが保存されます。値が一致しない場合は、エラーが返され、オペレーションは失敗します。パートのリスト (GET uploadID) オペレーションでは、特定のマルチパートアップロードでアップロードされたパートのリストが返されます。そのリストには、アップロード済みの各パートのチェックサム情報が含まれおり、チェックサムの誤りをデバッグするために使用することができます。

さらに、S3 Glacier は不足しているコンテンツ範囲がないかどうかを確認します。パートをアップロードするときに、最終的にアセンブルされたアーカイブ内でのパートの位置を特定できるように範囲の値を指定します。最終的にアーカイブをアセンブルする際に、S3 Glacier は不足しているコンテンツ範囲がないかどうかを確認します。不足しているコンテンツ範囲がある場合、S3 Glacier はエラーを返し、マルチパートアップロードの完了オペレーションは失敗します。

マルチパートアップロードの完了オペレーションはべき等です。マルチパートアップロードが初めて正常に完了してから短期間内にこのオペレーションを再度呼び出した場合、オペレーションは成功し、同じアーカイブ ID が返されます。これは、ネットワークの問題が発生して、接続が中断されたり、500 サーバーエラーを受け取ったりした場合に役に立ちます。このような問題が起きた場合に、マルチパートアップロードの完了リクエストを再度実行し、アーカイブが重複して作成されることなく、同じアーカイブ ID を取得できます。ただし、マルチパートアップロードの完了後、パートのリストオペレーションを呼び出すことはできません。また、マルチパートアップロードは完了オペレーションがべき等であっても、マルチパートアップロードのリストのレスポンスには表示されません。

リクエスト

マルチパートアップロードを完了するには、マルチパートアップロードの開始リクエストに対するレスポンスとして S3 Glacier によって作成されたアップロード ID の URI に HTTP POST リクエストを送信します。この URI は、パートをアップロードするときに使用した URI と同じです。必須の共通のヘッダーに加えて、アーカイブ全体の SHA256 木構造ハッシュの結果およびアーカイブの合計サイズ (バイト単位) を含める必要があります。

構文

```
POST /AccountId/vaults/VaultName/multipart-uploads/uploadID
Host: glacier.Region.amazonaws.com
Date: date
Authorization: SignatureValue
x-amz-sha256-tree-hash: SHA256 tree hash of the archive
x-amz-archive-size: ArchiveSize in bytes
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3

Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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

この操作では、すべての操作で共通のリクエストヘッダーに加えて、次のリクエストヘッダーを使用します。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

名前	説明	必須
`x-amz-archive-size`	アーカイブ全体の合計サイズ (バイト単位)。この値には、アップロードした個々のパートのすべてのサイズの合計値を指定する必要があります。 タイプ: 文字列 デフォルト: なし 制約: なし	はい
`x-amz-sha256-tree-hash`	アーカイブ全体の SHA256 木構造ハッシュ。個々のパートの SHA256 木構造ハッシュの木構造ハッシュです。リクエストに指定した値が、S3 Glacier によって計算された、最終的にアセンブルされたアーカイブの SHA256 木構造ハッシュと一致しない場合、S3 Glacier はエラーを返し、リクエストは失敗します。 タイプ: 文字列 デフォルト: なし 制約: なし	はい

リクエストの要素

このオペレーションではリクエストの要素を使用しません。

レスポンス

Amazon S3 Glacier (S3 Glacier) により、アーカイブ全体の SHA256 木構造ハッシュが作成されます。値が、ユーザーがリクエストで指定したアーカイブ全体の SHA256 木構造ハッシュと一致する場合、S3 Glacier はそのアーカイブをボールトに追加します。レスポンスでは、新しく追加されたアーカイブリソースの URL パスを含む HTTP `Location` ヘッダーが返されます。リクエストで送信したアーカイブのサイズまたは SHA256 木構造ハッシュが一致しない場合、S3 Glacier によってエラーが返され、アップロードは未完了の状態のままになります。後で正確な値を使用してマルチパートアップロードの完了オペレーションを再試行できます。再試行により、アーカイブを正常に作成することができます。マルチパートアップロードが完了しない場合は、最終的に S3 Glacier はアップロード ID を再利用します。

構文

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
x-amz-archive-id: ArchiveId
```

レスポンスヘッダー

成功したレスポンスには、すべての操作に共通のレスポンスヘッダーに加えて、次のレスポンスヘッダーが含まれます。共通のレスポンスヘッダーの詳細については、「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
`Location`	新しく作成したアーカイブの相対 URI パス。この URL には、S3 Glacier によって生成されたアーカイブ ID が含まれます。 タイプ: 文字列
`x-amz-archive-id`	アーカイブの ID。この値も `Location` ヘッダーの一部として含まれます。

名前	説明
	タイプ: 文字列

レスポンスフィールド

このオペレーションはレスポンス本文を返しません。

例

リクエストの例

この例では、マルチパートアップロードの開始リクエストで返された URI に HTTP POST リクエストを送信します。リクエストでは、アーカイブ全体の SHA256 木構造ハッシュとアーカイブの合計サイズの両方を指定します。

```
POST /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
z-amz-Date: 20170210T120000Z
x-amz-sha256-tree-hash:1ffc0f54dd5fdd66b62da70d25edacd0
x-amz-archive-size:8388608
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

以下のレスポンスの例は、アップロードしたパートから S3 Glacier によってアーカイブが正常に作成されたことを示しています。レスポンスには、アーカイブ ID と完全なパスが含まれています。

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Location: /111122223333/vaults/examplevault/archives/
NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
x-amz-archive-id: NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
```

新しく追加されたリソース/アーカイブの URI に HTTP リクエストを送信できます。たとえば、アーカイブを取得するために GET リクエストを送信できます。

関連するセクション

- マルチパートアップロードの開始 (POST multipart-uploads)
- パートのアップロード (PUT uploadID)
- マルチパートアップロードの中止 (DELETE uploadID)
- マルチパートアップロードのリスト (GET multipart-uploads)
- パートのリスト (GET uploadID)
- パート単位での大きなアーカイブのアップロード (マルチパートアップロード)
- アーカイブの削除 (DELETE archive)
- Amazon S3 Glacier の ID とアクセス管理

マルチパートアップロードの開始 (POST multipart-uploads)

説明

この操作では、マルチパートアップロードを開始します (パート単位での大きなアーカイブのアップロード (マルチパートアップロード)を参照してください。)。Amazon S3 Glacier (S3 Glacier) により、マルチパートアップロードリソースが作成され、レスポンスでその ID が返されます。後続のマルチパートアップロードオペレーションでは、このアップロード ID を使用します。

マルチパートアップロードを開始する際に、パートサイズをバイト単位で指定します。パートサイズは、メビバイト (MiB) (1024 キビバイト [KiB]) に 2 の累乗を乗じた値であることが必要です。例えば、1048576 (1 MiB)、2097152 (2 MiB)、4194304 (4 MiB)、8388608 (8 MiB) などです。許容される最小のパートサイズは 1 MiB で、最大は 4 ギビバイト (GiB) です。

このアップロード ID を使用してアップロードするパートは、最後のパートを除き、すべて同じサイズになります。最後のパートは、同じサイズ以下になります。たとえば、16.2 MiB のファイルをアップロードするとします。4 MiB のパートサイズでマルチパートアップロードを開始した場合、各 4 MiB のパートが 4 つと 0.2 MiB のパートが 1 つアップロードされます。

> **Note**
>
> S3 Glacier ではアーカイブ全体のサイズを指定する必要がないため、マルチパートアップロードを開始する際にアーカイブのサイズを把握している必要はありません。

マルチパートアップロードが完了すると、S3 Glacier は ID によって参照されるマルチパートアップロードリソースを削除します。マルチパートアップロードをキャンセルした場合も、S3 Glacier により、マルチパートアップロードリソースが削除されます。24 時間アクティビティがない場合も削除されることがあります。ID は 24 時間後も引き続き使用できる場合がありますが、アプリケーションでこの動作を想定しないでください。

リクエスト

マルチパートアップロードを開始するには、アーカイブの保存先となるボールトの POST サブリソースの URI に HTTP `multipart-uploads` リクエストを送信します。リクエストにはパートサイズを含める必要があります。また、オプションでアーカイブの説明を含めることができます。

構文

```
POST /AccountId/vaults/VaultName/multipart-uploads
Host: glacier.us-west-2.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
x-amz-archive-description: ArchiveDescription
x-amz-part-size: PartSize
```

> **Note**
>
> `AccountId` 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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

この操作では、すべての操作で共通のリクエストヘッダーに加えて、次のリクエストヘッダーを使用します。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

名前	説明	必須
`x-amz-part-size`	最後を除く各パートのサイズ (バイト単位)。最後のパートはこのパートサイズより小さくなる場合があります。 タイプ: 文字列 デフォルト: なし 制約: パートサイズは、メビバイト (1024 KiB) に 2 の累乗を乗じた値であることが必要です。例えば、1048576 (1 MiB)、2097152 (2 MiB)、4194304 (4 MiB)、8388608 (8 MiB) などです。許容される最小のパートサイズは 1 MiB で、最大は 4 GiB (4096 MiB) です。	はい
`x-amz-archive-description`	パート単位でアップロードするアーカイブの説明。わかりやすい説明や、割り当てる何らかの一意の識別子を自分で選択して指定できます。ボールトインベントリを取得した場合 (「ジョブの開始 (ジョブの POST)」を参照)、レスポンスに返されるアーカイブごとにこの説明がインベントリに含まれます。アーカイブの説明の先頭の空白は削除されます。 タイプ: 文字列 デフォルト: なし 制約事項: 説明は 1024 バイト以下である必要があります。使用可能な文字は、制御コード	いいえ

名前	説明	必須
	を除く 7 ビット ASCII コードです。具体的には、32 ~ 126 (10 進) または 0x20 ~ 0x7E (16 進) の ASCII 値です。	

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

レスポンスでは、S3 Glacier により、ID によって識別されたマルチパートアップロードリソースが作成され、マルチパートアップロード ID の相対 URI パスが返されます。

構文

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
x-amz-multipart-upload-id: multiPartUploadId
```

レスポンスヘッダー

成功したレスポンスには、すべての操作に共通のレスポンスヘッダーに加えて、次のレスポンスヘッダーが含まれます。共通のレスポンスヘッダーの詳細については、「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
`Location`	S3 Glacier によって作成されたマルチパートアップロード ID の相対 URI パス。この URI パス を使用して、パートをアップロードするリクエストの範囲を指定し、マルチパートアップロードを完了します。 タイプ: 文字列

名前	説明
x-amz-multipart-upload-id	マルチパートアップロードの ID。この値も Location ヘッダーの一部として含まれます。 タイプ: 文字列

レスポンス本文

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

次の例では、POST というボールトの multipart-uploads サブリソースの URI に HTTP examplevault リクエストを送信して、マルチパートアップロードを開始します。リクエストには、4 MiB (4194304 バイト) のパートサイズとオプションのアーカイブの説明を指定するヘッダーが含まれています。

```
POST /-/vaults/examplevault/multipart-uploads
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-archive-description: MyArchive-101
x-amz-part-size: 4194304
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

S3 Glacier により、マルチパートアップロードリソースが作成され、ボールトの multipart-uploads サブリソースにそのリソースが追加されます。Location レスポンスヘッダーには、マルチパートアップロード ID の相対 URI パスが含まれています。

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Location: /111122223333/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE
x-amz-multipart-upload-id:
 OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE
```

個々のパートのアップロードについては、「パートのアップロード (PUT uploadID)」を参照してください。

関連するセクション

- パートのアップロード (PUT uploadID)
- マルチパートアップロードの完了 (POST uploadID)
- マルチパートアップロードの中止 (DELETE uploadID)
- マルチパートアップロードのリスト (GET multipart-uploads)
- パートのリスト (GET uploadID)
- アーカイブの削除 (DELETE archive)
- パート単位での大きなアーカイブのアップロード (マルチパートアップロード)
- Amazon S3 Glacier の ID とアクセス管理

パートのリスト (GET uploadID)

説明

このマルチパートアップロードオペレーションは、アップロード ID によって識別される特定のマルチパートアップロードでアップロードしたアーカイブのパートをリスト表示します。マルチパートアップロードの詳細については、「パート単位での大きなアーカイブのアップロード (マルチパートアップロード)」を参照してください。

このリクエストは、マルチパートアップロードの完了前であれば、マルチパートアップロードの進行中にいつでも実行できます。S3 Glacier は、各パートのアップロードで指定した範囲でソートされたパート表を返します。マルチパートアップロードの完了後にパートのリストのリクエストを送信すると、Amazon S3 Glacier (S3 Glacier) はエラーを返します。

パートのリストオペレーションはページ分割をサポートしています。レスポンス本文の `Marker` フィールドに、リストの続きを表示するためのマーカーが含まれているかどうかを常に確認する必要があります。項目がそれ以上存在しない場合、`marker` フィールドは `null` です。`marker` が null でない場合にパートの次のセットを取得するには、`marker` リクエストパラメータのマーカー値を S3 Glacier として、新たにパートのリストのリクエストを送信します。前のパートのリストのリクエストに応じた結果が得られます。

リクエストで `limit` パラメータを指定して、レスポンスで返されるパート数を制限することもできます。

リクエスト

構文

マルチパートアップロード進行中のパートをリストするには、マルチパートアップロード ID リソースの URI に `GET` リクエストを送信します。マルチパートアップロードを開始すると、マルチパートアップロード ID が返されます (マルチパートアップロードの開始 (POST multipart-uploads))。オプションで `marker` パラメータと `limit` パラメータを指定できます。

```
GET /AccountId/vaults/VaultName/multipart-uploads/uploadID HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

名前	説明	必須
`limit`		いいえ

名前	説明	必須
	返されるパートの最大数。デフォルトの上限は 50 です。返されるパートの数は、指定した上限を下回ることはあっても、上限を上回ることはありません。 タイプ: 文字列 制約: 最小の整数値は 1 です。最大の整数値は 50 です。	
marker	パートのリスト表示を開始するパートをmarker指定する、ページ分割に使用される不透明な文字列。先行するパートのリストに対するレスポンスから、marker 値を取得します。以前にパートのリストのリクエストで開始された結果のページ分割を継続する場合にのみ、marker を指定する必要があります。 タイプ: 文字列 制約: なし	いいえ

リクエストヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「共通のレスポンスヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: Length

{
```

```
    "ArchiveDescription" : String,
    "CreationDate" : String,
    "Marker": String,
    "MultipartUploadId" : String,
    "PartSizeInBytes" : Number,
    "Parts" :
    [ {
      "RangeInBytes" : String,
      "SHA256TreeHash" : String
      },
      ...
     ],
    "VaultARN" : String
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

レスポンス本文には次の JSON フィールドが含まれています。

ArchiveDescription

マルチパートアップロードの開始リクエストに指定されたアーカイブの説明。マルチパートアップロードの開始オペレーションでアーカイブの説明が指定されていなければ、このフィールドは `null` です。

型: 文字列

CreationDate

マルチパートアップロードが開始された UTC 時間。

型: 文字列 たとえば、ISO 8601 の日付形式の文字列表現。`2013-03-20T17:03:43.221Z`。

Marker

結果のページ分割をどこから継続するかを表す不透明な文字列。リストに含まれるジョブをさらに取得するには、パートのリストの新規リクエストで `marker` を使用します。これ以上のパートが存在しない場合は、この値は `null` です。

型: 文字列

MultipartUploadId

パートに関連しているアップロードの ID。

型: 文字列

PartSizeInBytes

パートのサイズ (バイト単位)。これはマルチパートアップロードの開始リクエストで指定したものと同じ値です。

タイプ: 数値

Parts

マルチパートアップロードのパートサイズのリスト。配列内の各オブジェクトは、`RangeBytes` と `sha256-tree-hash` の名前と値のペアを含みます。

型: 配列

RangeInBytes

範囲の上限を含む、パートのバイト範囲。

型: 文字列

SHA256TreeHash

対象のパートについて S3 Glacier が計算した SHA256 木構造ハッシュ値。このフィールドが `null` になることはありません。

型: 文字列

VaultARN

マルチパートアップロードが開始されたボールトの Amazon リソースネーム (ARN) 。

型: 文字列

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

例: マルチパートアップロードのパートのリスト

以下の例は、アップロードのすべてのパートをリストします。この例では、進行中のマルチパートアップロードに対する特定のマルチパートアップロード ID の URI に HTTP GET リクエストを送信し、その結果、最大 1,000 パートが返されます。

リクエストの例

```
GET /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

レスポンスでは、S3 Glacier が指定のマルチパートアップロード ID に関連するアップロード済みパートのリストを返します。この例では、パートは 2 つのみです。返される `Marker` フィールドは `null` であり、マルチパートアップロードにはこれ以上のパートがないことを示します。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 412

{
    "ArchiveDescription" : "archive description",
    "CreationDate" : "2012-03-20T17:03:43.221Z",
    "Marker": null,
    "MultipartUploadId" :
 "OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE",
    "PartSizeInBytes" : 4194304,
    "Parts" :
    [ {
```

```
      "RangeInBytes" : "0-4194303",
      "SHA256TreeHash" : "01d34dabf7be316472c93b1ef80721f5d4"
      },
      {
      "RangeInBytes" : "4194304-8388607",
      "SHA256TreeHash" : "0195875365afda349fc21c84c099987164"
      }],
    "VaultARN" : "arn:aws:glacier:us-west-2:012345678901:vaults/demo1-vault"
}
```

例: マルチパートアップロードのパートのリスト (マーカーとリクエスト制限のパラメータを指定)

以下の例は、ページ分割を使用して結果の取得数を制限する方法を示しています。この例では、進行中のマルチパートアップロードに対する特定のマルチパートアップロード ID の URI に HTTP GET リクエストを送信し、その結果、1 パートが返されます。開始 marker パラメータは、パートリストを開始するパーツを指定します。パートリストに対する以前のリクエストで得られたレスポンスから、marker 値を取得できます。さらにこの例では、limit パラメータが 1 に設定されるため、1 つのパートが返されます。Marker フィールドが null ではないことに注目してください。取得できるパートが少なくともあと 1 つはあることを示しています。

リクエストの例

```
GET /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE?marker=1001&limit=1 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

レスポンスでは、S3 Glacier が指定された進行中のマルチパートアップロード ID に関連するアップロード済みパートのリストを返します。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: text/json
```

```
Content-Length: 412

{
    "ArchiveDescription" : "archive description 1",
    "CreationDate" : "2012-03-20T17:03:43.221Z",
    "Marker": "MfgsKHVjbQ6EldVl72bn3_n5h2TaGZQUO-Qb3B9j3TITf7WajQ",
    "MultipartUploadId" :
 "OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE",
    "PartSizeInBytes" : 4194304,
    "Parts" :
    [ {
      "RangeInBytes" : "4194304-8388607",
      "SHA256TreeHash" : "01d34dabf7be316472c93b1ef80721f5d4"
      }],
    "VaultARN" : "arn:aws:glacier:us-west-2:012345678901:vaults/demo1-vault"
}
```

関連するセクション

- マルチパートアップロードの開始 (POST multipart-uploads)
- パートのアップロード (PUT uploadID)
- マルチパートアップロードの完了 (POST uploadID)
- マルチパートアップロードの中止 (DELETE uploadID)
- マルチパートアップロードのリスト (GET multipart-uploads)
- パート単位での大きなアーカイブのアップロード (マルチパートアップロード)
- Amazon S3 Glacier の ID とアクセス管理

マルチパートアップロードのリスト (GET multipart-uploads)

説明

このマルチパートアップロードオペレーションでは、指定されたボールトの進行中のマルチパートアップロードのリストを表示します。進行中のマルチパートアップロードとは、マルチパートアップロードの開始 (POST multipart-uploads) リクエストによって開始されているものの、まだ完了または中止されていないマルチパートアップロードです。マルチパートアップロードのリストのレスポンスで返されるリストの順序は保証されていません。

マルチパートアップロードのリストオペレーションでは、ページ分割をサポートしています。このオペレーションによってレスポンスに返されるマルチパートアップロードは、デフォルトで最大 50 個です。レスポンス本文の `marker` フィールドに、リストの続きを表示するためのマーカーが含まれているかどうかを常に確認する必要があります。項目がそれ以上存在しない場合、 `marker` フィールドは `null` です。

`marker` が null でない場合に、マルチパートアップロードの次のセットを取得するには、前のマルチパートアップロードのリストのリクエストの結果として返されたマーカー値 Amazon S3 Glacier (S3 Glacier) を リクエストパラメータに設定した別のマルチパートアップロードのリストのリクエストを送信します。

このオペレーションと パートのリスト (GET uploadID) オペレーションの違いに注意してください。マルチパートアップロードのリストオペレーションでは、ボールトのすべてのマルチパートアップロードのリストを表示します。パートのリストオペレーションでは、アップロード ID によって識別される特定のマルチパートアップロードのパートが返されます。

マルチパートアップロードの詳細については、「パート単位での大きなアーカイブのアップロード (マルチパートアップロード)」を参照してください。

リクエスト

構文

マルチパートアップロードのリストを表示するには、ボールトの `GET` サブリソースの URI に `multipart-uploads` リクエストを送信します。オプションで `marker` パラメータと `limit` パラメータを指定できます。

```
GET /AccountId/vaults/VaultName/multipart-uploads HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

名前	説明	必須
limit	レスポンス本文に返されるアップロードの最大数を指定します。指定しない場合、アップロードのリストオペレーションでは、最大で 50 個のアップロードが返されます。 タイプ: 文字列 制約: 最小の整数値は 1 です。最大の整数値は 50 です。	いいえ
marker	ページ分割に使用する不透明な文字列。marker には、アップロードのリストを開始するアップロードを指定します。marker 値は、前回のアップロードのリストのレスポンスから取得します。前回のアップロードのリストのリクエストで開始された結果のページ分割を継続する場合のみ、marker を指定する必要があります。 タイプ: 文字列 制約: なし	いいえ

リクエストヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「共通のレスポンスヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

```
Content-Type: application/json
Content-Length: Length

{
  "Marker": String,
  "UploadsList" : [
    {
      "ArchiveDescription": String,
      "CreationDate": String,
      "MultipartUploadId": String,
      "PartSizeInBytes": Number,
      "VaultARN": String
    },
   ...
  ]
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

レスポンス本文には次の JSON フィールドが含まれています。

ArchiveDescription

マルチパートアップロードの開始リクエストに指定されたアーカイブの説明。マルチパートアップロードの開始オペレーションでアーカイブの説明が指定されていなければ、このフィールドは `null` です。

型: 文字列

CreationDate

マルチパートアップロードが開始された UTC 時間。

型: 文字列 たとえば、ISO 8601 の日付形式の文字列表現。`2013-03-20T17:03:43.221Z`。

Marker

結果のページ分割をどこから継続するかを表す不透明な文字列。リストに含まれるアップロードをさらに取得するには、マルチパートアップロードのリストの新規リクエストで `marker` を使用します。アップロードがそれ以上存在しない場合、この値は `null` です。

型: 文字列

PartSizeInBytes

マルチパートアップロードの開始 (POST multipart-uploads) リクエストで指定されたパートサイズ。これは、アップロードのすべてのパートのサイズです。ただし、最後のパートは、このサイズより小さくなる場合があります。

タイプ: 数値

MultipartUploadId

マルチパートアップロードの ID。

型: 文字列

UploadsList

マルチパートアップロードのオブジェクトに関するメタデータのリスト。`ArchiveDescription`、`CreationDate`、`MultipartUploadId`、`PartSizeInBytes`、`VaultA`
など、対応するアップロードの名前と値のペアのセットを含むリスト内の各項目。

型: 配列

VaultARN

アーカイブを含むボールトの Amazon リソースネーム (ARN) 。

型: 文字列

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

例: すべてのマルチパートアップロードのリスト

次の例では、ボールトの進行中のすべてのマルチパートアップロードのリストを表示します。この例では、指定されたボールトの `GET` サブリソースの URI に対する HTTP `multipart-uploads` リクエストを示しています。リクエストでは、`marker` および `limit` パラメータが指定されていないため、最大 1,000 個の進行中のマルチパートアップロードが返されます。

リクエストの例

```
GET /-/vaults/examplevault/multipart-uploads HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

レスポンスでは、S3 Glacier により、指定されたボールトの進行中のすべてのマルチパートアップロードのリストが返されます。`marker` フィールドは `null` です。これは、これ以上表示するアップロードがないことを示しています。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 1054

{
  "Marker": null,
  "UploadsList": [
    {
      "ArchiveDescription": "archive 1",
      "CreationDate": "2012-03-19T23:20:59.130Z",
      "MultipartUploadId":
 "xsQdFIRsfJr20CW2AbZBKpRZAFTZSJIMtL2hYf8mvp8dM0m4RUzlaqoEye6g3h3ecqB_zqwB7zLDMeSWhwo65re4C4Ev"
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    },
    {
      "ArchiveDescription": "archive 2",
      "CreationDate": "2012-04-01T15:00:00.000Z",
      "MultipartUploadId": "nPyGOnyFcx67qqX7E-0tSGiRi88hHMOwOxR-
_jNyM6RjVMFfV29lFqZ3rNsSaWBugg6OP92pRtufeHdQH7ClIpSF6uJc",
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    },
    {
      "ArchiveDescription": "archive 3",
```

```
      "CreationDate": "2012-03-20T17:03:43.221Z",
      "MultipartUploadId": "qt-RBst_7yO8gVIonIBsAxr2t-db0pE4s8MNeGjKjGdNpuU-
cdSAcqG62guwV9r5jh5mLyFPzFEitTpNE7iQfHiu1XoV",
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    }
  ]
}
```

例: マルチパートアップロードの部分的なリスト

以下の例は、ページ分割を使用して結果の取得数を制限する方法を示しています。この例では、指定されたボールトの GET サブリソースの URI に対する HTTP `multipart-uploads` リクエストを示しています。この例では、`limit` パラメータが 1 に設定されています。そのため、リストに返されるアップロードは 1 つのみで、`marker` パラメータには返されたリストが開始するマルチパートアップロード ID が示されます。

リクエストの例

```
GET /-/vaults/examplevault/multipart-uploads?
limit=1&marker=xsQdFIRsfJr20CW2AbZBKpRZAFTZSJIMtL2hYf8mvp8dM0m4RUzlaqoEye6g3h3ecqB_zqwB7zLDMeSW
 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

レスポンスでは、Amazon S3 Glacier (S3 Glacier) により、指定されたボールトの進行中のマルチパートアップロードのリストが 2 つまで返されます。指定されたマーカーから開始し、2 つの結果が返されます。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 470

{
```

```
  "Marker": "qt-RBst_7yO8gVIonIBsAxr2t-db0pE4s8MNeGjKjGdNpuU-
cdSAcqG62guwV9r5jh5mLyFPzFEitTpNE7iQfHiu1XoV",
  "UploadsList" : [
    {
      "ArchiveDescription": "archive 2",
      "CreationDate": "2012-04-01T15:00:00.000Z",
      "MultipartUploadId": "nPyGOnyFcx67qqX7E-0tSGiRi88hHMOwOxR-
_jNyM6RjVMFfV29lFqZ3rNsSaWBugg6OP92pRtufeHdQH7ClIpSF6uJc",
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    }
  ]
}
```

関連するセクション

- マルチパートアップロードの開始 (POST multipart-uploads)
- パートのアップロード (PUT uploadID)
- マルチパートアップロードの完了 (POST uploadID)
- マルチパートアップロードの中止 (DELETE uploadID)
- パートのリスト (GET uploadID)
- パート単位での大きなアーカイブのアップロード (マルチパートアップロード)
- Amazon S3 Glacier の ID とアクセス管理

パートのアップロード (PUT uploadID)

説明

このマルチパートアップロードオペレーションでは、アーカイブのパートをアップロードします。パートのアップロードのリクエストでは、アセンブル済みのアーカイブの中からこのパートでアップロードするバイト範囲を指定することになるため、任意の順序でアーカイブのパートをアップロードできます。このほか、複数のパートを並行してアップロードすることもできます。マルチパートアップロードでは、最大 10,000 パートを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、「パート単位での大きなアーカイブのアップロード (マルチパートアップロード)」を参照してください。

Amazon S3 Glacier (S3 Glacier) では、以下の条件のいずれかが true のときは、パートのアップロードのリクエストが拒否されます。

- SHA256 木構造ハッシュが一致しない- 送信中にパートデータが破損していないことを確認するため、パートの SHA256 木構造ハッシュを計算し、リクエストに含める必要があります。パートデータを受け取った時点で、S3 Glacier でも SHA256 木構造ハッシュを計算します。この 2 つのハッシュ値が一致しない場合、オペレーションは失敗となります。SHA256 木構造ハッシュの計算の詳細については、「チェックサムの計算」を参照してください。
- SHA256 の線形ハッシュが一致しない-認可のため、アップロードされるペイロード全体の SHA256 線形ハッシュを計算し、リクエストに含める必要があります。SHA256 線形ハッシュの計算については、「チェックサムの計算」を参照してください。
- パートのサイズが一致しない-(最後のパートを除く) 各パートのサイズは、対応する マルチパートアップロードの開始 (POST multipart-uploads) リクエストで指定されているサイズに一致している必要があります。最後のパートのサイズは、指定されたサイズ以下になっていることが必要です。

  > **Note**
  >
  > マルチパートアップロードの開始リクエストで指定したサイズよりも小さなサイズのパートをアップロードし、そのパートが最後のパートでない場合でも、パートのアップロードのリクエストは成功します。ただし、次のマルチパートアップロードの完了リクエストは失敗します。

- 範囲が一致しない-リクエストのバイト範囲の値が、対応する開始リクエストで指定されているパートサイズと一致しない場合です。たとえば、パートサイズを 4194304 バイト (4 MB) に指定した場合であれば、0-4194303 バイト (4 MB-1) および 4194304 (4 MB) -8388607 (8 MB-1) が有効なパート範囲になります。これに対して、範囲の値を 2 MB ~ 6 MB に設定した場合には、範囲がパートのサイズに一致しないため、アップロードが失敗します。

このオペレーションはべき等です。同じパートを複数回アップロードすると、最新のリクエストに含まれるデータによって、既にアップロードされたデータが上書きされます。

リクエスト

PUTリクエストは、マルチパートアップロードの開始リクエストによって返されるアップロード ID の URI に送信されます。S3 Glacier では、アップロード ID を使用してパートアップロードを特定のマルチパートアップロードに関連付けます。リクエストには、パートデータの SHA256 木構造ハッシュ (`x-amz-SHA256-tree-hash` ヘッダー)、ペイロード全体の SHA256 線形ハッシュ (`x-amz-`

content-sha256 ヘッダー)、バイト範囲 (Content-Range ヘッダー)、およびパートの長さのバイト数 (Content-Length ヘッダー) を含める必要があります。

構文

```
PUT /AccountId/vaults/VaultName/multipart-uploads/uploadID HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Range: ContentRange
Content-Length: PayloadSize
Content-Type: application/octet-stream
x-amz-sha256-tree-hash: Checksum of the part
x-amz-content-sha256: Checksum of the entire payload
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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

この操作では、すべての操作で共通のリクエストヘッダーに加えて、次のリクエストヘッダーを使用します。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

名前	説明	必須
Content-Length	パートの長さをバイト単位で特定します。 タイプ: 文字列	いいえ

名前	説明	必須
	デフォルト: なし 制約: なし	
Content-Range	このパートでアップロードされるアセンブル済みアーカイブのバイト範囲を識別します。S3 Glacier では、この情報を使用してアーカイブを適切な順序でアセンブルします。このヘッダーの形式は、RFC 2616 に準拠しています。ヘッダーはたとえば、`Content-Range:bytes 0-4194303/*` のようになります。 タイプ: 文字列 デフォルト: なし 制約事項: この範囲は、マルチパートアップロードを開始するときに指定したパートサイズよりも大きくすることはできません。	はい
x-amz-content-sha256	アップロードされるペイロードの SHA256 チェックサム (線形ハッシュ)。この値は、`x-amz-sha256-tree-hash` ヘッダーで指定する値とは異なります。 タイプ: 文字列 デフォルト: なし 制約: なし	はい

名前	説明	必須
x-amz-sha256-tree-hash	アップロードするデータの SHA256 木構造ハッシュを指定します。SHA256 木構造ハッシュの計算の詳細については、「チェックサムの計算」を参照してください。 タイプ: 文字列 デフォルト: なし 制約: なし	はい

リクエスト本文

リクエストボディには、アップロードするデータを含めます。

レスポンス

S3 Glacier は、パートのアップロードが正常に終了すると、204 No Content レスポンスを返します。

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-sha256-tree-hash: ChecksumComputedByAmazonGlacier
```

レスポンスヘッダー

成功したレスポンスには、すべての操作に共通のレスポンスヘッダーに加えて、次のレスポンスヘッダーが含まれます。共通のレスポンスヘッダーの詳細については、「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-sha256-tree-hash	

名前	説明
	アップロードされたパートについて S3 Glacier が計算した SHA256 木構造ハッシュ。 タイプ: 文字列

レスポンス本文

このオペレーションはレスポンス本文を返しません。

例

以下に示すリクエストでは、4 MB のパートをアップロードします。このリクエストでは、このパートをアーカイブの最初のパートにするようにバイト範囲を設定しています。

リクエストの例

この例では、4 MB のパートをアップロードする HTTP PUT リクエストを送信します。リクエストは、マルチパートアップロードの開始リクエストによって返されるアップロード ID の URI に送信されます。`Content-Range` ヘッダーでは、このパートがアーカイブの最初の 4 MB のデータに相当するものであることを指定しています。

```
PUT /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Range:bytes 0-4194303/*
x-amz-sha256-tree-hash:c06f7cd4baacb087002a99a5f48bf953
x-amz-content-sha256:726e392cb4d09924dbad1cc0ba3b00c3643d03d14cb4b823e2f041cff612a628
Content-Length: 4194304
Authorization: Authorization=AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-
amz-glacier-
version,Signature=16b9a9e220a37e32f2e7be196b4ebb87120ca7974038210199ac5982e792cace
```

次のパートをアップロードする手順も、これと同じです。ただし、アップロードするパートの新しい SHA256 木構造ハッシュを計算するとともに、新しいバイト範囲を指定し、そのパートが最後のアセンブリで置かれる場所を示す必要があります。次のリクエストは、同じアップロード ID を使用して

別のパートをアップロードするものです。このリクエストでは、アーカイブのうち、前回のリクエスト以降の 4 MB 分の範囲と、4 MB というパートサイズを指定しています。

```
PUT /-/vaults/examplevault/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Range:bytes 4194304-8388607/*
Content-Length: 4194304
x-amz-sha256-tree-hash:f10e02544d651e2c3ce90a4307427493
x-amz-content-sha256:726e392cb4d09924dbad1cc0ba3b00c3643d03d14cb4b823e2f041cff612a628
x-amz-glacier-version: 2012-06-01
Authorization: Authorization=AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/
us-west-2/glacier/aws4_request, SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-
amz-glacier-version,
 Signature=16b9a9e220a37e32f2e7be196b4ebb87120ca7974038210199ac5982e792cace
```

パートは、任意の順序でアップロードできます。S3 Glacier では、各パートの範囲として指定された値を使用して、アセンブルする順序を決定します。

レスポンスの例

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
x-amz-sha256-tree-hash: c06f7cd4baacb087002a99a5f48bf953
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

関連するセクション

- マルチパートアップロードの開始 (POST multipart-uploads)
- パートのアップロード (PUT uploadID)
- マルチパートアップロードの完了 (POST uploadID)
- マルチパートアップロードの中止 (DELETE uploadID)
- マルチパートアップロードのリスト (GET multipart-uploads)
- パートのリスト (GET uploadID)
- パート単位での大きなアーカイブのアップロード (マルチパートアップロード)
- Amazon S3 Glacier の ID とアクセス管理

ジョブのオペレーション

以下は、S3 Glacier で使用できるジョブオペレーションです。

トピック

- ジョブの説明 (GET JobID)
- ジョブの出力の取得 (GET output)
- ジョブの開始 (ジョブの POST)
- ジョブのリスト表示 (GET jobs)

ジョブの説明 (GET JobID)

説明

このオペレーションでは、以前に開始したジョブに関する情報 (ジョブの開始日、ジョブを開始したユーザー、ジョブのステータスコード、ジョブのステータスメッセージ)、Amazon S3 Glacier (S3 Glacier) がジョブの完了後に通知する Amazon Simple Notification Service (Amazon SNS) トピックなどが返されます。ジョブの開始の詳細については、「ジョブの開始 (ジョブの POST)」を参照してください。

> **Note**
>
> このオペレーションにより、ジョブのステータスを確認することができます。ただし、S3 Glacier がジョブの完了後に Amazon SNS トピックに通知できるように、トピックを設定してジョブの開始リクエストでそのトピックを指定することを強くお勧めします。

ジョブ ID は、S3 Glacier がジョブを完了してから少なくとも 24 時間は有効です。

リクエスト

構文

ジョブに関する情報を取得するには、HTTP GET メソッドを使用し、特定のジョブをリクエストの範囲として指定します。相対 URI パスは、ジョブを開始したときに S3 Glacier によって返されたパスと同じであることに注意してください。

```
GET /AccountID/vaults/VaultName/jobs/JobID HTTP/1.1
```

```
Host: glacier.Region.amazonaws.com
Date: date
Authorization: signatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

> **Note**
>
> リクエストで JobID を指定しなかった場合、レスポンスでは、指定したボールトのすべてのアクティブなジョブのリストが返されます。ジョブのリストの詳細については、「ジョブのリスト表示 (GET jobs)」を参照してください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

```
Content-Type: application/json
Content-Length: Length

{
    "Action": "string",
    "ArchiveId": "string",
    "ArchiveSHA256TreeHash": "string",
    "ArchiveSizeInBytes": number,
    "Completed": boolean,
    "CompletionDate": "string",
    "CreationDate": "string",
    "InventoryRetrievalParameters": {
        "EndDate": "string",
        "Format": "string",
        "Limit": "string",
        "Marker": "string",
        "StartDate": "string"
    },
    "InventorySizeInBytes": number,
    "JobDescription": "string",
    "JobId": "string",
    "JobOutputPath": "string",
    "OutputLocation": {
        "S3": {
            "AccessControlList": [
                {
                    "Grantee": {
                        "DisplayName": "string",
                        "EmailAddress": "string",
                        "ID": "string",
                        "Type": "string",
                        "URI": "string"
                    },
                    "Permission": "string"
                }
            ],
            "BucketName": "string",
            "CannedACL": "string",
            "Encryption": {
                "EncryptionType": "string",
                "KMSContext": "string",
                "KMSKeyId": "string"
            },
            "Prefix": "string",
```

```
            "StorageClass": "string",
            "Tagging": {
                "string": "string"
            },
            "UserMetadata": {
                "string": "string"
            }
        }
    },
    "RetrievalByteRange": "string",
    "SelectParameters": {
        "Expression": "string",
        "ExpressionType": "string",
        "InputSerialization": {
            "csv": {
                "Comments": "string",
                "FieldDelimiter": "string",
                "FileHeaderInfo": "string",
                "QuoteCharacter": "string",
                "QuoteEscapeCharacter": "string",
                "RecordDelimiter": "string"
            }
        },
        "OutputSerialization": {
            "csv": {
                "FieldDelimiter": "string",
                "QuoteCharacter": "string",
                "QuoteEscapeCharacter": "string",
                "QuoteFields": "string",
                "RecordDelimiter": "string"
            }
        }
    },
    "SHA256TreeHash": "string",
    "SNSTopic": "string",
    "StatusCode": "string",
    "StatusMessage": "string",
    "Tier": "string",
    "VaultARN": "string"
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

レスポンス本文には次の JSON フィールドが含まれています。

Action

ジョブのタイプ。`ArchiveRetrieval`、`InventoryRetrieval`、または `Select` です。

型: 文字列

ArchiveId

選択ジョブまたはアーカイブの取得ジョブにリクエストされたアーカイブ ID。それ以外の場合、このフィールドは `null` です。

型: 文字列

ArchiveSHA256TreeHash

アーカイブの取得ジョブを行うアーカイブ全体の SHA256 木構造ハッシュ。インベントリの取得ジョブの場合、このフィールドは `null` です。

型: 文字列

ArchiveSizeInBytes

`ArchiveRetrieval` ジョブの場合、これはダウンロードに必要なアーカイブのサイズ (バイト単位) です。`InventoryRetrieval` ジョブの場合、この値は `null` です。

タイプ: 数値

Completed

ジョブのステータス。アーカイブの取得ジョブまたはインベントリの取得ジョブが完了したら、ジョブの出力の取得 (GET output) を使用してジョブの出力を取得します。

型: ブール

CompletionDate

ジョブリクエストが完了した協定世界時 (UTC) 時間。ジョブが進行中の場合、値は null です。

型: 文字列

CreationDate

ジョブが作成された UTC 時間。

型: ISO 8601 日付形式の文字列表現。例えば、`2013-03-20T17:03:43.221Z` のようになります。

InventoryRetrievalParameters

インベントリの取得の範囲に使用される入力パラメータ。

型: InventoryRetrievalJobInput オブジェクト

InventorySizeInBytes

`InventoryRetrieval` ジョブの場合、これはダウンロードに必要なインベントリのサイズ (バイト単位) です。`ArchiveRetrieval` または `Select` ジョブの場合、値は `null` です。

型: 数値

JobDescription

ジョブを開始したときに指定したジョブの説明。

型: 文字列

JobId

S3 Glacier 内のジョブを識別する ID。

型: 文字列

JobOutputPath

ジョブの出力場所が含まれます。

型: 文字列

OutputLocation

選択ジョブの結果とエラーが保存されている場所についての情報を含むオブジェクト。

型: OutputLocation オブジェクト

RetrievalByteRange

"*`StartByteValue`*-*`EndByteValue`*" という形式で示される、アーカイブの取得ジョブで取得したバイト範囲。アーカイブの取得で範囲を指定しなかった場合は、アーカイブ全体が取得され、

また StartByteValue は 0、EndByteValue はアーカイブのサイズから 1 を引いた値になります。インベントリの取得ジョブまたは選択ジョブの場合、このフィールドは `null` です。

型: 文字列

SelectParameters

選択に使用されるパラメータに関する情報を含むオブジェクト。

型: SelectParameters オブジェクト

SHA256TreeHash

アーカイブのリクエストされた範囲の SHA256 木構造ハッシュ値。アーカイブのジョブの開始 (ジョブの POST)リクエストで木構造ハッシュ可能な範囲を指定した場合、このフィールドに値が返されます。アーカイブの範囲取得で木構造ハッシュを可能にするための調整の詳細については、「データをダウンロードするときのチェックサムの受信」を参照してください。

アーカイブ全体を取得する特別な場合は、この値は `ArchiveSHA256TreeHash` の値と同じです。

次の場合、このフィールドは `null` です。

- 木構造ハッシュ可能ではない範囲を指定したアーカイブの取得ジョブ。
- アーカイブ全体を範囲に指定したアーカイブのジョブで、ジョブのステータスが `InProgress` の場合。
- インベントリジョブ。
- 選択ジョブ。

型: 文字列

SNSTopic

通知を受け取る Amazon SNS トピック。

型: 文字列

StatusCode

ジョブのステータスを示すコード。

有効な値: `InProgress` | `Succeeded` | `Failed`

型: 文字列

StatusMessage

ジョブのステータスを説明するわかりやすいメッセージ。

型: 文字列

Tier

選択またはアーカイブの取得に使用するデータアクセス層。

有効な値: `Bulk` | `Expedited` | `Standard`

型: 文字列

VaultARN

ジョブがサブリソースとなるボールトの Amazon リソースネーム (ARN)。

型: 文字列

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

以下の例は、アーカイブの取得ジョブのリクエストを示しています。

リクエストの例: ジョブの説明を取得する

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

レスポンス本文には指定したジョブを説明する JSON が含まれます。インベントリの取得ジョブおよびアーカイブの取得ジョブの場合、JSON フィールドはどちらも同じであることに注意してくださ

い。ただし、ジョブのタイプに適用されないフィールドの値は null です。以下は、アーカイブの取得ジョブに対するレスポンスの例です。次の点に注意してください:

- Action フィールドの値は ArchiveRetrieval です。
- ArchiveSizeInBytes フィールドは、アーカイブの取得ジョブでリクエストされたアーカイブのサイズを示しています。
- ArchiveSHA256TreeHash フィールドは、アーカイブ全体の SHA256 木構造ハッシュを示しています。
- RetrievalByteRange フィールドは、ジョブの開始リクエストでリクエストされた範囲を示しています。この例では、アーカイブ全体をリクエストしています。
- SHA256TreeHash フィールドは、ジョブの開始リクエストでリクエストされた範囲の SHA256 木構造ハッシュを示しています。この例では、ArchiveSHA256TreeHash フィールドと同じ値になっています。これは、アーカイブ全体がリクエストされたことを意味します。
- InventorySizeInBytes フィールド値は null です。このフィールドは、アーカイブの取得ジョブには適用されないためです。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 419
{
  "Action": "ArchiveRetrieval",
  "ArchiveId": "NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
  "ArchiveSizeInBytes": 16777216,
  "ArchiveSHA256TreeHash":
 "beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60",
  "Completed": false,
  "CompletionDate": null,
  "CreationDate": "2012-05-15T17:21:39.339Z",
  "InventorySizeInBytes": null,
  "JobDescription": "My ArchiveRetrieval Job",
  "JobId": "HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID",
  "RetrievalByteRange": "0-16777215",
  "SHA256TreeHash": "beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60",
  "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic",
```

```
  "StatusCode": "InProgress",
  "StatusMessage": "Operation in progress.",
  "Tier": "Bulk",
  "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
}
```

以下は、インベントリの取得ジョブに対するレスポンスの例です。次の点に注意してください:

- Action フィールドの値は InventoryRetrieval です。
- ArchiveSizeInBytes フィールド、ArchiveSHA256TreeHash フィールド、および RetrievalByteRange フィールドの値は null です。これらのフィールドはインベントリの取得ジョブに適用されないためです。
- InventorySizeInBytes フィールドの値は null です。ジョブが進行中であり、インベントリをダウンロードする準備が完了していないためです。ジョブの説明リクエストの前にジョブが完了していた場合、このフィールドには出力のサイズが示されることになります。

```
{
   "Action": "InventoryRetrieval",
   "ArchiveId": null,
   "ArchiveSizeInBytes": null,
   "ArchiveSHA256TreeHash": null,
   "Completed": false,
   "CompletionDate": null,
   "CreationDate": "2012-05-15T23:18:13.224Z",
   "InventorySizeInBytes": null,
   "JobDescription": "Inventory Description",
   "JobId": "HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID",
   "RetrievalByteRange": null,
   "SHA256TreeHash": null,
   "SNSTopic": "arn:aws:sns:us-west-2:012345678901:mytopic",
   "StatusCode": "InProgress",
   "StatusMessage": "Operation in progress.",
   "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
}
```

以下は、完了したインベントリ取得ジョブのレスポンスの例です。ボールトインベントリの取得のページ分割を継続する場合に使用するマーカーが含まれています。

```
{
    "Action": "InventoryRetrieval",
    "ArchiveId": null,
    "ArchiveSHA256TreeHash": null,
    "ArchiveSizeInBytes": null,
    "Completed": true,
    "CompletionDate": "2013-12-05T21:51:13.591Z",
    "CreationDate": "2013-12-05T21:51:12.281Z",
    "InventorySizeInBytes": 777062,
    "JobDescription": null,
    "JobId": "sCC2RZNBF2nildYD_roe0J9bHRdPQUbDRkmTdg-mXi2u3lc49uW6TcEhDF2D9pB2phx-
BN30JaBru7PMyOlfXHdStzu8",
    "NextInventoryRetrievalMarker": null,
    "RetrievalByteRange": null,
    "SHA256TreeHash": null,
    "SNSTopic": null,
    "StatusCode": "Succeeded",
    "StatusMessage": "Succeeded",
    "Tier": "Bulk",
    "VaultARN": "arn:aws:glacier-devo:us-west-2:836579025725:vaults/inventory-
icecube-2",
    "InventoryRetrievalParameters": {
        "StartDate": "2013-11-12T13:43:12Z",
        "EndDate": "2013-11-20T08:12:45Z",
        "Limit": "120000",
        "Format": "JSON",
        "Marker":
 "vyS0t2jHQe5qbcDggIeD50chS1SXwYMrkVKo0KHiTUjEYxBGCqRLKaiySzdN7QXGVVV5XZpNVG67pCZ_uykQXFMLaxOSu
    },
}
```


関連するセクション

- ジョブの出力の取得 (GET output)
- Amazon S3 Glacier の ID とアクセス管理

ジョブの出力の取得 (GET output)

説明

このオペレーションでは、ジョブの開始 (ジョブの POST) を使用して開始したジョブの出力結果をダウンロードします。出力は、ジョブの開始時に指定したジョブタイプに応じて、アーカイブのコンテンツとボールトインベントリのいずれかとなります。

すべてのジョブの出力をダウンロードすることも、バイト範囲を指定して出力の一部をダウンロードすることもできます。アーカイブの取得ジョブでもインベントリの取得ジョブでも、ジョブの出力の取得レスポンスのヘッダーで返されたサイズとダウンロードされたサイズを比較して確認する必要があります。

アーカイブの取得ジョブでは、ダウンロードされたサイズが予期されたサイズであることも確認する必要があります。出力の一部をダウンロードする場合は、指定したバイト範囲が予期されるサイズとなります。たとえば、`bytes=0-1048575` の範囲を指定した場合、ダウンロードサイズが 1,048,576 バイトであることを確認する必要があります。アーカイブ全体をダウンロードする場合は、Amazon S3 Glacier (S3 Glacier) にアップロードした際のアーカイブのサイズが予期されるサイズになります。予期されるサイズは、ジョブの出力の取得レスポンスのヘッダーでも返されます。

アーカイブを取得するジョブの場合、指定したバイト範囲に応じて、S3 Glacier によってデータの一部のチェックサムが返されます。ダウンロードしたデータが正しい部分であることを確認するには、クライアント側でチェックサムを計算し、値が一致していること、および予期されたサイズがダウンロードされていることを確認します。

ジョブ ID は、S3 Glacier がジョブを完了してから少なくとも 24 時間は有効です。つまり、S3 Glacier がジョブを完了してから 24 時間以内であれば、ジョブの出力をダウンロードできます。

リクエスト

構文

ジョブの出力を取得するには、特定のジョブの `GET` の URI に HTTP `output` リクエストを送信します。

```
GET /AccountId/vaults/VaultName/jobs/JobID/output HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
```

```
Range: ByteRangeToRetrieve
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

このオペレーション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

この操作では、すべての操作で共通のリクエストヘッダーに加えて、次のリクエストヘッダーを使用します。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

名前	説明	必須
Range	出力から取得するバイト範囲。たとえば、最初の 1,048,576 バイトをダウンロードする場合は、範囲を「bytes=0-1048575 」と指定します。詳細については、「Range ヘッダーフィールドの定義」を参照してください。範囲は、ジョブの開始リクエストで指定された範囲に応じて決まります。このオペレーションではデフォルトで、出力全体をダウンロードするようになっています。 ジョブの出力が大きい場合には、Range リクエストヘッダーを使用して出力の一部を取得することができます。このようにすると、出力全体を小さなバイトから成るチャンクに分けてダウンロードできます。たとえば、ダウンロードしようとしているジョブの出力が 1 GB であるため、ジョブの出力の取得リクエストを計 8 回送信して、1 回につき 128 MB ずつダ	いいえ

名前	説明	必須
	ウンロードするとします。この場合、ジョブの出力のダウンロードのプロセスは以下のようになります。 1. Range ヘッダーを使用し、適切なバイト範囲を指定して 128 MB 分の出力のチャンクをダウンロードします。128 MB のデータがすべて受信されたことを確認します。 2. レスポンスには、データのほかに、ペイロードのチェックサムが含まれています。クライアント側でペイロードのチェックサムを計算したうえで、レスポンスで受信したチェックサムと比較し、必要なデータがすべて受信できたことを確認します。 3. 出力データを 128 MB に分けたチャンク 8 つ全部について、ステップ 1 と 2 を繰り返します。バイト範囲は、毎回適切なものを指定します。 4. ジョブの出力をすべてダウンロードしたら、チェックサムの値 8 つをリストにまとめたものができあがります。その 8 つの値の木構造ハッシュを計算して、出力全体のチェックサムを導き出します。ジョブの説明 (GET JobID) オペレーションを使用して、出力を提供したジョブのジョブ情報を取得します。レスポンスには、S3 Glacier に保存されているアーカイブ全体のチェックサムが含まれます。計算したチェックサムとこの値を比較して、アーカイブのコンテンツ全体がエラーもなくダウンロードできたことを確認します。 タイプ: 文字列 デフォルト: なし 制約: なし	

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

取得リクエストがジョブデータをすべて返す場合には、ジョブの出力レスポンスによってレスポンスコード 200 OK が返されます。リクエストで Range ヘッダーを指定するなど、コンテンツの一部をリクエストした場合には、レスポンスコード 206 Partial Content が返されます。

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: ContentType
Content-Length: Length
x-amz-sha256-tree-hash: ChecksumComputedByAmazonGlacier

[Body containing job output.]
```

レスポンスヘッダー

ヘッダー	説明
Content-Range	S3 Glacier によって返されたバイト範囲。出力の一部のみをダウンロードする場合、レスポンスには S3 Glacier によって返されたバイト範囲が示されます。 たとえば、bytes 0-1048575/8388608 であれば最初の 1 MB ~ 8 MB が返されます。 Content-Range ヘッダーの詳細については、「Content-Range ヘッダーフィールドの定義」を参照してください。 タイプ: 文字列
Content-Type	Content-Type は、ジョブの出力がアーカイブであるかボールトインベントリであるかによって異なります。 • アーカイブデータであれば、Content-Type は application/octet-stream になります。

ヘッダー	説明
	• ボールトインベントリであれば、ジョブの開始時に CSV 形式をリクエストした場合には、Content-Type は `text/csv` になります。それ以外の場合には、ボールトインベントリはデフォルトで JSON として返され、Content-Type は `application/json` となります。 タイプ: 文字列

ヘッダー	説明
x-amz-sha256-tree-hash	レスポンスのデータのチェックサム。このヘッダーは、アーカイブの取得ジョブの出力を取得するときにのみ返されます。さらに、このヘッダーは、ジョブの開始リクエストで指定して取得したデータ範囲と、ジョブの出力の取得でダウンロードするデータ範囲のどちらも木構造ハッシュ可能になっている場合にも表示されます。木構造ハッシュ可能な範囲の詳細については、「データをダウンロードするときのチェックサムの受信」を参照してください。 たとえば、ジョブの開始リクエストで木構造ハッシュ可能になっている (アーカイブ全体が含まれる) 範囲を取得対象に指定した場合には、以下の条件に該当すると、ダウンロードするデータのチェックサムを受け取ることになります。 • 取得対象のデータの全範囲を取得した。 • 取得対象としたデータのバイト範囲が、1 メガバイト (1024 KB) に 2 のべき乗の数を乗じたサイズであり、始点および終点がリクエストした範囲のサイズの倍数である。たとえば、取得するデータが 3.1 MB であり、始点が 1 MB、終点が 2 MB の範囲が返されるように指定した場合には、レスポンスヘッダーとして x-amz-sha256-tree-hash が返されます。 • 取得対象となるデータの返される範囲がデータの終わりまでであり、その範囲の始点が、取得対象となる範囲のサイズの倍数で、切り上げると次の 2 のべき乗の値になるものの、1 メガバイト (1024 KB) 未満ではない。たとえば、取得するデータが 3.1 MB であり、始点が 2 MB、終点が 3.1 MB (データの終わり) の範囲を指定した場合には、レスポンスヘッダーとして x-amz-sha256-tree-hash が返されます。 タイプ: 文字列

レスポンス本文

S3 Glacier は、レスポンス本文にジョブの出力を返します。出力は、ジョブのタイプに応じて、アーカイブのコンテンツとボールトインベントリのいずれかになります。ボールトインベントリの場合、デフォルトでは、インベントリのリストが以下の JSON 本文として返されます。

```
{
 "VaultARN": String,
 "InventoryDate": String,
 "ArchiveList": [
      {"ArchiveId": String,
       "ArchiveDescription": String,
       "CreationDate": String,
       "Size": Number,
       "SHA256TreeHash": String
      },
      ...
    ]
}
```

ボールトインベントリジョブの開始時に出力形式としてカンマ区切り値 (CSV) をリクエストした場合には、本文でボールトインベントリが CSV 形式で返されます。CSV 形式は "ArchiveId"、"ArchiveDescription"、"CreationDate"、"Size"、および "SHA256TreeHash" の 5 列で構成されます。定義は、対応する JSON フィールドと同じです。

> **Note**
>
> CSV 形式では、フィールド全体が二重引用符で囲まれた状態で返されることがあります。フィールドにカンマまたは二重引用符が含まれる場合には、常に二重引用句で囲まれた状態で返されます。たとえば、`my archive description,1` は `"my archive description,1"` として返されます。二重引用符で囲まれたフィールドが返される場合に、そのフィールドに二重引用符があるときは、バックスラッシュ文字を前に置くことによってエスケープします。たとえば、`my archive description,1"2` であれば `"my archive description,1\"2"`、`my archive description,1\"2` であれば `"my archive description,1\\"2"` という形式で返されます。バックスラッシュ文字は、エスケープされません。

JSON レスポンス本文には次の JSON フィールドが含まれています。

ArchiveDescription

アーカイブの説明。

型: 文字列

ArchiveId

アーカイブの ID。

型: 文字列

ArchiveList

アーカイブのメタデータの配列。配列内の各オブジェクトは、ボールトに含まれる 1 つのアーカイブのメタデータを表しています。

型: 配列

CreationDate

アーカイブが作成された日時 (UTC) 。

タイプ: ISO 8601 の日付形式の文字列表現。たとえば`2013-03-20T17:03:43.221Z`。

InventoryDate

対象となるボールトが変更されてから作成が完了した最新のボールトインベントリの日時 (UTC) 。S3 Glacier では 1 日に 1 回ボールトインベントリを作成するものの、インベントリの日付は、最後のインベントリ以降にボールトに対してアーカイブの追加または削除があった場合にのみ更新されます。

タイプ: ISO 8601 の日付形式の文字列表現。たとえば`2013-03-20T17:03:43.221Z`。

SHA256TreeHash

アーカイブの木構造ハッシュ。

型: 文字列

[Size] (サイズ)

アーカイブのサイズのバイト数。

タイプ: 数値

VaultARN

アーカイブ取得がリクエストされた Amazon リソース ネーム (ARN) のリソース。

型: 文字列

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

以下の例は、アーカイブの取得ジョブのリクエストを示しています。

例 1: ダウンロードの出力

この例では、アーカイブの取得開始ジョブのリクエストに対するレスポンスとして S3 Glacier によって作成されるデータを取得します。

リクエストの例

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID/output
 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

以下は、アーカイブの取得ジョブに対するレスポンスの例です。`Content-Type` ヘッダーが `application/octet-stream` になっており、レスポンスに `x-amz-sha256-tree-hash` ヘッダーが含まれていることに注目してください。これは、ジョブデータがすべて返されることを示すものです。

```
HTTP/1.1 200 OK
```

```
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
x-amz-sha256-tree-hash:
 beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/octet-stream
Content-Length: 1048576

[Archive data.]
```

以下は、インベントリの取得ジョブに対するレスポンスの例です。Content-Type ヘッダーが application/json であることに注目してください。また、レスポンスには x-amz-sha256-tree-hash ヘッダーがありません。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 906

{
 "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault",
 "InventoryDate": "2011-12-12T14:19:01Z",
 "ArchiveList": [
   {
     "ArchiveId": "DMTmICA2n5Tdqq5BV2z7og-
A20xnpAPKt3UXwWxdWsn_D6auTUrW6kwy5Qyj9xd1MCE1mBYvMQ63LWaT8yTMzMaCxB_9VBWrW4Jw4zsvg5kehAPDVKcppU
oA",
     "ArchiveDescription": "my archive1",
     "CreationDate": "2012-05-15T17:19:46.700Z",
     "Size": 2140123,
     "SHA256TreeHash":
 "6b9d4cf8697bd3af6aa1b590a0b27b337da5b18988dbcc619a3e608a554a1e62"
   },
   {
     "ArchiveId": "2lHzwhKhgF2JHyvCS-
ZRuF08IQLuyB4265Hs3AXj9MoAIhz7tbXAvcFeHusgU_hViO1WeCBe0N5lsYYHRyZ7rrmRkNRuYrXUs_sjl2K8ume_7mKO_
uHE1oHqaW9d37pabXrSA",
     "ArchiveDescription": "my archive2",
     "CreationDate": "2012-05-15T17:21:39.339Z",
     "Size": 2140123,
     "SHA256TreeHash":
 "7f2fe580edb35154041fa3d4b41dd6d3adaef0c85d2ff6309f1d4b520eeecda3"
   }
```

```
  ]
}
```

例 2: 出力の一部のみダウンロード

この例は、アーカイブの取得開始ジョブのリクエストに対するレスポンスとして S3 Glacier が作成するアーカイブの一部のみを取得するものです。このリクエストでは、オプションの Range ヘッダーを使用して最初の 1,024 バイトのみを取得しています。

リクエストの例

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID/output
 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Range: bytes=0-1023
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

以下の正常なレスポンスは、206 Partial Content レスポンスを示しています。この例では、レスポンスにはほかにも、S3 Glacier によって返されるバイト範囲を指定する Content-Range ヘッダーが含まれています。

```
HTTP/1.1 206 Partial Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Range: bytes 0-1023/8388608
Content-Type: application/octet-stream
Content-Length: 1024

[Archive data.]
```

関連するセクション

- ジョブの説明 (GET JobID)

- ジョブの開始 (ジョブの POST)
- Amazon S3 Glacier の ID とアクセス管理

ジョブの開始 (ジョブの POST)

この操作では、次のタイプの Amazon S3 Glacier (S3 Glacier) ジョブを開始します。

- `archive-retrieval`- アーカイブを取得
- `inventory-retrieval`- ボールトのインベントリ

トピック

- アーカイブまたはボールトインベントリ取得ジョブの初期化
- リクエスト
- レスポンス
- 例
- 関連するセクション

アーカイブまたはボールトインベントリ取得ジョブの初期化

アーカイブの取得やボールトインベントリの取得は、ユーザーがジョブを開始する必要がある非同期オペレーションです。一度開始したジョブはキャンセルできません。取得は、2 ステップのプロセスです。

1. ジョブの開始 (ジョブの POST) オペレーションを使用して、取得ジョブを開始します。

> **Important**
>
> データ取り出しポリシーにより、`PolicyEnforcedException` が発生して、取り出しジョブの開始リクエストが失敗することがあります。データ取り出しポリシーの詳細については、「S3 Glacier データ取得ポリシー」を参照してください。`PolicyEnforcedException` 例外の詳細については、「エラーレスポンス」を参照してください。

2. ジョブが完了したら、ジョブの出力の取得 (GET output) オペレーションを使用してバイトをダウンロードします。

取得のリクエストは非同期的に実行されます。取得ジョブを開始すると、S3 Glacier はジョブを作成し、レスポンスでジョブ ID を返します。S3 Glacier がジョブを完了すると、ジョブの出力 (アーカイブデータまたはインベントリデータ) を取得できます。ジョブの出力の取得については、「ジョブの出力の取得 (GET output)」を参照してください。

出力を取得する前にジョブが完了している必要があります。次のオプションを使用してジョブの完了を確認できます。

- Amazon SNS 通知を使用する-ジョブの完了後に S3 Glacier が通知を送信する トピックを指定できます。ジョブのリクエストごとに SNS トピックを指定できます。S3 Glacier がジョブを完了した後にのみ、通知が送信されます。ジョブのリクエストごとに SNS トピックを指定することに加え、すべての取得に対してジョブの通知が送信されるように、ボールトにボールト通知を設定することもできます。詳細については、「ボールトの通知設定の指定 (PUT notification-configuration)」を参照してください。
- ジョブの詳細を取得する-ジョブの進行中に ジョブの説明 (GET JobID) リクエストを実行して、ジョブのステータス情報を取得することができます。ただし、ジョブの完了を確認するには、Amazon SNS 通知を使用する方が効率的です。

> **Note**
>
> 通知により取得する情報は、ジョブの説明 (GET JobID) を呼び出して取得する情報と同じです。

特定のイベントに対して、ボールトの通知設定を追加し、さらに、ジョブの開始リクエストで SNS トピックを指定した場合、S3 Glacier は両方の通知を送信します。詳細については、「ボールトの通知設定の指定 (PUT notification-configuration)」を参照してください。

ボールトインベントリ

S3 Glacier はボールトインベントリを約 1 日 1 回のペースで更新します。この更新は、アーカイブがボールトに最初にアップロードされた日から開始します。最後のインベントリ以降に、ボールトに対してアーカイブの追加や削除が行われていない場合、インベントリの日付は更新されません。ボールトインベントリに対してジョブを開始すると、S3 Glacier により最後に生成されたインベントリが返されます。そのインベントリはポイントインタイムのスナップショットであり、リアルタイムのデータではありません。

S3 Glacier によりボールトに対して最初に作成されるインベントリは、取得できるようになるまで、通常半日から最大 1 日かかります。

アーカイブをアップロードするごとにボールトインベントリを取得することは、あまり便利には感じられないかもしれません。しかし、S3 Glacier にアップロードしたアーカイブに関するメタデータに関連付けられたデータベースをクライアント側で管理する場合を考えてみてください。そのような場合には、実際のボールトインベントリとデータベース内の情報とを必要に応じて照合できるため、ボールトインベントリの利便性が実感できるものと思われます。インベントリジョブの出力で返されるデータフィールドの詳細については、「レスポンス本文」を参照してください。

インベントリの取得の範囲

アーカイブの作成日でフィルタするか、制限を設定することによって、取得されるインベントリの項目数を制限できます。

アーカイブの作成日によるフィルタリング

次の間に作成されたアーカイブのインベントリアイテムを取得できます。StartDateそしてEndDateこれらのパラメータの値をジョブの開始リクエスト。StartDate 以降から EndDate までに作成されたアーカイブが返されます。StartDate を指定せずに、EndDate のみを指定した場合、StartDate 以降に作成されたすべてのアーカイブのインベントリを取得します。EndDate を指定せずに、StartDate のみを指定した場合、EndDate までに作成されたすべてのアーカイブのインベントリを取得します。

インベントリ項目の取得ごとの制限

以下を設定することによって、返品されるインベントリの項目数を制限できます。Limitパラメータのジョブの開始リクエスト。インベントリジョブの出力には、Limit に指定した最大数のインベントリ項目が含まれます。利用可能なインベントリ項目がまだ存在する場合、結果がページ分割されます。ジョブの完了後、ジョブの説明 (GET JobID) オペレーションを使用して、次のジョブ開始リクエストで使用するマーカーを取得することができます。マーカーは、次に取得するインベントリ項目のセットの開始点を示します。繰り返し作成することで、インベントリ全体をページスルーできます。ジョブの開始前のマーカーを使用したリクエストジョブの説明出力。インベントリ項目がこれ以上存在しないことを示す、ジョブの説明のマーカーに null が返されるまでこれを行います。

Limit パラメータは日付範囲のパラメータとともに使用できます。

アーカイブの取得範囲

アーカイブ全体またはアーカイブの特定の範囲に対して、アーカイブの取得を開始することができます。アーカイブの取得範囲を指定する場合、返されるバイト範囲か、アーカイブ全体を指定できま

す。指定する範囲は、メガバイト (MB) 単位に調整されている必要があります。つまり、範囲の開始値は 1 MB で割り切れる値で、範囲の終了値は 1 を足すと 1 MB で割り切れる値またはアーカイブの終了値と同じである必要があります。アーカイブの取得範囲がメガバイト単位に調整されていない場合、このオペレーションではレスポンスで 400 が返されます。さらに、ジョブの出力の取得 (ジョブの出力の取得 (GET output)) を使用して、ダウンロードするデータのチェックサムの値を取得するには、指定する範囲が木構造ハッシュ可能になっている必要があります。木構造ハッシュ可能な範囲の詳細については、「データをダウンロードするときのチェックサムの受信」を参照してください。

迅速、標準、および大容量階層

アーカイブの取得ジョブを開始する際、リクエストボディの Tier フィールドに以下のオプションのいずれかを指定できます。

- **Expedited** - 迅速では、アーカイブの復元についての緊急リクエストが臨時に必要になった場合に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 (250 MB 以上) を除くすべてのアーカイブについては、迅速階層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。
- **Standard** - 標準では、数時間以内にすべてのアーカイブにアクセスできます。標準階層を使用してアクセスしたデータは、通常 3-5 時間以内に利用できるようになります。このオプションは、階層オプションを指定しないでジョブリクエストを行った場合のデフォルトです。
- **Bulk** - バルクはS3 Glacierの最低コスト層であり、1日で大量のデータ (ペタバイト) を安価に取得できるようにします。大容量階層を使用してアクセスしたデータは、通常 5-12 時間以内に利用できるようになります。

迅速取り出しと大容量取り出しの詳細については、「S3 Glacier アーカイブの取得」を参照してください。

リクエスト

ジョブを開始するには、HTTP POST メソッドを使用し、ボールトの jobs サブリソースをリクエストの範囲として指定します。リクエストの JSON ドキュメントでジョブのリクエストの詳細を指定します。ジョブのタイプは Type フィールドで指定します。S3 Glacier がジョブの完了後に通知を送信する Amazon SNS トピックを指定する SNSTopic フィールドをオプションで指定できます。

> **Note**
>
> Amazon SNS に通知を送信するには、トピックが存在しない場合は新しくトピックを作成する必要があります。 ではトピックを自動的に作成しません。S3 Glacier はトピックを作成しません。トピックには、S3 Glacier ボールトからパブリケーションを受け取るためのアクセ

ス許可が必要です。S3 Glacier では、トピックへのパブリッシュのアクセス許可がボールトにあるかどうかは確認しません。アクセス許可が適切に設定されていない場合、ジョブが完了しても、通知が送信されない可能性があります。

構文

以下に示しているのは、ジョブの開始に対するリクエスト構文です。

```
POST /AccountId/vaults/VaultName/jobs HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01

{
   "jobParameters": {
      "ArchiveId": "string",
      "Description": "string",
      "Format": "string",
      "InventoryRetrievalParameters": {
         "EndDate": "string",
         "Limit": "string",
         "Marker": "string",
         "StartDate": "string"
      },
      "OutputLocation": {
         "S3": {
            "AccessControlList": [
               {
                  "Grantee": {
                     "DisplayName": "string",
                     "EmailAddress": "string",
                     "ID": "string",
                     "Type": "string",
                     "URI": "string"
                  },
                  "Permission": "string"
               }
            ],
            "BucketName": "string",
            "CannedACL": "string",
            "Encryption": {
```

```
                "EncryptionType": "string",
                "KMSContext": "string",
                "KMSKeyId": "string"
            },
            "Prefix": "string",
            "StorageClass": "string",
            "Tagging": {
                "string" : "string"
            },
            "UserMetadata": {
                "string" : "string"
            }
        }
    },
    "RetrievalByteRange": "string",
    "SelectParameters": {
        "Expression": "string",
        "ExpressionType": "string",
        "InputSerialization": {
            "csv": {
                "Comments": "string",
                "FieldDelimiter": "string",
                "FileHeaderInfo": "string",
                "QuoteCharacter": "string",
                "QuoteEscapeCharacter": "string",
                "RecordDelimiter": "string"
            }
        },
        "OutputSerialization": {
            "csv": {
                "FieldDelimiter": "string",
                "QuoteCharacter": "string",
                "QuoteEscapeCharacter": "string",
                "QuoteFields": "string",
                "RecordDelimiter": "string"
            }
        }
    },
    "SNSTopic": "string",
    "Tier": "string",
    "Type": "string"
  }
}
```

> **Note**
>
> `AccountId` 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエスト本文

リクエストは、リクエストの本文にある JSON 形式で以下のデータを受け入れます。

jobParameters

ジョブ情報を指定するためのオプションを提供します。

タイプ: jobParameters オブジェクト

必須: はい

レスポンス

S3 Glacier がジョブを作成します。レスポンスでは、ジョブの URI が返されます。

構文

```
HTTP/1.1 202 Accepted
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: location
x-amz-job-id: jobId
x-amz-job-output-path: jobOutputPath
```

レスポンスヘッダー

ヘッダー	説明
`Location`	

ヘッダー	説明
	ジョブの相対 URI パス。この URI パスを使用して、ジョブのステータスを確認することができます。詳細については、「ジョブの説明 (GET JobID)」を参照してください。 タイプ: 文字列 デフォルト: なし
`x-amz-job-id`	ジョブの ID。この値も `Location` ヘッダーの一部として含まれます。 タイプ: 文字列 デフォルト: なし
`x-amz-job-output-path`	Select の結果が保存されている場所へのパス。 タイプ: 文字列 デフォルト: なし

レスポンス本文

このオペレーションはレスポンス本文を返しません。

エラー

この操作には、すべての Amazon S3 Glacier オペレーションに共通する可能性のあるエラーに加えて、次のエラーが含まれます。Amazon S3 Glacier のエラーとエラーコードのリストについては、「」を参照してください。エラーレスポンス。

コード	説明	HTTP ステータスコード	タイプ
`InsufficientCapacityException`	この迅速リクエストを処理する容量が足りない場合に返されます。このエラーは、迅速取り出しにの	`503 Service Unavailable`	[Server] (サーバー)

コード	説明	HTTP ステータスコード	タイプ
	み該当し、標準取り出しまたは大容量取り出しには該当しません。		

例

リクエストの例: アーカイブ取得ジョブを開始する

```
POST /-/vaults/examplevault/jobs HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
  "Type": "archive-retrieval",
  "ArchiveId": "NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
  "Description": "My archive description",
  "SNSTopic": "arn:aws:sns:us-west-2:111111111111:Glacier-ArchiveRetrieval-topic-
Example",
  "Tier" : "Bulk"
}
```

以下は、RetrievalByteRange フィールドを使用して、取得するアーカイブの範囲を指定するリクエストの本文の例です。

```
{
  "Type": "archive-retrieval",
  "ArchiveId": "NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchi
  "Description": "My archive description",
  "RetrievalByteRange": "2097152-4194303",
  "SNSTopic": "arn:aws:sns:us-west-2:111111111111:Glacier-ArchiveRetrieval-topic-
Example",
  "Tier" : "Bulk"
```

```
}
```

レスポンスの例

```
HTTP/1.1 202 Accepted
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Location: /111122223333/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
x-amz-job-id: HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

リクエストの例: インベントリ取得ジョブを開始する

以下のリクエストでは、インベントリ取得ジョブを開始して、examplevault ボールトからアーカイブのリストを取得します。リクエストボディで Format が CSV に設定されています。これは、インベントリが CSV 形式で返されることを示しています。

```
POST /-/vaults/examplevault/jobs HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Content-Type: application/x-www-form-urlencoded
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
  "Type": "inventory-retrieval",
  "Description": "My inventory job",
  "Format": "CSV",
  "SNSTopic": "arn:aws:sns:us-west-2:111111111111:Glacier-InventoryRetrieval-topic-
Example"
}
```

レスポンスの例

```
HTTP/1.1 202 Accepted
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Location: /111122223333/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

```
x-amz-job-id: HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

リクエストの例: 日付によるフィルタリングを使用し、制限を設定してインベントリの取得ジョブを開始し、後続のリクエストでインベントリ項目の次のページを取得する。

次のリクエストでは、日付によるフィルタリングを使用し、制限を設定して、ボールトインベントリ取得ジョブを開始します。

```
{
    "ArchiveId": null,
    "Description": null,
    "Format": "CSV",
    "RetrievalByteRange": null,
    "SNSTopic": null,
    "Type": "inventory-retrieval",
    "InventoryRetrievalParameters": {
        "StartDate": "2013-12-04T21:25:42Z",
        "EndDate": "2013-12-05T21:25:42Z",
        "Limit" : "10000"
    },
}
```

以下のリクエストは、ジョブの説明 (GET JobID) で取得したマーカーを使用して、インベントリ項目の次のページを取得する後続のリクエストの例です。

```
{
    "ArchiveId": null,
    "Description": null,
    "Format": "CSV",
    "RetrievalByteRange": null,
    "SNSTopic": null,
    "Type": "inventory-retrieval",
    "InventoryRetrievalParameters": {
        "StartDate": "2013-12-04T21:25:42Z",
        "EndDate": "2013-12-05T21:25:42Z",
        "Limit": "10000",
        "Marker":
 "vyS0t2jHQe5qbcDggIeD50chS1SXwYMrkVKo0KHiTUjEYxBGCqRLKaiySzdN7QXGVVV5XZpNVG67pCZ_uykQXFMLaxOSu
    },
}
```

レスポンスの例

```
HTTP/1.1 202 Accepted
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Location: /111122223333/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
x-amz-job-id: HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
x-amz-job-output-path: test/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID/
```

関連するセクション

- ジョブの説明 (GET JobID)
- ジョブの出力の取得 (GET output)
- Amazon S3 Glacier の ID とアクセス管理

ジョブのリスト表示 (GET jobs)

説明

このオペレーションは、進行中および最近終了したジョブを含む、ボールトに対するジョブを一覧表示します。

> **Note**
>
> Amazon S3 Glacier (S3 Glacier) では、最近完了したジョブを削除する前に一定期間保持しますが、最終的には完了したジョブを削除します。完了したジョブの出力を取得できます。ジョブは完了後に一定期間保持されるため、ジョブの完了通知を見落とした場合や、最初のダウンロードの試行が失敗した場合に、ジョブの出力を取得できます。たとえば、アーカイブをダウンロードするために、アーカイブの取得ジョブを開始したとします。ジョブの完了後、そのアーカイブのダウンロードを開始しましたが、ネットワークエラーが発生しました。このシナリオでは、ジョブが存在する限り、再試行によりそのアーカイブをダウンロードできます。

List Jobs オペレーションは、ページ分割をサポートしています。常にレスポンスの Marker フィールドを確認する必要があります。それ以上表示するジョブがなくなると、Marker フィールドは null に設定されます。リストするジョブがまだある場合、Marker フィールドは Null 以外の値に設定され、これを使用してリストのページ分割を続行できます。特定のジョブで開始されるジョブのリストを返すには、marker リクエストパラメータを、前の Marker リクエストから取得した、そのジョブの List Jobs 値に設定します。

リクエストで limit パラメータを指定して、レスポンスで返されるジョブ数の最大の制限を設定することができます。デフォルトの上限は 50 です。返されるジョブの数は、上限を下回ることはあっても、上限を上回ることはありません。

さらに、オプションの statuscode パラメータと completed パラメータのいずれかまたは両方を指定することで、返されるジョブリストをフィルタすることができます。statuscode パラメータでは、InProgress、Succeeded、Failed のいずれかのステータスと一致するジョブのみを返すように指定できます。completed パラメータでは、完了済みのジョブ (true) または未完了のジョブ (false) のみを返すように指定できます。

リクエスト

構文

各タイプのジョブのリストを返すには、ボールトの GET サブリソースの URI に jobs リクエストを送信します。

```
GET /AccountId/vaults/VaultName/jobs HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は、ボールトを所有するアカウントの AWS アカウント ID です。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。アカウント ID を使用する場合は、ID にハイフン ('-') を含めないでください。

リクエストパラメータ

名前	説明	必須
completed	返すジョブの状態。true または false を指定できます。 型: ブール値 制約: なし	いいえ
limit	返されるジョブの最大数。デフォルトの上限は 50 です。返されるジョブの数は、指定した上限を下回ることはあっても、上限を上回ることはありません。 タイプ: 文字列 制約: 最小の整数値は 1 です。最大の整数値は 50 です。	いいえ
marker	ジョブのリスト表示を開始するジョブを指定する、ページ分割に使用される不透明な文字列。marker 値は、前の List Jobs レスポンスから取得します。前の marker リクエストで開始された結果のページ分割を継続する場合は、List Jobs を含めるだけで構いません。 タイプ: 文字列 制約: なし	いいえ
statuscode	返すジョブのステータスのタイプ。 タイプ: 文字列 制約: InProgress 、Succeeded 、または Failed のいずれかの値。	いいえ

リクエストヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「共通のレスポンスヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
Content-Type: application/json
Content-Length: Length

{
    "JobList": [
        {
            "Action": "string",
            "ArchiveId": "string",
            "ArchiveSHA256TreeHash": "string",
            "ArchiveSizeInBytes": number,
            "Completed": boolean,
            "CompletionDate": "string",
            "CreationDate": "string",
            "InventoryRetrievalParameters": {
                "EndDate": "string",
                "Format": "string",
                "Limit": "string",
                "Marker": "string",
                "StartDate": "string"
            },
            "InventorySizeInBytes": number,
            "JobDescription": "string",
            "JobId": "string",
            "JobOutputPath": "string",
            "OutputLocation": {
                "S3": {
                    "AccessControlList": [
```

```
                    {
                        "Grantee": {
                            "DisplayName": "string",
                            "EmailAddress": "string",
                            "ID": "string",
                            "Type": "string",
                            "URI": "string"
                        },
                        "Permission": "string"
                    }
                ],
                "BucketName": "string",
                "CannedACL": "string",
                "Encryption": {
                    "EncryptionType": "string",
                    "KMSContext": "string",
                    "KMSKeyId": "string"
                },
                "Prefix": "string",
                "StorageClass": "string",
                "Tagging": {
                    "string": "string"
                },
                "UserMetadata": {
                    "string": "string"
                }
            }
        },
        "RetrievalByteRange": "string",
        "SelectParameters": {
            "Expression": "string",
            "ExpressionType": "string",
            "InputSerialization": {
                "csv": {
                    "Comments": "string",
                    "FieldDelimiter": "string",
                    "FileHeaderInfo": "string",
                    "QuoteCharacter": "string",
                    "QuoteEscapeCharacter": "string",
                    "RecordDelimiter": "string"
                }
            },
            "OutputSerialization": {
                "csv": {
```

```
                        "FieldDelimiter": "string",
                        "QuoteCharacter": "string",
                        "QuoteEscapeCharacter": "string",
                        "QuoteFields": "string",
                        "RecordDelimiter": "string"
                    }
                }
            },
            "SHA256TreeHash": "string",
            "SNSTopic": "string",
            "StatusCode": "string",
            "StatusMessage": "string",
            "Tier": "string",
            "VaultARN": "string"
        }
    ],
    "Marker": "string"
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

レスポンス本文には次の JSON フィールドが含まれています。

JobList

ジョブオブジェクトのリスト。各ジョブオブジェクトには、そのジョブを説明するメタデータが含まれます。

型: GlacierJobDescription オブジェクトの配列

Marker

結果のページ分割をどこから継続するかを表す不透明な文字列。リストに含まれるジョブをさらに取得するには、新しいmarker リクエストで `List Jobs` 値を使用します。リスト表示するジョブがそれ以上存在しない場合、この値は `null` です。

型: 文字列

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

以下の例は、ボールトジョブに関する情報を返す方法を示しています。最初の例では 2 つのジョブのリストを返し、2 番目の例ではジョブのサブセットを返します。

例: すべてのジョブを返す

リクエストの例

次の GET リクエストでは、ボールトのジョブを返します。

```
GET /-/vaults/examplevault/jobs  HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

次のレスポンスには、アーカイブの取得ジョブおよびインベントリの取得ジョブが含まれています。インベントリの取得ジョブには、ボールトインベントリの取得のページ分割を継続する場合に使用するマーカーが含まれています。レスポンスでは、`Marker` フィールドが `null` に設定され、表示するジョブがそれ以上ないことを示しています。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 1444

{
  "JobList": [
    {
      "Action": "ArchiveRetrieval",
      "ArchiveId": "BDfaUQul0dVzYwAMr8YSa_6_8abbhZq-
i1oT69g8ByClfJyBgAGBkWl2QbF5os851P7Y7KdZDOHWJIn4rh1ZHaOYD3MgFhK_g0oDPesW34uHQoVGwoIqubf6BgUEfQm
```

```
      "ArchiveSizeInBytes": 1048576,
      "ArchiveSHA256TreeHash":
 "25499381569ab2f85e1fd0eb93c5406a178ab77c5933056eb5d6e7d4adda609b",
      "Completed": true,
      "CompletionDate": "2012-05-01T00:00:09.304Z",
      "CreationDate": "2012-05-01T00:00:06.663Z",
      "InventorySizeInBytes": null,
      "JobDescription": null,
      "JobId": "hDe9t9DTHXqFw8sBGpLQQOmIM0-
JrGtu1O_YFKLnzQ64548qJc667BRWTwBLZC76Ygy1jHYruqXkdcAhRsh0hYv4eVRU",
      "RetrievalByteRange": "0-1048575",
      "SHA256TreeHash":
 "25499381569ab2f85e1fd0eb93c5406a178ab77c5933056eb5d6e7d4adda609b",
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
      "Tier": "Bulk",
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    },
    {
      "Action": "InventoryRetrieval",
      "ArchiveId": null,
      "ArchiveSizeInBytes": null,
      "ArchiveSHA256TreeHash": null,
      "Completed": true,
      "CompletionDate": "2013-05-11T00:25:18.831Z",
      "CreationDate": "2013-05-11T00:25:14.981Z",
      "InventorySizeInBytes": 1988,
      "JobDescription": null,
      "JobId":
 "2cvVOnBL36btzyP3pobwIceiaJebM1bx9vZOOUtmNAr0KaVZ4WkWgVjiPldJ73VU7imlm0pnZriBVBebnqaAcirZq_C5"
      "RetrievalByteRange": null,
      "SHA256TreeHash": null,
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
      "InventoryRetrievalParameters": {
          "StartDate": "2013-11-12T13:43:12Z",
          "EndDate": "2013-11-20T08:12:45Z",
          "Limit": "120000",
          "Format": "JSON",
          "Marker":
 "vyS0t2jHQe5qbcDggIeD50chS1SXwYMrkVKo0KHiTUjEYxBGCqRLKaiySzdN7QXGVVV5XZpNVG67pCZ_uykQXFMLaxOSu
```

```
    }
  ],
  "Marker": null
}
```

例: ジョブの部分的なリストを返す

リクエストの例

次の GET リクエストでは、marker パラメータで指定されたジョブを返します。limit パラメータを に設定すると、最大 2 個のジョブを返すように指定されます。2

```
GET /-/vaults/examplevault/jobs?marker=HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID&limit=2
 HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

次のレスポンスは、返される 2 つのジョブと、ジョブリストのページ分割を続行するために使用できる Null 以外の値に設定された Marker フィールドを示しています。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 1744

{
  "JobList": [
    {
      "Action": "ArchiveRetrieval",
      "ArchiveId": "58-3KpZfcMPUznvMZNPaKyJx9wODCsWTnqcjtx2CjKZ6b-
XgxEuA8yvZOYTPQfd7gWR4GRm2XR08gcnWbLV4VPV_kDWtZJKi0TFhKKVPzwrZnA4-
FXuIBfViYUIVveeiBE51FO4bvg",
      "ArchiveSizeInBytes": 8388608,
      "ArchiveSHA256TreeHash":
 "106086b256ddf0fedf3d9e72f461d5983a2566247ebe7e1949246bc61359b4f4",
```

```
      "Completed": true,
      "CompletionDate": "2012-05-01T00:25:20.043Z",
      "CreationDate": "2012-05-01T00:25:16.344Z",
      "InventorySizeInBytes": null,
      "JobDescription": "aaabbbccc",
      "JobId": "s4MvaNHIh6mOa1f8iY4ioG2921SDPihXxh3Kv0FBX-
JbNPctpRvE4c2_BifuhdGLqEhGBNGeB6Ub-JMunR9JoVa8y1hQ",
      "RetrievalByteRange": "0-8388607",
      "SHA256TreeHash":
 "106086b256ddf0fedf3d9e72f461d5983a2566247ebe7e1949246bc61359b4f4",
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
      "Tier": "Bulk",
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    },
    {
      "Action": "ArchiveRetrieval",
      "ArchiveId": "2NVGpf83U6qB9M2u-
Ihh61yoFLRDEoh7YLZWKBn80A2i1xG8uieBwGjAr4RkzOHA0E07ZjtI267R03Z-6Hxd8pyGQkBdciCSH1-
Lw63Kx9qKpZbPCdU0uTW_WAdwF6lR6w8iSyKdvw",
      "ArchiveSizeInBytes": 1048576,
      "ArchiveSHA256TreeHash":
 "3d2ae052b2978727e0c51c0a5e32961c6a56650d1f2e4ceccab6472a5ed4a0",
      "Completed": true,
      "CompletionDate": "2012-05-01T16:59:48.444Z",
      "CreationDate": "2012-05-01T16:59:42.977Z",
      "InventorySizeInBytes": null,
      "JobDescription": "aaabbbccc",
      "JobId":
 "CQ_tf6fOR4jrJCL61Mfk6VM03oY8lmnWK93KK4gLig1UPAbZiN3UV4G_5nq4AfmJHQ_dOMLOX5k8ItFv0wCPN0oaz5dG"
      "RetrievalByteRange": "0-1048575",
      "SHA256TreeHash":
 "3d2ae052b2978727e0c51c0a5e32961c6a56650d1f2e4ceccab6472a5ed4a0",
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
      "Tier": "Standard",
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/examplevault"
    }
  ],
  "Marker":
 "CQ_tf6fOR4jrJCL61Mfk6VM03oY8lmnWK93KK4gLig1UPAbZiN3UV4G_5nq4AfmJHQ_dOMLOX5k8ItFv0wCPN0oaz5dG"
```

```
}
```

関連するセクション

- ジョブの説明 (GET JobID)
- Amazon S3 Glacier の ID とアクセス管理

ジョブオペレーションで使用されるデータ型

以下は、S3 Glacier のジョブオペレーションと使用できるデータ型です。

トピック

- CSVInput
- CSVOutput
- Encryption
- GlacierJobDescription
- 権限
- 被付与者
- InputSerialization
- InventoryRetrievalJobInput
- jobParameters
- OutputLocation
- OutputSerialization
- S3Location
- SelectParameters

CSVInput

カンマ区切り値 (CSV) ファイルについての情報が含まれています。

内容

コメント

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。

型: 文字列

必須: いいえ

FieldDelimiter

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。その文字は、`\n`、`\r`、または 32-126 の範囲の ASCII 文字である必要があります。デフォルトではカンマ (`,`) を使用します。

型: 文字列

デフォルト: ,

必須: いいえ

FileHeaderInfo

入力の 1 行目の処理方法を説明する値。

型: 文字列

有効な値: `Use` | `Ignore` | `None`

必須: いいえ

QuoteCharacter

フィールド区切り文字が値の一部であるエスケープ文字として使用される 1 文字。

型: 文字列

必須: いいえ

QuoteEscapeCharacter

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。

型: 文字列

必須: いいえ

RecordDelimiter

個々のレコードを区切るために使用される 1 文字。

型: 文字列

必須: いいえ

詳細情報

- ジョブの開始 (ジョブの POST)

CSVOutput

ジョブ結果が保存されているカンマ区切り値 (CSV) 形式についての情報が含まれています。

内容

FieldDelimiter

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。

型: 文字列

必須: いいえ

QuoteCharacter

フィールド区切り文字が値の一部であるエスケープ文字として使用される 1 文字。

型: 文字列

必須: いいえ

QuoteEscapeCharacter

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。

型: 文字列

必須: いいえ

QuoteFields

すべての出力フィールドが引用符内に含まれる必要があるかどうかを示す値。

有効な値: `ALWAYS` | `ASNEEDED`

型: 文字列

必須: いいえ

RecordDelimiter

個々のレコードを区切るために使用される 1 文字。

型: 文字列

必須: いいえ

詳細情報

- ジョブの開始 (ジョブの POST)

Encryption

Amazon S3 にジョブ結果を保存するのに使用された暗号化に関する情報が含まれます。

内容

Encryption

Amazon S3 への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。

型: 文字列

有効な値: `aws:kms` | `AES256`

必須: いいえ

KMSContext

オプション。暗号化タイプが `aws:kms`, である場合、この値を使用してジョブ結果の暗号化コンテキストを指定できます。

型: 文字列

必須: いいえ

KMSKeyId

オブジェクトの暗号化に使用する AWS Key Management Service（AWS KMS) キー ID。

型: 文字列

必須: いいえ

詳細情報

- ジョブの開始 (ジョブの POST)

GlacierJobDescription

Amazon S3 Glacier (S3 Glacier) ジョブの説明が含まれます。

内容

Action

ジョブのタイプ。`ArchiveRetrieval`、`InventoryRetrieval`、または `Select` です。

型: 文字列

ArchiveId

選択ジョブまたはアーカイブの取得ジョブにリクエストされたアーカイブ ID。それ以外の場合、このフィールドは `null` です。

型: 文字列

ArchiveSHA256TreeHash

アーカイブの取得を行うアーカイブ全体の SHA256 木構造ハッシュ。インベントリの取得ジョブの場合、このフィールドは `null` です。

型: 文字列

ArchiveSizeInBytes

`ArchiveRetrieval` ジョブの場合、これはダウンロードに必要なアーカイブのサイズ (バイト単位) です。`InventoryRetrieval` ジョブの場合、この値は `null` です。

タイプ: 数値

Completed

ジョブが完了済みの場合は `true`、それ以外の場合は `false`。

タイプ: ブール

CompletionDate

ジョブが完了した日付。

ジョブリクエストが完了した協定世界時 (UTC) 時間。ジョブが進行中の場合、値は null です。

タイプ: ISO 8601 の日付形式の文字列表現。たとえば`2013-03-20T17:03:43.221Z`。

CreationDate

ジョブを開始した協定世界時 (UTC) の日付。

タイプ: ISO 8601 の日付形式の文字列表現。たとえば`2013-03-20T17:03:43.221Z`。

InventoryRetrievalParameters

インベントリの取得の範囲に使用される入力パラメータ。

型: InventoryRetrievalJobInput オブジェクト

InventorySizeInBytes

`InventoryRetrieval` ジョブの場合、これはダウンロードに必要なインベントリのサイズ (バイト単位) です。`ArchiveRetrieval` または `Select` ジョブの場合、値は `null` です。

型: 数値

JobDescription

ジョブを開始したときに指定したジョブの説明。

型: 文字列

JobId

S3 Glacier 内のジョブを識別する ID。

型: 文字列

JobOutputPath

ジョブの出力場所が含まれます。

型: 文字列

OutputLocation

選択ジョブの結果とエラーが保存されている場所についての情報を含むオブジェクト。

型: OutputLocation オブジェクト

RetrievalByteRange

"*StartByteValue-EndByteValue*" という形式で示される、アーカイブの取得ジョブで取得したバイト範囲。アーカイブの取得で範囲を指定しなかった場合は、アーカイブ全体が取得され、StartByteValue は 0、EndByteValue はアーカイブのサイズから 1 を引いた値になります。インベントリの取得ジョブの場合、このフィールドは `null` です。

型: 文字列

SelectParameters

選択に使用されるパラメータに関する情報を含むオブジェクト。

型: SelectParameters オブジェクト

SHA256TreeHash

アーカイブのリクエストされた範囲の SHA256 木構造ハッシュ値。アーカイブのジョブの開始 (ジョブの POST)リクエストで木構造ハッシュ可能な範囲を指定した場合、このフィールドに値が返されます。アーカイブの範囲取得で木構造ハッシュを可能にするための調整の詳細については、「データをダウンロードするときのチェックサムの受信」を参照してください。

アーカイブ全体を取得する特別な場合は、この値は `ArchiveSHA256TreeHash` の値と同じです。

次の場合、このフィールドは `null` です。

- 木構造ハッシュ可能ではない範囲を指定したアーカイブの取得ジョブ。

- アーカイブ全体を範囲に指定したアーカイブのジョブで、ジョブのステータスが `InProgress` の場合。
- インベントリジョブ。
- 選択ジョブ。

型: 文字列

SNSTopic

ジョブの開始 (ジョブの開始 (ジョブの POST)) で通知を設定した場合に、ジョブの完了または失敗の通知が送信される Amazon SNS トピックを表す Amazon リソースネーム (ARN)。

型: 文字列

StatusCode

ジョブのステータスを示すコード。

有効な値: `InProgress` | `Succeeded` | `Failed`

型: 文字列

StatusMessage

ジョブのステータスメッセージ。

型: 文字列

Tier

選択またはアーカイブの取得に使用するデータアクセス層。

有効な値: `Expedited` | `Standard` | `Bulk`

型: 文字列

VaultARN

ジョブがサブリソースとなるボールトの ARN。

型: 文字列

詳細情報

- ジョブの開始 (ジョブの POST)

権限

許可についての情報が含まれています。

内容

被付与者

被付与者。

タイプ: 被付与者 オブジェクト

必須: いいえ

アクセス権限

被付与者に与えられるアクセス許可。

型: 文字列

有効な値: `FULL_CONTROL` | `WRITE` | `WRITE_ACP` | `READ` | `READ_ACP`

必須: いいえ

詳細情報

- ジョブの開始 (ジョブの POST)

被付与者

被付与者についての情報が含まれています。

内容

DisplayName

被付与者のスクリーンネーム。

型: 文字列

必須: いいえ

EmailAddress

被付与者の E メールアドレス。

型: 文字列

必須: いいえ

ID

被付与者の正規ユーザー ID。

型: 文字列

必須: いいえ

Type

被付与者のタイプ。

型: 文字列

有効な値: `AmazonCustomerByEmail` | `CanonicalUser` | `Group`

必須: いいえ

URI

被付与者グループの URI。

型: 文字列

必須: いいえ

詳細情報

- ジョブの開始 (ジョブの POST)

InputSerialization

アーカイブがシリアル化される方法について記述します。

内容

CSV

CSV でエンコードされたオブジェクトのシリアル化を記述するオブジェクト。

タイプ: CSVInput オブジェクト

必須: いいえ

詳細情報

- ジョブの開始 (ジョブの POST)

InventoryRetrievalJobInput

インベントリの取得の範囲ジョブを指定するためのオプションを提供します。

内容

EndDate

ボールトインベントリを取得した日付範囲の終了時 (UTC 時間)。この日付よりも前に作成されたアーカイブが含まれます。

有効な値: ISO 8601 日付形式の文字列表現 (`YYYY-MM-DDThh:mm:ssTZD`)。例えば、`2013-03-20T17:03:43Z` のようになります。

型: 文字列 ISO 8601 日付形式の文字列表現 (`YYYY-MM-DDThh:mm:ssTZD`)。例えば、`2013-03-20T17:03:43Z` のようになります。

必須: いいえ

Format

ボールトインベントリのリストの出力形式。ボールトインベントリを取得するジョブを開始するときに、ジョブの開始 (ジョブの POST) リクエストによって設定されます。

有効な値: `CSV | JSON`

必須: いいえ

型: 文字列

Limit

ボールトインベントリの取得リクエストごとに返すことができるインベントリ項目の最大数。

有効な値: 1 以上の整数値。

型: 文字列

必須: いいえ

Marker

ボールトインベントリの取得結果のページ分割をどこから継続するかを表す不透明な文字列。インベントリ項目を追加で取得するには、新しい `Initiate Job` リクエストでこのマーカーを使用します。インベントリ項目がそれ以上存在しない場合、この値は null です。

型: 文字列

必須: いいえ

StartDate

ボールトインベントリを取得した日付範囲の開始日 (UTC 時間)。この日付以降に作成されたアーカイブが含まれます。

有効な値: ISO 8601 日付形式の文字列表現 (`YYYY-MM-DDThh:mm:ssTZD`)、例:`2013-03-20T17:03:43Z`。

型: 文字列 ISO 8601 日付形式の文字列表現 (`YYYY-MM-DDThh:mm:ssTZD`)。例えば、`2013-03-20T17:03:43Z` のようになります。

必須: いいえ

詳細情報

- ジョブの開始 (ジョブの POST)

jobParameters

ジョブを定義するためのオプションを提供します。

内容

ArchiveId

必要なアーカイブの ID。このフィールドは、`Type` フィールドが `select` または `archive-retrieval` に設定されている場合に必要です。インベントリ取得ジョブのリクエストでこのフィールドを指定すると、エラーが発生します。

有効な値: Amazon S3 Glacier (S3 Glacier) に対する以前のリクエストで取得した有効なアーカイブ ID である必要があります。

型: 文字列

必須: はい (`Type` が `select` または `archive-retrieval` に設定されている場合)

説明

ジョブの任意の説明。

有効な値: 説明は 1,024 バイト以下である必要があります。使用可能な文字は、制御コードを除く 7 ビット ASCII コードです。具体的には、32-126 (10 進) または 0x20-0x7E (16 進) の ASCII 値です。

型: 文字列

必須: いいえ

Format

(オプション) ボールトインベントリを取得するジョブを開始する場合の出力形式。インベントリジョブを開始する際に、`Format` フィールドを指定しなかった場合、デフォルトの形式は JSON です。

有効な値: `CSV` | `JSON`

型: 文字列

必須: いいえ

InventoryRetrievalParameters

インベントリの取得の範囲に使用される入力パラメータ。

型: InventoryRetrievalJobInput オブジェクト

必須: いいえ

OutputLocation

選択ジョブの結果が保存されている場所についての情報を含むオブジェクト。

型: OutputLocation オブジェクト

必須: はい (`select` ジョブの場合)

RetrievalByteRange

取得するバイト範囲`archive-retrieval`、という形式にします。*`StartByteValue`*-*`EndByteValue`*「。このフィールドが指定されていない場合、アーカイブ全体が取得されます。このフィールドを指定する場合、バイト範囲はメガバイト単位 (1024 x 1024) に調整されている必要があります。メガバイト単位に調整するとは、StartByteValue は 1 MB で割り切れる値、EndByteValue は 1 を足すと 1 MB で割り切れる値またはアーカイブの終了値 (アーカイブのバイトサイズの値から 1 を引いた値) に等しい値である必要があるということです。RetrievalByteRange がメガバイト単位に調整されていない場合、このオペレーションではレスポンスで `400` が返されます。

`inventory-retrieval` または `select` ジョブのリクエストでこのフィールドを指定すると、エラーが発生します。

型: 文字列

必須: いいえ

SelectParameters

選択に使用されるパラメータに関する情報を含むオブジェクト。

型: SelectParameters オブジェクト

必須: いいえ

SNSTopic

ジョブが完了し、出力をダウンロードする準備ができたときに S3 Glacier が通知を送信する Amazon SNS トピックの Amazon リソースネーム (ARN)。指定したトピックから受信者に通知が発行されます。

SNS トピックは存在している必要があります。存在しない場合、S3 Glacier が自動的にトピックを作成することはありません。さらに、SNS トピックには、ジョブを作成したアカウントでト

ピックにメッセージを発行することを許可するポリシーが必要です。SNS トピック名については、を参照してください。CreateTopicのAmazon Simple Notification Service API リファレンス。

型: 文字列

必須: いいえ

Tier

選択またはアーカイブの取得ジョブに使用する層。 `Standard` は使用されるデフォルト値です。

有効な値: `Expedited` | `Standard` | `Bulk`

型: 文字列

必須: いいえ

Type

ジョブのタイプ。アーカイブに SELECT クエリを実行し、アーカイブを取得し、ボールトインベントリを取得する ジョブを開始することができます。

有効な値: `select` | `archive-retrieval` | `inventory-retrieval`

型: 文字列

必須: はい

詳細情報

- ジョブの開始 (ジョブの POST)

OutputLocation

ジョブ結果およびエラーが保存される場所に関する情報が含まれます。

内容

S3

復元リクエストの結果を受け取る Amazon S3 の場所を記述するオブジェクト。

タイプ: S3Location

必須: はい

詳細情報

- ジョブの開始 (ジョブの POST)

OutputSerialization

出力がシリアル化される方法について記述します。

内容

CSV

カンマ区切り値 (CSV) でエンコードされたクエリ結果のシリアル化を記述するオブジェクト。

タイプ: CSVOutput オブジェクト

必須: いいえ

詳細情報

- ジョブの開始 (ジョブの POST)

S3Location

Amazon S3 内の、ジョブ結果が保存される場所に関する情報が含まれます。

内容

AccessControlList

保存した結果へのアクセスを制御する、許可のリスト。

型: 権限 オブジェクトの配列

必須: いいえ

BucketName

ジョブ結果が保存されている Amazon 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ボールトと同じ AWS リージョンに存在する必要があります。

型: 文字列

必須: はい

CannedACL

ジョブ結果に適用する、既定アクセスコントロールリスト (ACL)。

型: 文字列

有効な値: `private` | `public-read` | `public-read-write` | `aws-exec-read` | `authenticated-read` | `bucket-owner-read` | `bucket-owner-full-control`

必須: いいえ

Encryption

Amazon S3 にジョブ結果を保存するのに使用された、暗号化に関する情報を含むオブジェクト。

型: Encryption オブジェクト

必須: いいえ

Prefix

このリクエストの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。

型: 文字列

必須: はい

StorageClass

ジョブ結果を保存するために使用される、ストレージのクラス。

型: 文字列

有効な値: `STANDARD` | `REDUCED_REDUNDANCY` | `STANDARD_IA`

必須: いいえ

Tagging

ジョブ結果に適用されるタグセット。

型: 文字列から文字列へのマップ

必須: いいえ

UserMetadata

ジョブ結果とともに Amazon S3 に保存されるメタデータのマップ。

型: 文字列から文字列へのマップ

必須: いいえ

詳細情報

- ジョブの開始 (ジョブの POST)

SelectParameters

選択に使用されるパラメータに関する情報を含みます。

内容

Expression

オブジェクトの選択に使用される式。式は、128,000 文字のクォータを超えることはできません。

型: 文字列

必須: はい

ExpressionType

提供される式のタイプ (たとえば、SQL)。

有効な値: SQL

型: 文字列

必須: はい

InputSerialization

選択されたオブジェクトのシリアル化形式について記述します。

タイプ: InputSerialization オブジェクト

必須: いいえ

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。

必須: いいえ

タイプ: OutputSerialization オブジェクト

詳細情報

- ジョブの開始 (ジョブの POST)

データ取り出しオペレーション

S3 Glacier で使用できるデータ取得関連のオペレーションを次に示します。

トピック

- データ取り出しポリシーの取得 (ポリシーの GET)
- プロビジョニングされた容量を表示する (GET provisioned-capacity)
- プロビジョニングされた容量の購入 (POST provisioned-capacity)
- データ取り出しポリシーの設定 (ポリシーの PUT)

データ取り出しポリシーの取得 (ポリシーの GET)

説明

このオペレーションは、GETリクエストで指定された AWS アカウント および AWS リージョンの現在のデータ取得ポリシーを返します。データ取り出しポリシーの詳細については、「S3 Glacier データ取得ポリシー」を参照してください。

リクエスト

現在のデータ取り出しポリシーを返すには、以下の構文例に示しているように、データ取り出しポリシーの URI に HTTP GET リクエストを送信します。

構文

```
GET /AccountId/policies/data-retrieval HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> AccountId 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

構文

```
HTTP/1.1 200 OK
```

```
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length
{
  "Policy":
    {
      "Rules":[
         {
            "BytesPerHour": Number,
            "Strategy": String
         }
       ]
    }
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

レスポンス本文には次の JSON フィールドが含まれています。

BytesPerHour

1 時間あたりに取り出すことのできるデータの最大サイズ (バイト) 。

このフィールドが present になるのは、Strategy フィールドの値が `BytesPerHour` に設定されている場合のみです。

タイプ: 数値

ルール

ポリシールール。これは列挙型ですが、現在は 1 つのルールのみを設定します。Strategy フィールドと必要に応じて BytesPerHour フィールドを含みます。

型: 配列

方針

データ取り出しポリシーのタイプです。

型: 文字列

有効な値: `BytesPerHour|FreeTier|None`。`BytesPerHour` は、コンソールで 最大取得率 を選択することに相当します。`FreeTier` は、コンソールで 無料利用枠のみ を選択することに相当します。None は、コンソールで No Retrieval Policy (取り出しポリシーなし) を選択することに相当します。コンソールでのデータ取り出しポリシーの選択の詳細については、「S3 Glacier データ取得ポリシー」を参照してください。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

以下の例では、データ取り出しポリシーの取得方法を示しています。

リクエストの例

この例では、`GET` リクエストをデータ取り出しポリシーの URI に送信しています。

```
GET /-/policies/data-retrieval HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

正常なレスポンスでは、レスポンス本文に JSON 形式でデータ取り出しポリシーが示されます。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
Content-Type: application/json
Content-Length: 85

{
  "Policy":
```

```
    {
      "Rules":[
         {
           "BytesPerHour":10737418240,
           "Strategy":"BytesPerHour"
          }
       ]
    }
}
```

関連するセクション

- データ取り出しポリシーの設定 (ポリシーの PUT)
- ジョブの開始 (ジョブの POST)

プロビジョニングされた容量を表示する (GET provisioned-capacity)

このオペレーションでは、指定された AWS アカウントのプロビジョニングされた容量を表示します。プロビジョンドキャパシティーの詳細については、「アーカイブの取り出しオプション」を参照してください。

プロビジョニングされた容量単位は、購入日時 (開始日) から 1 か月間有効です。単位に有効期限に失効します。これは開始日から正確に 1 か月後であり、1 秒単位で四捨五入されます。

開始日が 31 日の場合、有効期限は翌月の最終日となります。たとえば、開始日が 8 月 31 日の場合、有効期限は 9 月 30 日です。開始日が 1 月 31 日の場合、有効期限は 2 月 28 日です。この機能は「レスポンスの例」で確認できます。

リクエストの構文

アカウントのプロビジョニングされた取得容量を表示するには、以下の構文例に示すように、provisioned-capacity URI に HTTP GET リクエストを送信します。

```
GET /AccountId/provisioned-capacity HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
```

```
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

オペレーションが成功した場合、サービスは HTTP レスポンス `200 OK` を返します。

レスポンスの構文

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: Length
{
   "ProvisionedCapacityList":
      {
         "CapacityId" : "string",
         "StartDate" : "string"
```

```
        "ExpirationDate" : "string"
    }
}
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

レスポンス本文には次の JSON フィールドが含まれています。

CapacityId

プロビジョニングされた容量単位を識別する ID。

型:: 文字列

StartDate

プロビジョニングされた容量単位を購入した協定世界時 (UTC) の日付。

型: 文字列 たとえば、ISO 8601 の日付形式の文字列表現。`2013-03-20T17:03:43.221Z`。

ExpirationDate

プロビジョニングされた容量単位が期限切れになった協定世界時 (UTC) の日付。

型: 文字列 たとえば、ISO 8601 の日付形式の文字列表現。`2013-03-20T17:03:43.221Z`。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

次の例は、アカウントのプロビジョニングされた容量単位を示します。

リクエストの例

この例では、GET リクエストを送信して、指定されたアカウントのプロビジョニングされた容量単位のリストを取得しています。

```
GET /123456789012/priority-capacity HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

レスポンスの例

リクエストが成功すると、次の例に示すように Amazon S3 Glacier (S3 Glacier) からアカウントのプロビジョニングされた容量単位とともに `HTTP 200 OK` が返されます。

最初にリストされているプロビジョニングされた容量単位は、開始日が 2017 年 1 月 31 日であり有効期限が 2017 年 2 月 28 日である単位の例です。前述のとおり、開始日が 31 日の場合、有効期限は翌月の最終日となります。

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
Content-Type: application/json
Content-Length: length

{
   "ProvisionedCapacityList",
      {
         "CapacityId": "zSaq7NzHFQDANTfQkDen4V7z",
         "StartDate": "2017-01-31T14:26:33.031Z",
         "ExpirationDate": "2017-02-28T14:26:33.000Z",
      },
      {
         "CapacityId": "yXaq7NzHFQNADTfQkDen4V7z",
         "StartDate": "2016-12-13T20:11:51.095Z"",
         "ExpirationDate": "2017-01-13T20:11:51.000Z" ",
      },
      ...
}
```

関連するセクション

- プロビジョニングされた容量の購入 (POST provisioned-capacity)

プロビジョニングされた容量の購入 (POST provisioned-capacity)

このオペレーションでは、 AWS アカウントのプロビジョニングされた容量単位を購入します。

プロビジョニングされた容量単位は、購入日時 (開始日) から 1 か月間有効です。単位に有効期限に失効します。これは開始日から正確に 1 か月後であり、1 秒単位で四捨五入されます。

開始日が 31 日の場合、有効期限は翌月の最終日となります。たとえば、開始日が 8 月 31 日の場合、有効期限は 9 月 30 日です。開始日が 1 月 31 日の場合、有効期限は 2 月 28 日です。

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。容量の各単位について 5 分ごとに 3 回以上の迅速取り出しを保証し、最大 150 MB/秒の取り出しスループットを提供します。プロビジョンドキャパシティーの詳細については、「アーカイブの取り出しオプション」を参照してください。

> **Note**
>
> プロビジョニングされたキャパシティーユニットは あたり 2 つに制限されています AWS アカウント。

リクエスト

のプロビジョンドキャパシティユニットを購入するには、プロビジョンドキャパシティ URI に HTTP POSTリクエスト AWS アカウント を送信します。

構文

```
POST /AccountId/provisioned-capacity HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId` 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier は

リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

リクエストパラメータ

リクエストヘッダー

このオペレーションは、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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

この操作にリクエストボディはありません。

レスポンス

オペレーションリクエストが成功した場合、サービスは HTTP 応答 `201 Created` を返します。

構文

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-capacity-id: CapacityId
```

レスポンスヘッダー

成功したレスポンスには、すべての操作に共通のレスポンスヘッダーに加えて、次のレスポンスヘッダーが含まれます。共通のレスポンスヘッダーの詳細については、「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-capacity-id`	プロビジョニングされた容量単位を識別する ID。 タイプ: 文字列

レスポンス本文

このオペレーションはレスポンス本文を返しません。

エラー

この操作には、すべての Amazon S3 Glacier オペレーションに共通する可能性のあるエラーに加えて、次のエラーが含まれます。Amazon S3 Glacier のエラーとエラーコードのリストについては、「」を参照してください。エラーレスポンス。

コード	説明	HTTP ステータスコード	タイプ
`LimitExceededException`	リクエストした容量が、アカウントでプロビジョニングされた容量単位に設定している上限を超えると、返されます。	`400 Bad Request`	クライアント

例

次の例では、アカウントのプロビジョニングされた容量を購入します。

リクエストの例

次の例では、HTTP POST リクエストを送信して、プロビジョニングされた容量単位を購入します。

```
POST /123456789012/provisioned-capacity HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
Content-Length: length
x-amz-glacier-version: 2012-06-01
```

レスポンスの例

リクエストが成功した場合、次の例に示すように、Amazon S3 Glacier (S3 Glacier) は `HTTP 201 Created` レスポンスを返します。

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

```
x-amz-capacity-id: zSaq7NzHFQDANTfQkDen4V7z
```

関連するセクション

- プロビジョニングされた容量を表示する (GET provisioned-capacity)

データ取り出しポリシーの設定 (ポリシーの PUT)

説明

このオペレーションは、PUTリクエストで指定された AWS リージョンでデータ取り出しポリシーを設定して有効にします。の AWS リージョンごとに 1 つのポリシーを設定できます AWS アカウント。ポリシーは PUT オペレーションが成功してから数分以内に有効になります。

ポリシーの設定オペレーションは、ポリシーが有効になる前に進行中だった取り出しジョブには影響を与えません。データ取り出しポリシーの詳細については、「S3 Glacier データ取得ポリシー」を参照してください。

リクエスト

構文

データ取り出しポリシーを設定するには、以下の構文例に示しているように、データ取り出しポリシーの URI に HTTP PUT リクエストを送信します。

```
PUT /AccountId/policies/data-retrieval HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
Content-Length: Length
x-amz-glacier-version: 2012-06-01

{
  "Policy":
    {
      "Rules":[
         {
             "Strategy": String,
             "BytesPerHour": Number
```

```
            }
        ]
    }
}
```

> **Note**
>
> `AccountId` 値は AWS アカウント ID です。この値は、リクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID と一致する必要があります。 AWS アカウント ID またはオプションで単一の '-' (ハイフン) を指定できます。この場合、Amazon S3 Glacier はリクエストの署名に使用される認証情報に関連付けられた AWS アカウント ID を使用します。お客様のアカウント ID を指定する場合は、ハイフン(`-`)を含めないでください。

リクエストパラメータ

このオペレーション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。共通のリクエストヘッダーの詳細については、「一般的なリクエストヘッダー」を参照してください。

リクエスト本文

リクエストボディ には、次の JSON フィールドが含まれます。

BytesPerHour

1 時間あたりに取り出すことのできるデータの最大サイズ (バイト) 。

このフィールドは、Strategy フィールドの値が `BytesPerHour` に設定されている場合にのみ必要です。Strategy フィールドが `BytesPerHour` に設定されていない場合にこのフィールドを設定すると、PUT オペレーションは拒否されます。

タイプ: 数値

Required: Strategy フィールドが `BytesPerHour` に設定されている場合は Yes です。それ以外の場合は No です。

Valid Values: 最小の整数値は 1 です。最大の整数値は 2 の 63 乗 - 1 までです。

ルール

ポリシールール。これは列挙型ですが、現在は 1 つのルールのみを設定します。Strategy フィールドと必要に応じて BytesPerHour フィールドを含みます。

型: 配列

必須: はい

方針

設定するデータ取り出しポリシーのタイプです。

型: 文字列

必須: はい

有効な値: `BytesPerHourFreeTierNone||`。`BytesPerHour` は、コンソールで 最大取得率 を選択することに相当します。`FreeTier` は、コンソールで 無料利用枠のみ を選択することに相当します。None は、コンソールで No Retrieval Policy (取り出しポリシーなし) を選択することに相当します。コンソールでのデータ取り出しポリシーの選択の詳細については、「S3 Glacier データ取得ポリシー」を参照してください。

レスポンス

構文

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

レスポンスヘッダー

この操作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。共通のレスポンスヘッダーの詳細については、「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

このオペレーションはレスポンス本文を返しません。

エラー

Amazon S3 Glacier の例外とエラーメッセージについては、「エラーレスポンス」を参照してください。

例

リクエストの例

以下の例では、Strategy フィールドを `BytesPerHour` に設定する HTTP PUT リクエストを送信しています。

```
PUT /-/policies/data-retrieval HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
  "Policy":
    {
      "Rules":[
         {
             "Strategy":"BytesPerHour",
             "BytesPerHour":10737418240
          }
       ]
    }
}
```

以下の例では、Strategy フィールドを `FreeTier` に設定する HTTP PUT リクエストを送信しています。

```
PUT /-/policies/data-retrieval HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

```
{
  "Policy":
    {
      "Rules":[
         {
             "Strategy":"FreeTier"
          }
       ]
    }
}
```

以下の例では、Strategy フィールドを None に設定する HTTP PUT リクエストを送信しています。

```
PUT /-/policies/data-retrieval HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20170210T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/
us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-
version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
  "Policy":
    {
      "Rules":[
         {
             "Strategy":"None"
          }
       ]
    }
}
```

レスポンスの例

リクエストが成功した場合、次の例に示しているように、Amazon S3 Glacier (S3 Glacier) はポリシーを設定し、`HTTP 204 No Content` を返します。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

関連するセクション

- データ取り出しポリシーの取得 (ポリシーの GET)

- ジョブの開始 (ジョブの POST)

ドキュメント履歴

- 現行製品バージョン: 2012-06-01

2018 年 7 月 5 日以降の Amazon S3 Glacier 開発者ガイドの各リリースにおける重要な変更点を以下の表に示します。このドキュメントの更新に関する通知を受け取るには、RSS フィードにサブスクライブできます。

変更	説明	日付
S3 バッチオペレーションによる標準復元リクエストの開始時間が短縮されました	S3 バッチオペレーションによる復元リクエストの標準取得を数分で開始できるようになりました。詳細については、「アーカイブの取り出しオプション」を参照してください。	2023 年 8 月 9 日
Amazon S3 で、S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive のより高い復元リクエストのレートをサポート	Amazon S3 は、S3 Glacier Flexible Retrieval および S3 Glacier Deep Archive ストレージクラスで、 AWS アカウント ごとに、1 秒あたり最大 1,000 件のトランザクションの復元リクエストをサポートします。	2022 年 11 月 15 日
Amazon Glacier の名前変更	Amazon Glacier は Amazon S3 との Glacier の統合をより適切に反映し、Amazon S3 Glacier と名前を変えました。	2018 年 11 月 20 日
更新を RSS で今すぐ入手可能	RSS フィードにサブスクライブすると、Amazon S3 Glacier デベロッパー ガイドの更新に	2018 年 7 月 5 日

関する通知を受け取れるようになりました。

以前の更新

2018 年 7 月 5 日より前の Amazon S3 Glacier 開発者 Guideの各リリースにおける重要な変更点を以下の表に示します。

変更	説明	リリース日
データの迅速取り出しと大容量取り出し	S3 Glacier では、標準取り出しに加えて迅速取り出しと大容量取り出しがサポートされるようになりました。詳細については、「アーカイブの取り出しオプション」を参照してください。	2016 年 11 月 21 日
Vault Lock	S3 Glacier では、ボールトロックがサポートされるようになりました。ボールトロックでは、ボールトロックポリシーを使用して、S3 Glacier の各ボールトに対するコンプライアンスコントロールを簡単にデプロイして適用することができます。詳細については、S3 Glacier ボールトロックおよびボールトロックポリシーを参照してください。	2015 年 7 月 8 日
ボールトのタグ付け	S3 Glacier では、リソースとコストの管理を簡単にするため、S3 Glacier ボールトにタグ付けできるようになりました。タグは、ボールトを定義して関連付けることができるラベルであり、タグを使用すると、 AWS コストレポートなどのオペレーションにフィルタリング機能が追加されます。詳細については、Amazon S3 Glacier リソースのタグ付けおよびS3 Glacier ボールトにタグを付けるを参照してください。	2015 年 6 月 22 日
ボールトアクセスポリシー	S3 Glacier は、ボールトアクセスポリシーを使用して、個別の S3 Glacier ボールトへのアクセスの管理をサポートするようになりました。ボールトで直接アクセスポリシーを定義し、組織に対して内部的に、およ	2015 年 4 月 27 日

変更	説明	リリース日
	び外部のビジネスパートナーに対して、ユーザーとビジネスグループにボールトアクセスを簡単に付与できるようになりました。詳細については、「ボールトアクセスポリシー」を参照してください。	
データ取り出しポリシーおよび監査ログ記録	S3 Glacier では、データ取り出しポリシーと監査ログ記録がサポートされるようになりました。データ取り出しポリシーを使用することで、データ取り出し制限を簡単に設定して、データ取り出しコストの管理を簡素化できます。で数回クリックするか、S3 Glacier API AWS Management Console を使用して、独自のデータ取得制限を定義できます。詳細については、「S3 Glacier データ取得ポリシー」を参照してください。 さらに、S3 Glacier は による監査ログ記録をサポートするようになりました。これにより AWS CloudTrail、アカウントの S3 Glacier API コールが記録され、指定した Amazon S3 バケットにログファイルが配信されます。詳細については、「を使用した Amazon S3 Glacier API コールのログ記録 AWS CloudTrail」を参照してください。	2014 年 12 月 11 日
Java のサンプルの更新	このガイド内の AWS SDK for Javaを使用する Java のコードの例を更新しました。	2014 年 6 月 27 日
ボールトインベントリの取得制限	アーカイブの作成日でフィルタリングするか、制限を設定することによって、取得されるボールトインベントリの項目数を制限できるようになりました。インベントリの取得の制限の詳細については、「インベントリの取得の範囲」トピックの「ジョブの開始 (ジョブの POST)」を参照してください。	2013 年 12 月 31 日
古い URL の削除	コード例から古いセキュリティ認証情報のページの URL を削除しました。	2013 年 7 月 26 日

変更	説明	リリース日
範囲取得のサポート	S3 Glacier が、アーカイブの特定の範囲の取得をサポートするようになりました。S3 Glacier に対してアーカイブ全体またはアーカイブの一部をダウンロードするための準備をリクエストするジョブを開始できます。アーカイブが非常に大きい場合には、アーカイブの準備を複数のシーケンシャルジョブに分けて開始すると、すぐれた費用対効果が得られることがあります。 詳細については、「S3 Glacier でのアーカイブのダウンロード」を参照してください。	2012 年 11 月 13 日
新規ガイド	Amazon S3 Glacier 開発者ガイドの初回リリース。	2012 年 8 月 20 日

AWS 用語集

最新の AWS 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。