开发人员指南

Amazon S3 Glacier

API 版本 2012-06-01

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

Amazon S3 Glacier: 开发人员指南

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

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆或者贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

... x
什么是 Amazon S3 Glacier？ ... 1
您目前正在使用 S3 Glacier 吗？ ... 1
数据模型 ... 2
文件库 ... 3
档案 ... 4
作业 ... 4
通知配置 ... 5
支持的操作 ... 5
文件库操作 ... 6
归档操作 ... 6
作业 ... 6
访问 S3 Glacier ... 6
区域和端点 ... 7
开始使用 ... 8
第 1 步：准备工作 ... 9
设置一个 AWS 账户 ... 9
下载相应的 AWS SDK ... 10
步骤 2：创建文件库 ... 11
步骤 3：将存档上传到文件库 ... 12
使用 Java 上传档案 ... 13
使用 .NET 上传档案 ... 18
步骤 4：从文件库下载存档 ... 20
使用 Java 下载档案 ... 21
使用 .NET 下载档案 ... 22
步骤 5：从文件库删除存档 ... 24
相关部分 ... 25
使用 Java 删除档案 ... 25
使用 .NET 删除档案 ... 26
使用删除档案 AWS CLI ... 27
步骤 6：删除文件库 ... 30
我从这里可以了解哪些内容？ ... 31
处理文件库 ... 32
S3 Glacier 中的文件库操作 ... 32

创建和删除文件库 33
检索文件库元数据 33
下载文件库清单 33
配置文件库通知 34
创建文件库 34
使用 Java 创建文件库 35
使用 .NET 创建文件库 38
使用 REST 创建文件库 42
使用控制台创建文件库 42
使用创建文件库 AWS CLI 43
检索文件库元数据 44
使用 Java 检索文件库元数据 44
使用 .NET 检索文件库元数据 47
使用 REST 检索文件库元数据 49
使用检索文件库元数据 AWS CLI 49
下载文件库清单 51
关于库存 52
使用 Java 下载文件库清单 53
使用 .NET 下载文件库清单 60
使用 REST 下载文件库清单 67
使用下载文件库清单 AWS CLI 67
配置文件库通知 70
一般概念 71
使用 Java 配置文件库通知 72
使用 .NET 配置文件库通知 75
使用 REST API 配置文件库通知 78
使用控制台配置文件库通知 78
使用 CLI 配置文件库通知 80
删除文件库 81
使用 Java 删除文件库 82
使用 .NET 删除文件库 83
使用 REST 删除文件库 84
使用控制台删除空文件库 84
使用删除文件库 AWS CLI 85
标记文件库 88
使用 Amazon S3 Glacier 控制台标记文件库 89

使用标记文件库 AWS CLI 90
使用 Amazon S3 Glacier API 标记文件库 91
相关部分 91
保管库锁定 92
文件库锁定概述 92
使用 API 锁定文件库 93
使用 CLI 锁定文件库 94
使用控制台锁定文件库 96
处理档案 98
归档操作 98
上传档案 99
查找存档 99
下载档案 99
删除档案 99
更新档案 99
维护客户端档案元数据 100
上传档案 100
上传档案的选项 100
在单一操作中上传档案 101
分段上传大型档案 111
下载档案 127
检索档案 127
使用 Java 下载档案 131
使用 .NET 下载档案 147
使用 Python 下载大型档案 162
使用 REST 下载档案 170
使用下载档案 AWS CLI 171
删除档案 174
使用 Java 删除档案 175
使用 .NET 删除档案 177
使用 REST 删除档案 180
使用删除档案 AWS CLI 180
使用 AWS SDKs 183
AWS Java 和.NET 的 SDK 库 183
什么是低级 API？ 183
什么是高级 API？ 184

何时使用高级和低级 API？ ... 184
与... 合作 AWS SDKs ... 184
使用 适用于 Java 的 AWS SDK ... 185
使用低级 API ... 186
使用高级 API ... 187
使用 Eclipse 运行 Java 示例 ... 187
设置端点 ... 188
使用 适用于 .NET 的 AWS SDK ... 189
使用低级 API ... 189
使用高级 API ... 190
运行 .NET 示例 ... 191
设置端点 ... 191
代码示例 ... 192
基本功能 ... 194
Hello Amazon S3 Glacier ... 195
操作 ... 196
场景 ... 260
归档文件，获取通知并启动任务 ... 261
获取档案内容并删除档案 ... 267
安全性 ... 273
数据保护 ... 273
数据加密 ... 274
密钥管理 ... 274
互联网络流量保密性 ... 274
身份和访问管理 ... 275
受众 ... 276
使用身份进行身份验证 ... 276
使用策略管理访问 ... 279
Amazon S3 Glacier 如何与 IAM 配合使用 ... 280
基于身份的策略示例 ... 287
基于资源的策略示例 ... 294
故障排除 ... 299
Amazon S3 Glacier API 权限参考 ... 300
日志记录和监控 ... 308
合规性验证 ... 309
恢复能力 ... 311

基础设施安全性 312
VPC 端点 312
数据检索策略 313
选择 S3 Glacier 数据检索策略 313
“Free Tier Only (仅免费套餐)”策略 314
“Max Retrieval Rate (最大检索速率)”策略 314
“No Retrieval Limit (无检索限制)”策略 314
使用 S3 Glacier 控制台设置数据检索策略 315
使用 Amazon S3 Glacier API 设置数据检索策略 315
使用 Amazon S3 Glacier REST API 设置数据检索策略 315
使用 AWS SDKs 来设置数据检索策略 316
为资源添加标签 317
标签基本知识 317
标签限制 318
使用标签跟踪成本 318
使用标签管理访问控制 318
相关部分 319
使用审核日志 AWS CloudTrail 320
中的亚马逊 S3 Glacier 信息 CloudTrail 320
了解 Amazon S3 Glacier 日志文件条目 321
API 参考 324
常见请求标头 324
常用响应标头 327
对请求进行签名 328
实例签名计算 329
为流式处理操作计算签名 330
计算校验和 332
树形哈希示例 1：在单一请求中上传档案 334
树形哈希示例 2：使用分段上传来上传档案 334
计算文件的树形哈希 335
下载数据时接收校验和 344
错误响应 347
示例 1：具有不存在的任务 ID 的描述任务请求 349
示例 2：请求参数具有无效值的列出任务请求 350
文件库操作 351
中止文件库锁定 352

向文件库添加标签 354
创建文件库 357
完成文件库锁定 360
删除文件库 363
删除文件库访问策略 366
删除文件库通知 368
描述文件库 370
获取文件库访问策略 374
获取文件库锁定 377
获取文件库通知 382
启动文件库锁定 385
列出文件库的标签 389
列出文件库 392
从文件库中删除标签 399
设置文件库访问策略 402
设置文件库通知配置 405
归档操作 408
删除档案 408
上传档案 411
分段上传操作 416
中止分段上传 417
完成分段上传 419
开始分段上传 423
列出分段 428
列出分段上传 434
上传分段 441
任务操作 446
描述作业 446
获取任务输出 456
启动任务 465
列出作业 475
在任务操作中使用的数据类型 484
CSVInput 485
CSVOutput 486
加密 488
GlacierJobDescription 488

授权 492
被授权者 493
InputSerialization 494
InventoryRetrievalJobInput 494
jobParameters 496
OutputLocation 498
OutputSerialization 499
S3Location 499
SelectParameters 501
数据检索操作 502
获取数据检索策略 502
列出预配置容量 506
购买预配置容量 509
设置数据检索策略 512
文档历史记录 518
早期更新 518
AWS 词汇表 521

此页面仅适用于使用文件库和 2012 年原始 REST API 的 S3 Glacier 服务的现有客户。

如果您正在寻找归档存储解决方案，建议使用 Amazon S3 中的 S3 Glacier 存储类 S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive。要了解有关这些存储选项的更多信息，请参阅《Amazon S3 用户指南》中的 S3 Glacier 存储类和使用 S3 Glacier 存储类的长期数据存储。这些存储类使用 Amazon S3 API，适用于所有区域，并且可以在 Amazon S3 控制台中管理。它们提供存储成本分析、Storage Lens 存储分析功能、高级可选加密功能等功能。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。

什么是 Amazon S3 Glacier？

如果您目前正在使用 Amazon S3 Glacier（S3 Glacier）服务并想了解更多信息，可在本指南中找到所需的信息。S3 Glacier 是一项安全、持久的服务，通过使用文件库实现低成本的数据归档和长期备份。有关 S3 Glacier 服务定价的更多信息，请参阅 S3 Glacier 定价。

主题

- 您目前正在使用 S3 Glacier 吗？
- Amazon S3 Glacier 数据模型
- S3 Glacier 中支持的操作
- 访问 Amazon S3 Glacier

您目前正在使用 S3 Glacier 吗？

> **Note**
>
> 本部分将介绍 S3 Glacier 服务。如果您目前使用 S3 Glacier 存储类（S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive），请参阅《Amazon S3 用户指南》中的归档对象的存储类。

如果您目前使用 S3 Glacier 服务并想了解更多信息，建议您先阅读以下部分：

- 什么是 Amazon S3 Glacier — 本节的其余部分描述了底层数据模型、其支持的操作以及可用于与服务交互的操作。 AWS SDKs
- 入门 - Amazon S3 Glacier 入门部分将指导您完成创建文件库、上传档案、创建下载档案的任务、检索任务输出以及删除档案的过程。

> **Important**
>
> S3 Glacier 确实提供了一个控制台。但是，任何存档操作（例如上传、下载或删除）都需要您使用 AWS Command Line Interface (AWS CLI) 或编写代码。存档操作没有控制台支持。例如，要上传数据（例如照片、视频和其他文档），您必须使用 AWS CLI 或编写代码来发出请求，方法是直接使用 REST API 或使用 AWS SDKs。

要安装 AWS CLI，请参阅AWS Command Line Interface。有关将 S3 Glacier 与 AWS CLI配合使用的更多信息，请参阅 S3 Glacier 的AWS CLI 参考。有关使用将档案上传 AWS CLI 到 S3 Glacier 的示例，请参阅将 S3 Glacier 与一起使用 AWS Command Line Interface。

除了入门章节之外，您可能还想了解有关 S3 Glacier 操作的更多信息。以下各节提供了有关使用 REST API 以及 AWS SDKs 适用于 Java 和 Microsoft .NET 的 REST API 使用 S3 Glacier 的详细信息：

- AWS SDKs 与 Amazon S3 Glacier 搭配使用

  本节概述了本指南中各种代码示例中 AWS SDKs 使用的代码。回顾此部分将有助于阅读以下部分。它包括对它们 SDKs 提供的高级和低级APIs 的概述、何时使用它们，以及运行本指南中提供的代码示例的常见步骤。

- 在 Amazon S3 Glacier 中处理文件库

  此部分详细介绍了各种文件库操作，例如，创建文件库、检索文件库元数据、使用任务来检索文件库清单，以及配置文件库通知。除了使用 S3 Glacier 控制台外，您还可以使用 AWS SDKs 进行各种保管库操作。本节介绍了 API，并使用 适用于 Java 的 AWS SDK 和提供了工作示例 适用于 .NET 的 AWS SDK。

- 在 Amazon S3 Glacier 中处理档案

  此部分详细介绍了归档操作，例如，在单一请求中上传档案，或者使用分段上传操作来分段上传大型档案。此外，该部分还说明了如何创建任务来异步下载档案的操作。此部分提供了使用 适用于 Java 的 AWS SDK 和 适用于 .NET 的 AWS SDK的示例。

- Amazon S3 Glacier 的 API 参考

  S3 Glacier 是一项 RESTful 服务。此部分描述了 REST 操作，包括所有操作的语法以及示例请求和响应。S AWS DK 库封装了此 API，从而简化了您的编程任务。

Amazon S3 Glacier 数据模型

Amazon S3 Glacier 数据模型核心组件包括文件库和档案。S3 Glacier 是一项基于 REST 的 Web 服务。根据 REST，文件库和档案是资源。此外，S3 Glacier 数据模型还包括任务和通知配置资源。这些资源是对核心资源的补充。

主题

- 文件库
- 档案
- 作业
- 通知配置

文件库

在 S3 Glacier 中，文件库 是用于存储档案的容器。文件库与 Amazon S3 存储桶类似。创建文件库时，需要指定名称并选择要在 AWS 区域 何处创建文件库。

每个文件库资源都有唯一的地址。一般格式为：

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

例如，假设您使用 ID 为 111122223333 的账户在美国西部（俄勒冈州）区域创建一个文件库 (`examplevault`)。那么，您可以使用以下 URI 为此文件库编址：

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

以下是 URI 的各个组成部分的含义：

- `glacier.us-west-2.amazonaws.com` 标识美国西部（俄勒冈州）区域。
- `111122223333`是拥有文件库的 AWS 账户 ID。
- `vaults` 是指 AWS 账户拥有的文件库集合。
- `examplevault` 标识了文件库集合中的特定文件库。

AWS 账户 可以在任何支持的 AWS 区域存储库中创建文件库。有关支持的列表 AWS 区域，请参阅访问 Amazon S3 Glacier。在一个区域内，一个账户必须使用唯一的文件库名称。 AWS 账户 可以在不同的区域创建同名文件库。

您可以在文件库中存储无限多个档案。根据您的业务或应用程序需求，您可以将这些档案存储在一个或多个文件库中。

S3 Glacier 支持各种文件库操作。文件库操作是特定于区域的。例如，创建文件库时，您会在特定的区域创建。当您请求文件库列表时，您需要从特定区域请求文件库 AWS 区域，结果列表仅包含在该特定区域创建的文件库。

档案

档案 可以是任何数据，例如照片、视频或文档。档案与 Amazon S3 对象类似，是 S3 Glacier 中的基本存储单位。每个档案都有唯一的 ID 和可选的描述。您只能在上传档案时指定此可选描述。S3 Glacier 为档案分配一个 ID，该ID在存储档案的地方是唯一的。 AWS 区域

每个档案都有唯一的地址。一般格式如下：

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

以下是存储在美国西部（俄勒冈州）区域中账户 111122223333 下的 `examplevault` 文件库中的档案的示例 URI：

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

您可以在文件库中存储无限多个档案。

作业

S3 Glacier 任务可以检索档案，也可以获取文件库的清单。

在 S3 Glacier 中，检索档案和文件库清单（档案列表）是异步操作，您首先要在其中启动任务，然后在 S3 Glacier 完成任务后下载任务输出。

> **Note**
>
> S3 Glacier 提供了冷存储数据归档解决方案。如果您的应用程序需要一套要求进行实时数据检索的存储解决方案，则可以考虑使用 Amazon S3。有关更多信息，请参阅 Amazon Simple Storage Service (Amazon S3)。

要启动文件库清单任务，您需要提供文件库名称。档案检索任务需要文件库名称和档案 ID。您还可以提供可选的任务描述来帮助标识任务。

档案检索和文件库清单任务与文件库相关联。在任何时间点，一个文件库可以同时进行多个任务。在您发送任务请求（启动任务）时，S3 Glacier 会向您返回一个任务 ID，以跟踪该任务。每个任务都会由以下格式的 URI 唯一标识：

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

以下是与美国西部（俄勒冈州）区域中账户 111122223333 下的 `examplevault` 文件库关联的任务示例。

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

对于每个任务，S3 Glacier 都会保留相关信息，例如，任务类型、描述、创建日期、完成日期以及任务状态。您可以获取有关特定任务的信息，也可以获取与某个文件库相关联的所有任务的列表。S3 Glacier 返回的作业列表包括所有正在进行的任务以及最近完成的任务。

通知配置

由于运行任务需要花费时间，因此，S3 Glacier 支持一种通知机制，以便在任务完成时通知您。您可以将文件库配置为在任务完成时向 Amazon Simple Notification Service (Amazon SNS) 主题发送通知。您可以在通知配置中为每个文件库指定一个 Amazon SNS 主题。

S3 Glacier 会将通知配置存储为 JSON 文档。以下是文件库通知配置示例：

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

通知配置与文件库相关联；每个文件库都可以有一个通知配置。每个通知配置资源都会由以下格式的 URI 唯一标识：

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

S3 Glacier 支持设置、获取和删除通知配置的操作。如果您删除了通知配置，则针对该文件库的任何数据检索操作完成时，系统都不会发送任何通知。

S3 Glacier 中支持的操作

为了处理文件库和档案（请参阅Amazon S3 Glacier 数据模型），Amazon S3 Glacier 支持一组操作。在所有受支持的操作中，只有以下操作是异步操作：

- 检索档案
- 检索文件库清单（档案列表）

这些操作要求您先启动任务，然后下载任务输出。以下部分概述了 S3 Glacier 操作：

文件库操作

S3 Glacier 提供了创建和删除文件库的操作。您可以获取特定文件库或 AWS 区域中所有文件库的文件库描述。文件库描述提供创建日期、文件库中的档案数、文件库中所有档案使用的总大小（以字节为单位）以及 S3 Glacier 生成文件库清单的日期等信息。S3 Glacier 还提供了用于在文件库中设置、检索和删除通知配置的操作。有关更多信息，请参阅 在 Amazon S3 Glacier 中处理文件库。

归档操作

S3 Glacier 为您提供了上传和删除档案的操作。您无法更新现有的档案；您必须删除现有的档案，然后上传新档案。您每次上传档案时，S3 Glacier 都会生成新的档案 ID。有关更多信息，请参阅 在 Amazon S3 Glacier 中处理档案。

作业

您可以启动 S3 Glacier 任务以对档案执行检索或获取文件库的清单。

下面是 S3 Glacier 任务的类型：

- `archive-retrieval` - 检索档案。

  有关更多信息，请参阅 在 S3 Glacier 中下载档案。
- `inventory-retrieval` - 清点文件库。

  有关更多信息，请参阅 在 Amazon S3 Glacier 中下载文件库清单。

访问 Amazon S3 Glacier

Amazon S3 Glacier 是一项 RESTful 网络服务，它使用 HTTP 和 HTTPS 作为传输协议，使用 JavaScript 对象表示法 (JSON) 作为消息序列化格式。您的应用程序代码可以直接向 S3 Glacier Web 服务 API 发送请求。在直接使用 REST API 时，您必须编写必要的代码来对您的请求签名以及验证您的请求。有关该 API 的更多信息，请参阅Amazon S3 Glacier 的 API 参考。

或者，您可以使用封装 S3 Glacier REST API 调用的方法来简化应用程序开发。 AWS SDKs 您提供证书后，这些库会处理身份验证和请求登录事宜。有关如何使用 AWS SDKs的更多信息，请参阅AWS SDKs 与 Amazon S3 Glacier 搭配使用。

此外，S3 Glacier 确实提供了一个控制台。但是，所有存档和任务操作都需要您直接使用 REST API 或 AWS SDK 包装器库来编写代码和发出请求。要访问 S3 Glacier 控制台，请转到 S3 Glacier 控制台。

区域和端点

您在特定的文件库中创建文件库 AWS 区域。您始终要将 S3 Glacier 请求发送到特定于 AWS 区域的端点。有关 S3 Glacier 支持的 AWS 区域 列表，请参阅《AWS 一般参考》中的 Amazon S3 Glacier 端点和配额。

Amazon S3 Glacier 入门

您可以通过使用文件库和档案开始使用 Amazon S3 Glacier (S3 Glacier)。文件库 是用于存储档案的容器，档案是您存储在文件库中的任何对象（例如照片、视频或文档）。档案是 S3 Glacier 中的基本存储单位。此入门练习为您提供了相关说明，帮助您了解如何对文件库和档案资源进行基本 S3 Glacier 操作。有关这些资源的更多信息，请参阅Amazon S3 Glacier 数据模型部分。

在入门练习中，您将创建文件库、上传和下载档案，然后删除档案和文件库。您可以编程方式执行所有的这些操作。但是，入门练习会使用 S3 Glacier 管理控制台来创建和删除文件库。要上传和下载档案，本入门部分使用了 适用于 Java 的 AWS SDK 和的高级别 API 适用于 .NET 的 AWS SDK。使用 S3 Glacier 时，高级 API 将提供简化的编程体验。有关将高级 API 与配合使用的更多信息 AWS SDKs，请参阅AWS SDKs 与 Amazon S3 Glacier 搭配使用。

> **Important**
>
> S3 Glacier 确实提供了一个控制台。但是，任何存档操作（例如上传、下载或删除）都需要您使用 AWS Command Line Interface (CLI) 或编写代码。存档操作没有控制台支持。例如，要上传数据（例如照片、视频和其他文档），您必须使用 AWS CLI 或编写代码来发出请求，方法是直接使用 REST API 或使用 AWS SDKs。
>
> 要安装 AWS CLI，请参阅AWS Command Line Interface。有关将 S3 Glacier 与配合使用的更多信息 AWS CLI，请参阅 S3 Glacier AWS CLI 参考资料。有关使用将档案上传 AWS CLI 到 S3 Glacier 的示例，请参阅将 S3 Glacier 与一起使用 AWS Command Line Interface。

此入门练习为您提供了上传和下载存档的 Java 和 C# 代码示例。入门练习的最后一个部分为您提供了相关步骤，您可以通过这些步骤详细了解开发人员的 S3 Glacier 使用体验。

主题

- 步骤 1：开始使用 S3 Glacier 之前
- 步骤 2：在 S3 Glacier 中创建文件库
- 步骤 3：在 S3 Glacier 中将档案上传到文件库
- 步骤 4：在 S3 Glacier 中从文件库下载档案
- 步骤 5：从 S3 Glacier 中的文件库删除档案
- 步骤 6：在 S3 Glacier 中删除文件库
- 我从这里可以了解哪些内容？

步骤 1：开始使用 S3 Glacier 之前

在开始本练习之前，你必须先注册 AWS 账户 （如果你还没有），然后下载其中一个 AWS SDKs。有关说明，请参阅以下部分。

主题

- 设置管理员用户 AWS 账户 和管理员用户
- 下载相应的 AWS SDK

设置管理员用户 AWS 账户 和管理员用户

如果您尚未这样做，则必须注册 AWS 账户 并在该帐户中创建管理员用户。

要完成设置，请遵循以下主题中的说明。

设置 AWS 账户 并创建管理员用户

报名参加 AWS

当您注册 Amazon Web Services (AWS) AWS 账户 时，系统会自动注册中的所有服务 AWS，包括 S3 Glacier。您只需为使用的服务付费。有关 S3 Glacier 使用费率的更多信息，请参阅 Amazon S3 Glacier 定价页面。

如果您已经有 AWS 账户，请跳至下载相应的 AWS SDK。如果您没有 AWS 账户，请按以下步骤创建一个。

如果您没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 AWS 账户

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

   在注册时，将接到电话或收到短信，要求使用电话键盘输入一个验证码。

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

要创建管理员用户，请选择以下选项之一。

选择一种方法来管理您的管理员	目的	方式	您也可以
在 IAM Identity Center 中 （建议）	使用短期凭证访问 AWS。 这符合安全最佳实操。有关最佳实践的信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。	有关说明，请参阅《AWS IAM Identity Center 用户指南》中的入门。	通过在《AWS Command Line Interface 用户指南》 AWS IAM Identity Center中配置 AWS CLI 要使用的来配置编程访问权限。
在 IAM 中 （不推荐使用）	使用长期凭证访问 AWS。	按照《IAM 用户指南》中的创建用于紧急访问的 IAM 用户中的说明进行操作。	按照《IAM 用户指南》中的管理 IAM 用户的访问密钥，配置编程式访问。

下载相应的 AWS SDK

要尝试入门练习，您必须决定要使用哪种编程语言，然后为您的开发平台下载相应的 AWS SDK。

入门练习提供了 Java 和 C# 示例。

下载 适用于 Java 的 AWS SDK

要测试此开发人员指南中的 Java 示例，需要 适用于 Java 的 AWS SDK。您有以下几种下载选择：

- 如果你使用的是 Eclipse，你可以使用更新网站 AWS Toolkit for Eclipse http://aws.amazon.com/eclipse/ 下载并安装。有关更多信息，请参阅 AWS Toolkit for Eclipse。
- 如果您使用任何其他 IDE 来创建应用程序，请下载适用于 Java 的 AWS SDK。

下载 适用于 .NET 的 AWS SDK

要测试此开发人员指南中的 C# 示例，需要 适用于 .NET 的 AWS SDK。您有以下几种下载选择：

- 如果您使用的是 Visual Studio，则可以同时安装 适用于 .NET 的 AWS SDK 和 AWS Toolkit for Visual Studio。该工具包提供了 Visual Studio 的 AWS 资源管理器以及可用于开发的项目模板。要下载 适用于 .NET 的 AWS SDK，请访问 http://aws.amazon.com/sdkfornet。默认情况下，安装脚本会同时安装 S AWS DK 和 AWS Toolkit for Visual Studio。要了解有关工具包的更多信息，请参阅AWS Toolkit for Visual Studio 用户指南。
- 如果您使用任何其他 IDE 创建应用程序，则可以使用上述步骤中提供的相同链接并仅安装 适用于 .NET 的 AWS SDK。

步骤 2：在 S3 Glacier 中创建文件库

文件库是用于存储存档的容器。您的第一步是在其中一个支持的存储库中创建一个保管库 AWS 区域。有关 Amazon S3 Glacier 支持的终端节点列表，请参阅《 AWS 区域 AWS 一般参考》中的 Amazon S3 Glacier 终端节点和配额。

您可以编程方式创建文件库，也可以通过使用 S3 Glacier 控制台创建文件库。此部分使用控制台创建文件库。

创建文件库

1. 登录 AWS Management Console 并在https://console.aws.amazon.com/glacier/家中打开 S3 Glacier 控制台。
2. 在左侧导航窗格中，选择文件库。
3. 选择创建文件库。

   将打开 创建文件库页面。
4. 在 “选择区域” 下， AWS 区域 从 “区域” 选择器中选择。您的文件库将位于您选择的区域中。
5. 对于文件库名称，输入文件库的名称。

   以下是文件库的命名要求：

   - 文件库名称在 AWS 账户 和创建文件库时必须是唯一的。 AWS 区域
   - 文件库名称的长度必须介于 1 到 255 个字符之间。
   - 文件库名称只能包含以下字符：a–z、A–Z、0–9、_（下划线）、-（连字符）和 .（句点）。

6. 在事件通知下，要在文件库中打开或关闭任务完成时的通知，请选择以下设置之一：

   - 关闭通知 - 将关闭通知，在指定任务完成后不会向 Amazon Simple Notification Service (Amazon SNS) 主题发送通知。

   - 打开通知 - 将打开通知，在指定任务完成后将向提供的 Amazon SNS 主题发送通知。

     如果您选择了打开通知，请参阅使用 Amazon S3 Glacier 控制台配置文件库通知。

7. 如果 AWS 区域 和保管库名称正确，请选择创建文件库。

新的文件库现已列在 S3 Glacier 控制台的文件库页面上。

步骤 3：在 S3 Glacier 中将档案上传到文件库

在此步骤中，您将示例档案上传到您在前面的步骤中创建的文件库（请参阅步骤 2：在 S3 Glacier 中创建文件库）。请根据您使用的开发平台选择本节末尾的链接之一。

> **Important**
>
> 任何归档操作（例如上传、下载或删除）均要求您使用 AWS Command Line Interface (CLI) 或编写代码。存档操作没有控制台支持。例如，要上传数据（例如照片、视频和其他文档），您必须使用 AWS CLI 或编写代码来发出请求，方法是直接使用 REST API 或使用 AWS SDKs。
> 要安装 AWS CLI，请参阅AWS Command Line Interface。有关将 S3 Glacier 与配合使用的更多信息 AWS CLI，请参阅 S3 Glacier AWS CLI 参考。有关使用将档案上传 AWS CLI 到 S3 Glacier 的示例，请参阅将 S3 Glacier 与一起使用 AWS Command Line Interface。

存档是您存储在文件库中的任何对象（例如，照片、视频或文档）。档案是 S3 Glacier 中的基本存储单位。您可以在单一请求中上传存档。对于大型档案，S3 Glacier 提供了分段上传 API 操作，它可让您分段上传档案。

在此入门部分中，您会在单一请求中上传示例存档。对于此练习，您会指定一个较小的文件。对于较大的文件，适合使用分段上传。有关更多信息，请参阅 分段上传大型档案（分段上传）。

主题

- 使用以下方法将档案上传到 S3 Glacier 中的文件库 适用于 Java 的 AWS SDK

- 使用以下方法将档案上传到 S3 Glacier 中的文件库 适用于 .NET 的 AWS SDK

使用以下方法将档案上传到 S3 Glacier 中的文件库 适用于 Java 的 AWS SDK

以下 Java 代码示例使用的高级别 API 将示例档案上传 适用于 Java 的 AWS SDK 到文件库。在代码示例中，请注意以下情况：

- 以下示例创建 `AmazonGlacierClient` 类的实例。
- 该示例使用了 `ArchiveTransferManager` 类的 `upload` API 操作，该类属于 适用于 Java 的 AWS SDK高级 API。
- 该示例使用美国西部（俄勒冈州）区域 (`us-west-2`)。

有关如何运行此示例的 step-by-step说明，请参阅使用 Eclipse 运行 Amazon S3 Glacier 的 Java 示例。您必须更新待上传档案文件名称旁显示的代码。

> **Note**
>
> Amazon S3 Glacier 在文件库中保留一份所有档案的清单。当您上传以下示例中的档案时，该档案直到文件库清单已更新后才会在管理控制台的文件库中显示。此更新通常每天进行一次。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
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();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UploadArchive中的。

使用以下方法将档案上传到 S3 Glacier 中的文件库 适用于 .NET 的 AWS SDK

以下 C# 代码示例使用的高级别 API 将示例档案上传 适用于 .NET 的 AWS SDK 到文件库。在代码示例中，请注意以下情况：

- 该示例为指定的 Amazon S3 Glacier 区域端点创建 `ArchiveTransferManager` 类的实例。
- 该代码示例使用美国西部（俄勒冈州）区域 (`us-west-2`)。
- 该示例使用 `Upload` 类的 `ArchiveTransferManager` API 操作上传档案。对于小型档案，此操作会将档案直接上传到 S3 Glacier。对于大型档案，此操作将使用 S3 Glacier 的分段上传 API 操作将上传内容拆分为多个部分，以便在将数据流式传输到 S3 Glacier 时出错的情况下更好地进行错误恢复。

有关如何运行以下示例的 step-by-step说明，请参阅运行代码示例。您必须更新文件库名称和待上传档案文件名称旁显示的代码。

> **Note**
>
> S3 Glacier 在文件库中保留一份所有档案的清单。当您上传以下示例中的档案时，该档案直到文件库清单已更新后才会在管理控制台的文件库中显示。此更新通常每天进行一次。

Example — 使用的高级别 API 上传档案 适用于 .NET 的 AWS SDK

```
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();
        }
    }
```

```
}
```

步骤 4：在 S3 Glacier 中从文件库下载档案

在此步骤中，您将下载之前在步骤 3：在 S3 Glacier 中将档案上传到文件库中上传的示例档案。

> **Important**
>
> Amazon S3 Glacier 确实提供了一个控制台。但是，任何存档操作（例如上传、下载或删除）都需要您使用 AWS Command Line Interface (CLI) 或编写代码。存档操作没有控制台支持。例如，要上传数据（例如照片、视频和其他文档），您必须使用 AWS CLI 或编写代码来发出请求，方法是直接使用 REST API 或使用 AWS SDKs。
> 要安装 AWS CLI，请参阅AWS Command Line Interface。有关将 S3 Glacier 与配合使用的更多信息 AWS CLI，请参阅 S3 Glacier AWS CLI 参考。有关使用将档案上传 AWS CLI 到 S3 Glacier 的示例，请参阅将 S3 Glacier 与一起使用 AWS Command Line Interface。

通常，从 S3 Glacier 检索数据是一个分为两个步骤的过程：

1. 启动检索任务。
2. 任务完成后，按字节下载数据。

要从 S3 Glacier 检索档案，您首先要启动任务。任务完成后，您应下载数据。有关存档检索的更多信息，请参阅检索 S3 冰川档案。

您请求的访问时间取决于所选的检索选项：加速、标准还是批量检索。对于除了最大型档案 (250 MB+) 之外的所有其他档案，使用加速检索访问的档案通常在 1 到 5 分钟内可用。使用标准检索来检索的档案通常在 3 到 5 小时内可用。批量检索通常在 5 到 12 小时内可用。有关各个检索选项的更多信息，请参阅 S3 Glacier 常见问题。有关数据检索费用的更多信息，请参阅 S3 Glacier 定价页面。

以下主题中显示的代码示例会启动任务，等待任务完成，然后下载档案的数据。

主题

- 使用 S3 Glacier 中的文件库下载档案 适用于 Java 的 AWS SDK
- 使用 S3 Glacier 中的文件库下载档案 适用于 .NET 的 AWS SDK

使用 S3 Glacier 中的文件库下载档案 适用于 Java 的 AWS SDK

以下 Java 代码示例使用的 适用于 Java 的 AWS SDK 高级别 API 下载您在上一步中上传的档案。在代码示例中，请注意以下情况：

- 以下示例创建 `AmazonGlacierClient` 类的实例。
- 该代码使用美国西部（俄勒冈州）区域 (`us-west-2`) 匹配您之前在步骤 2：在 S3 Glacier 中创建文件库中创建文件库的位置。
- 该示例使用了 `ArchiveTransferManager` 类的 `download` API 操作，该类属于 适用于 Java 的 AWS SDK高级 API。该示例将创建 Amazon Simple Notification Service (Amazon SNS) 主题，以及该主题订阅的 Amazon Simple Queue Service (Amazon SQS) 队列。如果您按照中的说明创建了 AWS Identity and Access Management (IAM) 管理员用户步骤 1：开始使用 S3 Glacier 之前，则您的用户拥有创建和使用 Amazon SNS 主题和 Amazon SQS 队列所必需的 IAM 权限。

有关如何运行此示例的 step-by-step说明，请参阅使用 Eclipse 运行 Amazon S3 Glacier 的 Java 示例。您需要更新 步骤 3：在 S3 Glacier 中将档案上传到文件库 中已上传文件的档案 ID 旁显示的代码。

Example 使用 适用于 Java 的 AWS SDK下载档案

```
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);
        }
    }
}
```

使用 S3 Glacier 中的文件库下载档案 适用于 .NET 的 AWS SDK

以下 C# 代码示例使用的高级别 API 下载您之前在中上传的档案使用以下方法将档案上传到 S3 Glacier 中的文件库 适用于 .NET 的 AWS SDK。 适用于 .NET 的 AWS SDK 在代码示例中，请注意以下情况：

- 该示例为指定的 Amazon S3 Glacier 区域端点创建 `ArchiveTransferManager` 类的实例。
- 该代码示例使用美国西部（俄勒冈州）区域 (`us-west-2`) 匹配您之前在步骤 2：在 S3 Glacier 中创建文件库中创建文件库的位置。
- 该示例使用 `Download` 类的 `ArchiveTransferManager` API 操作下载档案。该示例将创建 Amazon Simple Notification Service (Amazon SNS) 主题，以及该主题订阅的 Amazon Simple Queue Service (Amazon SQS) 队列。如果您按照中的说明创建了 AWS Identity and Access Management (IAM) 管理员用户步骤 1：开始使用 S3 Glacier 之前，则您的用户拥有创建和使用 Amazon SNS 主题和 Amazon SQS 队列所必需的 IAM 权限。

- 此示例启动了存档检索任务，并对队列进行轮询以便找到可用存档。如果档案可用，则开始下载。有关检索时间的详细信息，请参阅 档案检索选项。

有关如何运行此示例的 step-by-step说明，请参阅运行代码示例。您需要更新 步骤 3：在 S3 Glacier 中将档案上传到文件库 中已上传文件的档案 ID 旁显示的代码。

Example — 使用的高级别 API 下载档案 适用于 .NET 的 AWS SDK

```
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);
            }
        }
    }
}
```

步骤 5：从 S3 Glacier 中的文件库删除档案

在此步骤中，您将删除在步骤 3：在 S3 Glacier 中将档案上传到文件库中上传的示例档案。

> **Important**
>
> 无法使用 Amazon S3 Glacier 控制台删除档案。任何存档操作（例如上传、下载或删除）都需要您使用 AWS Command Line Interface (CLI) 或编写代码。要上传数据（例如照片、视频和其他文档），您必须使用 AWS CLI 或编写代码来发出请求，方法是直接使用 REST API 或使用 AWS SDKs。
>
> 要安装 AWS CLI，请参阅AWS Command Line Interface。有关将 S3 Glacier 与配合使用的更多信息 AWS CLI，请参阅 S3 Glacier AWS CLI 参考。有关使用将档案上传 AWS CLI 到 S3 Glacier 的示例，请参阅将 S3 Glacier 与一起使用 AWS Command Line Interface。

通过以下任一操作 SDKs 或以下方法删除示例存档 AWS CLI：

- 使用 S3 Glacier 中的文件库中删除档案 适用于 Java 的 AWS SDK
- 使用 S3 Glacier 中的文件库中删除档案 适用于 .NET 的 AWS SDK
- 使用 AWS CLI在 S3 Glacier 中删除档案

相关部分

- 步骤 3：在 S3 Glacier 中将档案上传到文件库
- 删除 Amazon S3 Glacier 中的档案

使用 S3 Glacier 中的文件库中删除档案 适用于 Java 的 AWS SDK

以下代码示例使用 适用于 Java 的 AWS SDK 删除档案。在代码中，请注意以下情况：

- `DeleteArchiveRequest` 数据元描述删除请求，包括档案所在的文件库名称和档案 ID。
- `deleteArchive` API 操作向 Amazon S3 Glacier 发送删除档案的请求。
- 该示例使用美国西部（俄勒冈州）区域 (`us-west-2`)。

有关如何运行此示例的 step-by-step说明，请参阅使用 Eclipse 运行 Amazon S3 Glacier 的 Java 示例。您需要更新 步骤 3：在 S3 Glacier 中将档案上传到文件库 中已上传文件的档案 ID 旁显示的代码。

Example —使用 适用于 Java 的 AWS SDK删除档案

```
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);
        }
    }
}
```

使用 S3 Glacier 中的文件库中删除档案 适用于 .NET 的 AWS SDK

以下 C# 代码示例使用的 适用于 .NET 的 AWS SDK 高级别 API 删除您在上一步中上传的档案。在代码示例中，请注意以下情况：

- 该示例为指定的 Amazon S3 Glacier 区域端点创建 `ArchiveTransferManager` 类的实例。
- 该代码示例使用美国西部（俄勒冈州）区域 (`us-west-2`)。
- 该示例使用 `ArchiveTransferManager` 类的 `Delete` API 操作，该类属于 适用于 .NET 的 AWS SDK的高级 API。

有关如何运行此示例的 step-by-step说明，请参阅运行代码示例。您需要更新 步骤 3：在 S3 Glacier 中将档案上传到文件库 中已上传文件的档案 ID 旁显示的代码。

Example — 使用的高级别 API 删除档案 适用于 .NET 的 AWS SDK

```
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();
    }
  }
}
```

使用 AWS CLI在 S3 Glacier 中删除档案

您可以使用 AWS Command Line Interface (AWS CLI) 删除 Amazon S3 Glacier 中的档案。

主题

- （先决条件）设置 AWS CLI
- 示例：使用删除档案 AWS CLI

（先决条件）设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   正在安装 AWS Command Line Interface

   正在配置 AWS Command Line Interface

2. 在命令提示符下输入以下命令来验证您的 AWS CLI 设置。这些命令没有显式提供凭证，因此将使用默认配置文件的凭证。

   - 尝试使用 help 命令。

```
aws help
```

- 要获取已配置账户上 S3 Glacier 文件库的列表，请使用 `list-vaults` 命令。*123456789012*用您的 AWS 账户 身份证替换。

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

- 要查看的当前配置数据 AWS CLI，请使用`aws configure list`命令。

```
aws configure list
```

示例：使用删除档案 AWS CLI

1. 使用 `initiate-job` 命令启动清单检索作业。有关 `initiate-job` 命令的更多信息，请参阅启动任务。

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

预期输出：

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

2. 使用 `describe-job` 命令检查上一个检索任务的状态。有关 `describe-job` 命令的更多信息，请参阅描述任务。

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

预期输出：

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
    },
```

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

3. 等待任务完成。

   您必须等到任务输出已作好供您下载的准备。如果您在文件库中设置了通知配置，或者在启动任务时指定了 Amazon Simple Notification Service (Amazon SNS) 主题，则 S3 Glacier 会在完成任务后向该主题发送消息。

   您可以设置文件库的特定事件的通知配置。有关更多信息，请参阅 在 Amazon S3 Glacier 中配置文件库通知。只要发生特定事件，S3 Glacier 就会向指定的 Amazon SNS 主题发送消息。

4. 任务完成后，使用 `get-job-output` 命令将检索任务下载到文件 `output.json`。有关 `get-job-output` 命令的更多信息，请参阅获取任务输出。

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

   此命令会生成一个包含以下字段的文件。

```
{
"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. 使用 `delete-archive` 命令从文件库中删除每个存档，直到不保留任何存档。

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

有关 `delete-archive` 命令的更多信息，请参阅删除档案。

步骤 6：在 S3 Glacier 中删除文件库

文件库是用于存储存档的容器。要删除 Amazon S3 Glacier 文件库，您必须首先删除文件库中截至 S3 Glacier 计算的最后清单的所有现有档案。

您可以通过编程方式或使用 S3 Glacier 控制台删除文件库。有关以编程方式删除文件库的信息，请参阅在 Amazon S3 Glacier 中删除文件库。

> **Important**
>
> 如果您在最近 24 小时内将档案上传到文件库或从文件库中删除档案，则必须等到最后一次更新文件库清单以反映最新信息。S3 Glacier 每 24 小时会定期为每个文件库准备一份清单。

删除空文件库

1. 登录 AWS Management Console 并在https://console.aws.amazon.com/glacier/家中打开 S3 Glacier 控制台。
2. 从 “选择区域” 菜单中， AWS 区域 为要删除的文件库选择。

   在此入门练习中，您的示例文件库位于美国西部（俄勒冈州）区域。
3. 选择要删除的空文件库旁边的选项按钮。如果文件库不为空，则必须先删除所有档案，然后才能删除文件库。有关更多信息，请参阅 删除 Amazon S3 Glacier 中的档案。

   > **Important**
   >
   > 删除文件库的操作无法撤消。
4. 选择删除。
5. 此时显示删除文件库对话框。选择删除。

删除非空文件库

1. 如果要删除非空文件库，则必须先删除所有现有档案，然后再删除文件库。为此，您可以编写代码，使用 REST API、、 适用于 .NET 的 AWS SDK 或，发出删除档案请求 AWS CLI。 适用于 Java 的 AWS SDK有关删除存档的信息，请参阅步骤 5：从 S3 Glacier 中的文件库删除档案。
2. 文件库为空后，按照上述步骤删除空文件库。

我从这里可以了解哪些内容？

至此，您已完成了入门练习，您可以探索以下部分以了解有关 Amazon S3 Glacier 的更多信息。

- 在 Amazon S3 Glacier 中处理文件库
- 在 Amazon S3 Glacier 中处理档案

在 Amazon S3 Glacier 中处理文件库

文件库是用于存储存档的容器。创建文件库时，需要指定文件库名称和要 AWS 区域 在其中创建文件库。有关 S3 Glacier 支持的 AWS 区域 列表，请参阅《AWS 一般参考》中的 Amazon S3 Glacier 端点和配额。

您可以在文件库中存储无限多个档案。

> **Important**
>
> S3 Glacier 确实提供了一个控制台。但是，任何存档操作（例如上传、下载或删除）都需要您使用 AWS Command Line Interface (AWS CLI) 或编写代码。存档操作没有控制台支持。例如，要上传数据（例如照片、视频和其他文档），您必须使用 AWS CLI 或编写代码来发出请求，方法是直接使用 REST API 或使用 AWS SDKs。
> 要安装 AWS CLI，请参阅AWS Command Line Interface。有关将 S3 Glacier 与配合使用的更多信息 AWS CLI，请参阅 S3 Glacier AWS CLI 参考资料。有关使用将档案上传 AWS CLI 到 S3 Glacier 的示例，请参阅将 S3 Glacier 与 AWS Command Line Interface。

主题

- S3 Glacier 中的文件库操作
- 在 Amazon S3 Glacier 中创建文件库
- 在 Amazon S3 Glacier 文件库中检索文件库元数据
- 在 Amazon S3 Glacier 中下载文件库清单
- 在 Amazon S3 Glacier 中配置文件库通知
- 在 Amazon S3 Glacier 中删除文件库
- 标记 S3 Glacier 文件库
- S3 Glacier 文件库锁定

S3 Glacier 中的文件库操作

S3 Glacier 支持各种文件库操作。文件库操作取决于特定 AWS 区域。换而言之，创建文件库时，您在特定 AWS 区域中创建。在您列出文件库时，S3 Glacier 会从您在请求中指定的 AWS 区域 返回文件库列表。

创建和删除文件库

每个文件库最多 AWS 账户 可以创建 1,000 个。 AWS 区域有关 S3 Glacier 支持的 AWS 区域 列表，请参阅《AWS 一般参考》中的 Amazon S3 Glacier 端点和配额。

仅当自 S3 Glacier 计算的上次清单起文件库中没有任何档案，并且自上次清单盘点以来没有对文件库执行过任何写入操作时，您才能删除文件库。

> **Note**
>
> S3 Glacier 每 24 小时会定期为每个文件库准备一份清单。由于清单可能没有反映最新信息，因此，S3 Glacier 会通过检查自上次文件库清单盘点以来是否执行过任何写入操作来确保文件库确实是空的。

有关更多信息，请参阅在 Amazon S3 Glacier 中创建文件库 和在 Amazon S3 Glacier 中删除文件库。

检索文件库元数据

您可以检索文件库信息，例如文件库的创建日期、文件库中的档案数，以及文件库中所有档案的总大小。S3 Glacier 提供 API 调用，供您检索账户中特定文件库或特定文件库中所有文件库 AWS 区域 的此信息。有关更多信息，请参阅 在 Amazon S3 Glacier 文件库中检索文件库元数据。

下载文件库清单

文件库清单指的是文件库中的档案列表。对于列表中的每个档案，清单都提供了档案信息，例如档案 ID、创建日期和大小。从您将第一个档案上传到文件库的日期开始，S3 Glacier 大约每天都会更新一次文件库清单。文件库清单必须存在，您才能下载它。

下载文件库清单是一种异步操作。您必须先启动下载清单的任务。收到任务请求后，S3 Glacier 会为下载准备清单。任务完成后，您可以下载清单数据。

鉴于任务具有异步性，您可以使用 Amazon Simple Notification Service (Amazon SNS) 通知在任务完成时通知您。您可以为每个任务请求指定 Amazon SNS 主题，或者将您的文件库配置为在特定文件库事件发生时发送通知。

S3 Glacier 每 24 小时会定期为每个文件库准备一份清单。如果在上次清单盘点后没有对文件库执行过添加或删除档案的操作，则不会更新库存日期。

当您启动文件库清单任务时，S3 Glacier 会返回其生成的最后一个清单，即 point-in-time快照而不是实时数据。您可能没有发现为每个档案上传操作检索文件库清单有什么好处。但是，假设您在客户端维护数据库，且该数据库中包含与您上传到 S3 Glacier 的档案关联的元数据。此时，您可能会发现，文件库清单存对于将您数据库中的信息与实际文件库清单进行协调很有用。

有关检索文件库清单的更多信息，请参阅在 Amazon S3 Glacier 中下载文件库清单。

配置文件库通知

从 S3 Glacier 检索任何内容（例如文件库中的档案或文件库清单）是一个分为两步的过程。首先，启动一项任务。任务完成后，下载输出。要了解您的任务何时完成，您可以使用 S3 Glacier 通知。S3 Glacier 会将通知消息发送到您提供的 Amazon Simple Notification Service (Amazon SNS) 主题。

您可以配置文件库通知，并确定文件库事件以及要在事件发生时通知的 Amazon SNS 主题。每当有文件库事件发生时，S3 Glacier 都会向指定的 Amazon SNS 主题发送通知。有关更多信息，请参阅 在 Amazon S3 Glacier 中配置文件库通知。

在 Amazon S3 Glacier 中创建文件库

创建文件库的操作会向您账户中的文件库集合添加文件库。每个 AWS 区域最多 AWS 账户 可以创建 1,000 个文件库。有关 Amazon S3 Glacier（S3 Glacier）支持的 AWS 区域列表，请参阅AWS 一般参考中的区域和终端节点。

创建文件库时，您必须提供文件库名称。以下是文件库的命名要求：

- 名称长度在 1 和 255 个字符之间。
- 允许的字符包括 a-z、A-Z、0-9、'-'（下划线）、'_'（连字符）和 '.'（半角句点）。

在一个账户和创建文件库的 AWS 区域中，文件库名称必须是唯一的。也就是说，一个账户可以在不同的 AWS 区域创建同名保管库，但不能在同一个区域中创建同 AWS 名保管库。

主题

- 使用 Amazon S3 Glacier 创建文件库 适用于 Java 的 AWS SDK
- 在 Amazon S3 Glacier 中使用创建文件库 适用于 .NET 的 AWS SDK
- 使用 REST API 在 Amazon S3 Glacier 中创建文件库
- 使用 Amazon S3 Glacier 控制台创建文件库

- 使用 Amazon S3 Glacier 创建文件库 AWS Command Line Interface

使用 Amazon S3 Glacier 创建文件库 适用于 Java 的 AWS SDK

低级 API 为所有文件库操作提供了方法，包括创建和删除文件库、获取文件库描述以及获取在特定文件库中创建的文件库的列表。 AWS 区域以下是使用 适用于 Java 的 AWS SDK创建文件库的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定要 AWS 区域 在其中创建文件库的。您使用此客户端执行的所有操作都会应用到该 AWS 区域。

2. 通过创建一个 `CreateVaultRequest` 类的实例提供请求信息。

   Amazon S3 Glacier（S3 Glacier）要求您提供文件库名称和您的账户 ID。如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

3. 以参数形式提供请求对象，运行 `createVault` 方法。

   S3 Glacier 返回的响应在 `CreateVaultResult` 对象中提供。

以下 Java 代码段说明了前面的步骤。该代码段在 `us-west-2` 区域创建了文件库。`Location`它打印的是文件库的相对 URI，其中包括您的账户 ID AWS 区域、和文件库名称。

```
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**
>
> 有关底层 REST API 的信息，请参阅创建文件库（设置文件库）。

示例：使用创建文件库 适用于 Java 的 AWS SDK

以下 Java 代码示例在该`us-west-2`地区创建了一个文件库（有关更多信息 AWS 区域，请参阅访问 Amazon S3 Glacier）。此外，该代码示例还会检索文件库信息，列出同一个文件库中的所有文件库 AWS 区域，然后删除创建的文件库。

有关如何运行以下示例的 step-by-step说明，请参阅使用 Eclipse 运行 Amazon S3 Glacier 的 Java 示例。

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);
    }

}
```

在 Amazon S3 Glacier 中使用创建文件库 适用于 .NET 的 AWS SDK

适用于.NET 的 Amazon SDK APIs 提供的高级和低级都提供了一种创建文件库的方法。

主题

- 使用 适用于 .NET 的 AWS SDK高级 API 创建文件库
- 使用的低级 API 创建文件库 适用于 .NET 的 AWS SDK

使用 适用于 .NET 的 AWS SDK高级 API 创建文件库

高级 API 的 `ArchiveTransferManager` 类提供了您可以用来在 AWS 区域创建文件库的 `CreateVault` 方法。

示例：使用高级 API 进行文件库操作 适用于 .NET 的 AWS SDK

以下 C# 代码示例在美国西部（俄勒冈州）区域创建了文件库，然后删除了该文件库。有关可以在其中创建文件库 AWS 区域 的列表，请参阅访问 Amazon S3 Glacier。

有关如何运行以下示例的 step-by-step说明，请参阅运行代码示例。您需要更新文件库名称旁显示的代码。

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();
    }
  }
}
```

使用的低级 API 创建文件库 适用于 .NET 的 AWS SDK

低级 API 为所有文件库操作提供了方法，包括创建和删除文件库、获取文件库描述以及获取在特定文件库中创建的文件库的列表。 AWS 区域以下是使用 适用于 .NET 的 AWS SDK创建文件库的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定要 AWS 区域 在其中创建文件库的。您使用此客户端执行的所有操作都会应用到该 AWS 区域。

2. 通过创建一个 `CreateVaultRequest` 类的实例提供请求信息。

   Amazon S3 Glacier（S3 Glacier）要求您提供文件库名称和您的账户 ID。如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 .NET 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

3. 以参数形式提供请求对象，运行 `CreateVault` 方法。

   S3 Glacier 返回的响应在 `CreateVaultResponse` 对象中提供。

示例：使用的低级 API 进行文件库操作 适用于 .NET 的 AWS SDK

以下 C# 示例说明了前面的步骤。此示例可在美国西部（俄勒冈州）区域创建文件库。此外，该代码示例还会检索文件库信息，列出同一个文件库中的所有文件库 AWS 区域，然后删除创建的文件库。`Location`打印的是文件库的相对 URI，其中包括您的账户 ID AWS 区域、和文件库名称。

> **Note**
>
> 有关底层 REST API 的信息，请参阅创建文件库（设置文件库）。

有关如何运行以下示例的 step-by-step说明，请参阅运行代码示例。您需要更新文件库名称旁显示的代码。

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);
    }
  }
}
```

使用 REST API 在 Amazon S3 Glacier 中创建文件库

要使用 REST API 创建文件库，请参阅“创建文件库（设置文件库）”。

使用 Amazon S3 Glacier 控制台创建文件库

要使用 Amazon S3 Glacier（S3 Glacier）控制台创建文件库，请参阅《入门》教程中的步骤 2：在 S3 Glacier 中创建文件库。

使用 Amazon S3 Glacier 创建文件库 AWS Command Line Interface

按照以下步骤使用 AWS Command Line Interface (AWS CLI) 在 Amazon S3 Glacier（S3 Glacier）中创建文件库。

主题

- （先决条件）设置 AWS CLI
- 示例：使用创建文件库 AWS CLI

（先决条件）设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   正在安装 AWS Command Line Interface

   正在配置 AWS Command Line Interface

2. 在命令提示符下输入以下命令来验证您的 AWS CLI 设置。这些命令没有显式提供凭证，因此将使用默认配置文件的凭证。

   - 尝试使用 help 命令。

     ```
     aws help
     ```

   - 要获取已配置账户上 S3 Glacier 文件库的列表，请使用 `list-vaults` 命令。*123456789012*用您的 AWS 账户 身份证替换。

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

   - 要查看的当前配置数据 AWS CLI，请使用`aws configure list`命令。

     ```
     aws configure list
     ```

示例：使用创建文件库 AWS CLI

1. 使用`create-vault`命令创建名为 *awsexamplevault* account 的保管库*111122223333*。

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

预期输出：

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

2. 使用 `describe-vault` 命令验证创建。

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

在 Amazon S3 Glacier 文件库中检索文件库元数据

您可以检索文件库信息，例如文件库的创建日期、文件库中的档案数，以及文件库中所有档案的总大小。Amazon S3 Glacier (S3 Glacier) 提供 API 调用，供您检索账户中特定文件库或特定 AWS 区域内所有文件库的此信息。

如果您检索文件库列表，S3 Glacier 将返回按文件库名称的 ASCII 值排序的列表。该列表最多包含 1000 个文件库。您应始终检查响应，以查看是否有继续该列表的标记；如果没有更多项目，则标记字段为 `null`。您可以选择性地限制响应中返回的文件库数。如果实际的文件库数大于响应中返回的文件库数，则结果会分页。您需要发送附加请求来获取下一组文件库。

主题

- 使用 Amazon S3 Glacier 检索文件库元数据 适用于 Java 的 AWS SDK
- 使用 Amazon S3 Glacier 中检索文件库元数据 适用于 .NET 的 AWS SDK
- 使用 REST API 检索文件库元数据
- 使用 Amazon S3 Glacier 中检索文件库元数据 AWS Command Line Interface

使用 Amazon S3 Glacier 检索文件库元数据 适用于 Java 的 AWS SDK

主题

- 检索文件库的文件库元数据

- 检索一个区域所有文件库的文件库元数据
- 示例：使用适用于 Java 的 Amazon SDK 检索文件库元数据

检索文件库的文件库元数据

您可以检索特定文件库或特定 AWS 区域内所有文件库的元数据。以下是使用适用于 Java 的 Amazon SDK 低级 API 检索特定文件库的文件库元数据的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定文件库所在的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过创建一个 `DescribeVaultRequest` 类的实例提供请求信息。

   Amazon S3 Glacier（S3 Glacier）要求您提供文件库名称和您的账户 ID。如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

3. 以参数形式提供请求对象，运行 `describeVault` 方法。

   S3 Glacier 返回的文件库元数据信息在 `DescribeVaultResult` 对象中提供。

以下 Java 代码段说明了前面的步骤。

```
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**
>
> 有关底层 REST API 的信息，请参阅描述文件库（获取文件库）。

检索一个区域所有文件库的文件库元数据

您还可以使用该`listVaults`方法检索特定 AWS 区域中所有文件库的元数据。

以下 Java 代码段会检索 `us-west-2` 区域的文件库的列表。该请求会将响应中返回的文件库数限制为 5 个。然后，该代码段会进行一系列`listVaults`调用，以从该 AWS 地区检索整个保管库列表。

```
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);
```

在前面的代码段中，如果您在请求中未指定 `Limit` 值，则 S3 Glacier 最多返回 S3 Glacier API 设置的 10 个文件库。如果有更多文件库要列出，则响应 `marker` 字段会包含文件库的Amazon 资源名称 (ARN)，新请求会从该名称处继续列表；否则，`marker` 字段为空。

请注意，列表中返回的每个文件库的信息与您通过调用特定文件库的 `describeVault` 方法获取的信息相同。

> **Note**
>
> `listVaults` 方法会调用底层 REST API（请参阅“列出文件库（获取文件库）”）。

示例：使用适用于 Java 的 Amazon SDK 检索文件库元数据

有关工作代码示例，请参阅“示例：使用创建文件库 适用于 Java 的 AWS SDK”。该 Java 代码示例会创建文件库并检索文件库元数据。

使用 Amazon S3 Glacier 中检索文件库元数据 适用于 .NET 的 AWS SDK

主题

- 检索文件库的文件库元数据
- 检索一个区域所有文件库的文件库元数据
- 示例：使用的低级 API 检索文件库元数据 适用于 .NET 的 AWS SDK

检索文件库的文件库元数据

您可以检索特定文件库或特定 AWS 区域内所有文件库的元数据。以下是使用 适用于 .NET 的 AWS SDK低级 API 检索特定文件库的文件库元数据的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定文件库所在的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。
2. 通过创建一个 `DescribeVaultRequest` 类的实例提供请求信息。

   Amazon S3 Glacier（S3 Glacier）要求您提供文件库名称和您的账户 ID。如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 .NET 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。
3. 以参数形式提供请求对象，运行 `DescribeVault` 方法。

   S3 Glacier 返回的文件库元数据信息在 `DescribeVaultResult` 对象中提供。

以下 C# 代码段说明了前面的步骤。此代码段检索美国西部（俄勒冈州）区域中现有文件库的元数据信息。

```
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**
>
> 有关底层 REST API 的信息，请参阅描述文件库（获取文件库）。

检索一个区域所有文件库的文件库元数据

您还可以使用该`ListVaults`方法检索特定 AWS 区域中所有文件库的元数据。

以下 C# 代码段检索美国西部（俄勒冈州）区域中的文件库的列表。该请求会将响应中返回的文件库数限制为 5 个。然后，该代码段会进行一系列`ListVaults`调用，以从该 AWS 地区检索整个保管库列表。

```
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);
```

在前面的代码段中，如果您在请求中未指定 `Limit` 值，则 S3 Glacier 最多返回 S3 Glacier API 设置的 10 个文件库。

请注意，列表中返回的每个文件库的信息与您通过调用特定文件库的 `DescribeVault` 方法获取的信息相同。

> **Note**
>
> `ListVaults` 方法会调用底层 REST API（请参阅“列出文件库（获取文件库）”）。

示例：使用的低级 API 检索文件库元数据 适用于 .NET 的 AWS SDK

有关工作代码示例，请参阅“示例：使用的低级 API 进行文件库操作 适用于 .NET 的 AWS SDK”。该 C# 代码示例会创建文件库并检索文件库元数据。

使用 REST API 检索文件库元数据

要使用 REST API 列出文件库，请参阅“列出文件库（获取文件库）”。要描述一个文件库，请参阅“描述文件库（获取文件库）”。

使用 Amazon S3 Glacier 中检索文件库元数据 AWS Command Line Interface

此示例说明如何使用 () 在 Amazon S3 Glacier (S3 Glacier) 中检索文件库信息和元数据。 AWS Command Line Interface AWS CLI

主题

- （先决条件）设置 AWS CLI
- 示例：使用检索文件库元数据 AWS CLI

（先决条件）设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   正在安装 AWS Command Line Interface

   正在配置 AWS Command Line Interface

2. 在命令提示符下输入以下命令来验证您的 AWS CLI 设置。这些命令没有显式提供凭证，因此将使用默认配置文件的凭证。

   - 尝试使用 help 命令。

     ```
     aws help
     ```

   - 要获取已配置账户上 S3 Glacier 文件库的列表，请使用 `list-vaults` 命令。*123456789012*用您的 AWS 账户 身份证替换。

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

   - 要查看的当前配置数据 AWS CLI，请使用`aws configure list`命令。

     ```
     aws configure list
     ```

示例：使用检索文件库元数据 AWS CLI

- 使用`describe-vault`命令描述账户*awsexamplevault*下名为的文件库*111122223333*。

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

在 Amazon S3 Glacier 中下载文件库清单

您向您的文件库上传第一个档案后，Amazon S3 Glacier（S3 Glacier）会自动创建文件库清单，然后大约每天更新它一次。S3 Glacier 创建第一份清单后，通常需要经过半天（最多一天）时间，该清单才可供检索。您可以通过以下流程（该流程分为两个步骤）从 S3 Glacier 检索文件库清单：

1. 使用 启动作业（POST 作业） 操作启动清单检索任务。

   > **Important**
   >
   > 数据检索策略可能导致您启动检索作业的请求失败，并发生 `PolicyEnforcedException` 异常。有关数据检索策略的更多信息，请参阅S3 Glacier 数据检索策略。有关 `PolicyEnforcedException` 异常的更多信息，请参阅错误响应。

2. 在任务完成后，使用 获取任务输出（获取输出） 操作下载字节。

例如，检索档案或文件库清单的操作要求您首先启动检索任务。任务请求会异步运行。当您启动检索任务时，S3 Glacier 会创建任务并在响应中返回任务 ID。S3 Glacier 完成任务时，您可以获取任务输出（档案字节或文件库清单数据）。

任务必须先完成，然后，您才能获取其输出。要确定任务的状态，您有以下选择：

- 等待任务完成通知—您可以指定 S3 Glacier 在完成任务后可以向其发布通知的 Amazon Simple Notification Service (Amazon SNS) 主题。您可以使用以下方法指定 Amazon SNS 主题：
  - 为每个任务指定 Amazon SNS 主题。

    启动任务时，您可以选择性地指定 Amazon SNS 主题。
  - 设置文件库的通知配置。

    您可以设置文件库的特定事件的通知配置（参阅在 Amazon S3 Glacier 中配置文件库通知）。只要发生特定事件，S3 Glacier 就会向指定的 SNS 主题发送消息。

  如果您设置了文件库的通知配置，并且在启动任务时也指定了 Amazon SNS 主题，则 S3 Glacier 会向这两个主题发送任务完成消息。

  您可以将 SNS 主题配置为通过电子邮件通知您或者将消息存储在应用程序可以轮询的 Amazon Simple Queue Service (Amazon SQS) 中。当该队列中出现消息时，您可以检查任务是否已成功完成，然后下载任务输出。

- 显式请求任务信息—S3 Glacier 也提供了描述任务操作 (描述任务（获取任务 ID）)，该操作可让您轮询任务信息。您可以定期发送此请求，以获取任务信息。但是，使用 Amazon SNS 通知才是推荐的选择。

> **Note**
>
> 您通过 SNS 通知获取的信息与通过调用描述任务获取的信息相同。

主题

- 关于库存
- 使用 Amazon S3 Glacier 下载文件库清单 适用于 Java 的 AWS SDK
- 使用 适用于 .NET 的 AWS SDK在 Amazon S3 Glacier 中下载文件库清单
- 使用 REST API 下载文件库清单
- 使用 Amazon S3 Glacier 下载文件库清单 AWS Command Line Interface

关于库存

从您第一次将档案上传到文件库的日期开始，S3 Glacier 至少每天都会更新一次文件库清单。如果在上次清单盘点后没有对文件库执行过添加或删除档案的操作，则不会更新库存日期。当您启动文件库清单任务时，S3 Glacier 会返回其生成的最后一个清单，即 point-in-time快照而不是实时数据。请注意，S3 Glacier 为文件库创建第一份清单后，通常需要经过半天（最多一天）时间，该清单才可用于检索操作。

您可能没有发现为每个档案上传操作检索文件库清单有什么好处。但是，假设您在客户端维护数据库，且该客户端关联了您上传到 S3 Glacier 的档案的元数据。此时，您可能会发现，文件库库存对于根据需要将您数据库中的信息与实际文件库库存进行协调很有用。您可以通过筛选存档创建日期或设置配额，来限制检索的清单项目数。有关限制清单检索的更多信息，请参阅确定清单检索范围。

清单可以按两种格式 (逗号分隔值 (CSV) 或 JSON) 返回。启动清单任务时，您可以选择性地指定格式。默认格式为 JSON。有关清单任务输出中返回的数据字段的更多信息，请参阅“获取任务输出 API”的响应正文。

使用 Amazon S3 Glacier 下载文件库清单 适用于 Java 的 AWS SDK

以下是使用 适用于 Java 的 AWS SDK低级 API 检索文件库清单的步骤。该高级 API 不支持检索文件库库存。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定文件库所在的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过执行 `initiateJob` 方法启动清单检索任务。

   通过在 `initiateJob` 对象中提供任务信息来运行 `InitiateJobRequest`。

   > **Note**
   > 请注意，如果文件库的清单操作尚未完成，则会返回错误。Amazon S3 Glacier（S3 Glacier）每 24 小时会定期为每个文件库准备一份库存。

   作为响应，S3 Glacier 返回任务 ID。该响应位于一个 `InitiateJobResult` 类的实例中。

```
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. 等待作业完成。

   您必须等到任务输出已作好供您下载的准备。如果您在文件库中设置了通知配置，或者在启动任务时指定了 Amazon Simple Notification Service (Amazon SNS) 主题，则 S3 Glacier 会在完成任务后向该主题发送消息。

此外，您还可以通过调用 `describeJob` 方法轮询 S3 Glacier 来确定任务完成状态。但是，使用 Amazon SNS 主题进行通知才是推荐的方法。以下部分给出的代码示例使用适用于 S3 Glacier 的 Amazon SNS 来发布消息。

4. 通过执行 `getJobOutput` 方法下载任务输出（文件库清单数据）。

   您可以通过创建一个 `GetJobOutputRequest` 类的实例来提供您的账户 ID、任务 ID 和文件库名称。如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

   S3 Glacier 返回的输出位于 `GetJobOutputResult` 对象中。

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

> **Note**
>
> 有关任务相关的底层 REST API 的信息，请参阅任务操作。

示例：使用适用于 Java 的 Amazon SDK 检索文件库清单

以下 Java 代码示例会检索指定文件库的文件库清单。

该示例执行以下任务：

- 创建 Amazon Simple Notification Service (Amazon SNS) 主题。

  完成任务后，S3 Glacier 会向此主题发送通知。
- 创建 Amazon Simple Queue Service (Amazon SQS) 队列。

  该示例会向该队列附加策略，以使 Amazon SNS 主题能够向该队列发布消息。
- 启动任务以下载指定的档案。

在任务请求中，指定了创建的 Amazon SNS 主题，以便 S3 Glacier 可以在完成任务后向该主题发布通知。

- 检查 Amazon SQS 队列是否有包含该任务 ID 的消息。

  如果有消息，则分析 JSON，并检查任务是否已成功完成。如果已成功完成，则下载档案。

- 通过删除它创建的 Amazon SNS 主题和 Amazon SQS 队列清除相关数据。

```
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));
    }
}
```

使用 适用于 .NET 的 AWS SDK在 Amazon S3 Glacier 中下载文件库清单

以下是使用 适用于 .NET 的 AWS SDK低级 API 检索文件库清单的步骤。该高级 API 不支持检索文件库库存。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定文件库所在的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过执行 `InitiateJob` 方法启动清单检索任务。

   您在 `InitiateJobRequest` 对象中提供任务信息。作为响应，Amazon S3 Glacier（S3 Glacier）返回任务 ID。该响应位于一个 `InitiateJobResponse` 类的实例中。

```
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. 等待作业完成。

您必须等到任务输出已作好供您下载的准备。如果您在文件库中设置了标识 Amazon Simple Notification Service (Amazon SNS) 主题的通知配置，或者在启动任务时指定了Amazon SNS 主题，则 S3 Glacier 会在完成任务后向该主题发送消息。以下部分给出的代码示例使用适用于 S3 Glacier 的 Amazon SNS 来发布消息。

此外，您还可以通过调用 `DescribeJob` 方法轮询 S3 Glacier 来确定任务完成状态。尽管如此，使用 Amazon SNS 主题进行通知才是推荐的方法。

4. 通过执行 `GetJobOutput` 方法下载任务输出（文件库清单数据）。

   您可以通过创建一个 `GetJobOutputRequest` 类的实例来提供您的账户 ID、文件库名称和任务 ID 信息。如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 .NET 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

   S3 Glacier 返回的输出位于 `GetJobOutputResponse` 对象中。

```
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**
>
> 有关任务相关的底层 REST API 的信息，请参阅任务操作。

示例：使用的低级 API 检索文件库库存 适用于 .NET 的 AWS SDK

以下 C# 代码示例会检索指定文件库的文件库清单。

该示例执行以下任务：

- 设置 Amazon SNS 主题。

  完成任务后，S3 Glacier 会向此主题发送通知。
- 设置 Amazon SQS 队列。

  该示例会向该队列附加策略，以使 Amazon SNS 主题能够发布消息。
- 启动任务以下载指定的档案。

  在任务请求中，该示例会指定 Amazon SNS 主题，以便 S3 Glacier 可以在完成任务后发送消息。
- 定期检查 Amazon SQS 队列是否有消息。

  如果有消息，则分析 JSON，并检查任务是否已成功完成。如果已成功完成，则下载档案。该代码示例使用 JSON.NET 库（请参阅 JSON.NET）来分析 JSON。
- 通过删除它创建的 Amazon SNS 主题和 Amazon SQS 队列清除相关数据。

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);
      }
    }
  }
}
```

使用 REST API 下载文件库清单

使用 REST API 下载文件库清单

下载文件库清单是一个分为两个步骤的流程。

1. 启动 `inventory-retrieval` 类型的任务。有关更多信息，请参阅 启动作业（POST 作业）。
2. 任务完成后，下载清单数据。有关更多信息，请参阅 获取任务输出（获取输出）。

使用 Amazon S3 Glacier 下载文件库清单 AWS Command Line Interface

按照以下步骤操作，使用 AWS Command Line Interface (AWS CLI) 在 Amazon S3 Glacier（S3 Glacier）中下载库存清单。

主题

- （先决条件）设置 AWS CLI
- 示例：使用下载文件库清单 AWS CLI

（先决条件）设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   正在安装 AWS Command Line Interface

   正在配置 AWS Command Line Interface

2. 在命令提示符下输入以下命令来验证您的 AWS CLI 设置。这些命令没有显式提供凭证，因此将使用默认配置文件的凭证。

   - 尝试使用 help 命令。

     ```
     aws help
     ```

   - 要获取已配置账户上 S3 Glacier 文件库的列表，请使用 `list-vaults` 命令。*123456789012*用您的 AWS 账户 身份证替换。

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

   - 要查看的当前配置数据 AWS CLI，请使用`aws configure list`命令。

     ```
     aws configure list
     ```

示例：使用下载文件库清单 AWS CLI

1. 使用 `initiate-job` 命令启动清单检索作业。

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

   预期输出：

   ```
   {
   ```

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

2. 使用 `describe-job` 命令检查上一个检索作业的状态。

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

预期输出：

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

3. 等待作业完成。

   您必须等到任务输出已作好供您下载的准备。S3 Glacier 完成任务后，任务 ID 至少在 24 小时内都不会过期。如果您在文件库中设置了通知配置，或者在启动任务时指定了 Amazon Simple Notification Service (Amazon SNS) 主题，则 S3 Glacier 会在完成任务后向该主题发送消息。

   您可以设置文件库的特定事件的通知配置。有关更多信息，请参阅 在 Amazon S3 Glacier 中配置文件库通知。只要发生特定事件，S3 Glacier 就会向指定的 SNS 主题发送消息。

4. 完成后，使用 `get-job-output` 命令将检索作业下载到文件 `output.json`。

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

此命令会生成一个包含以下字段的文件。

```
{
"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":
...
]}
```

在 Amazon S3 Glacier 中配置文件库通知

从 Amazon S3 Glacier 检索任何内容（例如文件库中的档案或文件库清单）是一个分为两步的过程。

1. 启动检索任务。
2. 任务完成后，下载任务输出。

您可以在文件库上设置通知配置，以便在任务完成时向 Amazon Simple Notification Service (Amazon SNS) 主题发送消息。

主题

- 在 S3 Glacier 中配置文件库通知：一般概念
- 在 Amazon S3 Glacier 中使用 适用于 Java 的 AWS SDK中配置文件库通知
- 在 Amazon S3 Glacier 中使用 适用于 .NET 的 AWS SDK中配置文件库通知
- 使用 REST API 在 S3 Glacier 中配置文件库通知
- 使用 S3 Glacier 控制台配置文件库通知
- 使用 AWS Command Line Interface配置文件库通知

在 S3 Glacier 中配置文件库通知：一般概念

S3 Glacier 检索任务请求是异步执行的。您必须等到 S3 Glacier 完成任务，然后才能获取其输出。您可以定期轮询 S3 Glacier 以确定任务状态，但这不是最佳方法。S3 Glacier 还支持通知。任务完成后，任务可以将消息发布到 Amazon Simple Notification Service (Amazon SNS) 主题。要使用此功能，您必须在文件库上设置通知配置。在配置中，您可以标识一个或多个事件，以及您希望 S3 Glacier 在事件发生时向其发送消息的 Amazon SNS 主题。

S3 Glacier 定义了与任务完成特别有关的事件（`ArchiveRetrievalCompleted`、`InventoryRetrievalCompleted`），您可以将这些事件添加到文件库的通知配置中。当特定任务完成时，S3 Glacier 会向 SNS 主题发布通知消息。

通知配置是 JSON 文档，如以下示例所示。

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

请注意，对于一个文件库，您只能配置一个 Amazon SNS 主题。

> **Note**
>
> 向文件库添加通知配置会使 S3 Glacier 在通知配置中指定的事件每次发生时都发送通知。此外，您还可以选择性地在每个任务启动请求中指定 Amazon SNS 主题。如果您在文件库中添加了通知配置，并且在您的启动任务请求中也指定了 Amazon SNS 主题，则 S3 Glacier 会发送这两种通知。

S3 Glacier 发送的任务完成消息包括任务类型（`InventoryRetrieval`、`ArchiveRetrieval`）、任务完成状态、SNS 主题名称、任务状态代码和文件库 ARN 等信息。以下是 S3 Glacier 在 `InventoryRetrieval` 任务完成后发送到 SNS 主题的示例通知。

```
{
 "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"
}
```

此外，如果 `Completed` 字段为 true，则您还必须检查 `StatusCode` 以查看任务是成功完成了还是失败了。

> **Note**
>
> 请注意，Amazon SNS 主题必须允许文件库发布通知。默认情况下，只有 Amazon SNS 主题拥有者才能向主题发布消息。但是，如果 Amazon SNS 主题和文件库归其他人所有 AWS 账户，则必须将 Amazon SNS 主题配置为接受文件库中的出版物。您可以在 Amazon SNS 控制台中配置 Amazon SNS 主题策略。

有关 Amazon SNS 的更多信息，请参阅 Amazon SNS 入门。

在 Amazon S3 Glacier 中使用 适用于 Java 的 AWS SDK中配置文件库通知

以下是使用 适用于 Java 的 AWS SDK低级 API 在文件库中配置通知的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定文件库所在的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。
2. 通过创建一个 `SetVaultNotificationsRequest` 类的实例提供通知配置信息。

   您需要提供文件库名称、通知配置信息和账户 ID。在指定通知配置时，您可以提供一个现有的 Amazon SNS 主题的 Amazon 资源名称（ARN），以及您希望获得其通知的一个或多个事件。有关受支持的事件的列表，请参阅“设置文件库通知配置（设置通知配置）”。
3. 以参数形式提供请求对象，运行 `setVaultNotifications` 方法。

以下 Java 代码段说明了前面的步骤。该代码段在文件库中设置了通知配置。当 `ArchiveRetrievalCompleted` 事件或 `InventoryRetrievalCompleted` 事件发生时，该配置会请求 Amazon S3 Glacier（S3 Glacier）向指定的 Amazon SNS 主题发送通知。

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

> **Note**
>
> 有关底层 REST API 的信息，请参阅文件库操作。

示例：使用在文件库上设置通知配置 适用于 Java 的 AWS SDK

以下 Java 代码示例设置了文件库的通知配置并删除了配置，然后恢复了配置。有关如何运行以下示例的 step-by-step说明，请参阅适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

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);
    }
}
```

在 Amazon S3 Glacier 中使用 适用于 .NET 的 AWS SDK中配置文件库通知

以下是使用 适用于 .NET 的 AWS SDK低级 API 在文件库中配置通知的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定文件库所在的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过创建一个 `SetVaultNotificationsRequest` 类的实例提供通知配置信息。

   您需要提供文件库名称、通知配置信息和账户 ID。如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 .NET 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

   在指定通知配置时，您可以提供一个现有的 Amazon SNS 主题的 Amazon 资源名称（ARN），以及您希望获得其通知的一个或多个事件。有关受支持的事件的列表，请参阅“设置文件库通知配置（设置通知配置）”。

3. 以参数形式提供请求对象，运行 `SetVaultNotifications` 方法。

4. 在文件库中设置通知配置后，您可以通过调用 `GetVaultNotifications` 方法检索配置信息，也可以通过调用客户端提供的 `DeleteVaultNotifications` 方法删除它。

示例：使用在文件库上设置通知配置 适用于 .NET 的 AWS SDK

以下 C# 代码示例说明了前面的步骤。该示例在美国西部（俄勒冈州）区域的文件库（“`examplevault`”）中设置了通知配置并检索了配置，然后删除了配置。当

`ArchiveRetrievalCompleted` 事件或 `InventoryRetrievalCompleted` 事件发生时，该配置会请求 Amazon S3 Glacier（S3 Glacier）向指定的 Amazon SNS 主题发送通知。

> **Note**
>
> 有关底层 REST API 的信息，请参阅文件库操作。

有关运行以下示例的 step-by-step说明，请参阅运行代码示例。您需要更新该代码并提供现有的文件库名称和 Amazon SNS 主题。

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);
```

```
    }
  }
}
```

使用 REST API 在 S3 Glacier 中配置文件库通知

要使用 REST API 配置文件库通知，请参阅“设置文件库通知配置（设置通知配置）”。此外，您还可以获取文件库通知 (获取文件库通知（获取通知配置）) 以及删除文件库通知 (删除文件库通知（删除通知配置）)。

使用 S3 Glacier 控制台配置文件库通知

此部分描述了如何使用 Amazon S3 Glacier 控制台配置文件库通知。在配置通知时，您可以指定向 Amazon Simple Notification Service (Amazon SNS) 主题发送通知的任务完成事件。除了为文件库配置通知以外，您还可以指定您在启动任务时要向其发布通知的主题。如果您的文件库已配置为针对特定的事件发送通知，并且您还在任务启动请求中配置了通知，则系统会发送两份通知。

配置文件库通知

1. 登录 AWS Management Console 并在https://console.aws.amazon.com/glacier/家中打开 S3 Glacier 控制台。
2. 在左侧导航窗格中，选择文件库。
3. 在文件库列表中，选择一个文件库。
4. 在通知部分中，选择编辑。
5. 在事件通知页面上，选择启用通知。
6. 在通知部分，选择以下 Amazon Simple Notification Service (Amazon SNS) 选项之一，然后按照相应的步骤进行操作：

Amazon SNS 选项	操作
创建新 SNS 主题	1. 选择新创建新 SNS 主题。 2. 对于主题名称，输入新主题的名称。

Amazon SNS 选项	操作
	主题名称最多可以包含 256 个字符。允许使用字母数字字符、连字符 (-) 和下划线 (_)。主题名称在账户中必须是唯一的，并且 AWS 区域。 3. （可选）如果要使用 SMS 消息订阅主题，请输入显示名称的名称。 显示名称最多可以包含 100 个字符。
选择现有的 SNS 主题	1. 选择选择现有的 SNS 主题。 2. 在指定 SNS 主题下，选择以下选项之一： • 从您的 SNS 主题中选择 此时会出现 SNS 主题下拉列表。 从下拉列表中选择一个现有的主题。 • 输入 SNS 主题 ARN 将出现 Amazon SNS 主题 ARN 文本框。 输入 SNS 主题的 Amazon 资源名称（ARN）。SNS 主题 ARN 格式如下： `arn:aws:sns: region:account-id :topic-name` 您可以在 Amazon SNS 控制台中查找 SNS 主题的 ARN：

7. 在事件下，选择一个或两个要发送通知的事件：
   - 要仅在档案检索任务完成时才发送通知，请选择档案检索任务完成。
   - 要仅在文件库清单任务完成时才发送通知，请选择文件库清单检索任务完成。

使用 AWS Command Line Interface配置文件库通知

此部分描述了如何使用 AWS Command Line Interface配置文件库通知。配置通知时，您可以指定任务完成事件，这些事件会触发向 Amazon Simple Notification Service (Amazon SNS) 主题发送通知。除了为文件库配置通知以外，您还可以指定您在启动任务时要向其发布通知的主题。如果您的文件库已配置为针对特定的事件发送通知，并且您在任务启动请求中指定了通知，则系统会发送两份通知。

按照以下步骤使用 AWS CLI配置文件库通知。

主题

- （先决条件）设置 AWS CLI
- 示例：使用配置文件库通知 AWS CLI

（先决条件）设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   正在安装 AWS Command Line Interface

   正在配置 AWS Command Line Interface

2. 在命令提示符下输入以下命令来验证您的 AWS CLI 设置。这些命令没有显式提供凭证，因此将使用默认配置文件的凭证。

   - 尝试使用 help 命令。

     ```
     aws help
     ```

   - 要获取已配置账户上 S3 Glacier 文件库的列表，请使用 `list-vaults` 命令。*123456789012*用您的 AWS 账户 身份证替换。

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

   - 要查看的当前配置数据 AWS CLI，请使用`aws configure list`命令。

     ```
     aws configure list
     ```

示例：使用配置文件库通知 AWS CLI

1. 使用 `set-vault-notifications` 命令配置在文件库发生特定事件时将发送的通知。默认情况下，您不会收到任何通知。

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

2. 通知配置是 JSON 文档，如以下示例所示。

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

有关对 S3 Glacier 使用 Amazon SNS 主题的更多信息，请参阅在 S3 Glacier 中配置文件库通知：一般概念

有关 Amazon SNS 的更多信息，请参阅 Amazon SNS 入门。

在 Amazon S3 Glacier 中删除文件库

只有在自上次计算的清单起文件库中没有任何档案，并且自上次清单盘点以来没有对文件库执行过任何写入操作时，Amazon S3 Glacier（S3 Glacier）才会删除文件库。有关删除存档的信息，请参阅删除 Amazon S3 Glacier 中的档案。有关下载文件库清单，在 Amazon S3 Glacier 中下载文件库清单

> **Note**
>
> S3 Glacier 每 24 小时会定期为每个文件库准备一份清单。由于清单可能没有反映最新信息，因此，S3 Glacier 会通过检查自上次文件库清单盘点以来是否执行过任何写入操作来确保文件库确实是空的。

> **Note**
>
> 有关自动删除文件库存档的信息，请参阅在 Amazon S3 Glacier 中自动删除文件库档案。

主题

- 使用 Amazon S3 Glacier 中删除文件库 适用于 Java 的 AWS SDK
- 使用 Amazon S3 Glacier 中删除文件库 适用于 .NET 的 AWS SDK
- 使用 REST API 在 S3 Glacier 中删除文件库
- 使用 S3 Glacier 控制台删除空文件库
- 使用 Amazon S3 Glacier 中删除文件库 AWS Command Line Interface

使用 Amazon S3 Glacier 中删除文件库 适用于 Java 的 AWS SDK

以下是使用 适用于 Java 的 AWS SDK低级 API 删除文件库的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定要从中删除文件库的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过创建一个 `DeleteVaultRequest` 类的实例提供请求信息。

   您需要提供文件库名称和账户 ID。如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

3. 以参数形式提供请求对象，运行 `deleteVault` 方法。

   Amazon S3 Glacier（S3 Glacier）只会删除空文件库。有关更多信息，请参阅 删除文件库。

以下 Java 代码段说明了前面的步骤。

```
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**
>
> 有关底层 REST API 的信息，请参阅删除文件库。

示例：使用删除文件库 适用于 Java 的 AWS SDK

有关工作代码示例，请参阅“示例：使用创建文件库 适用于 Java 的 AWS SDK”。该 Java 代码示例显示了基本文件库操作，包括创建和删除文件库。

使用 Amazon S3 Glacier 中删除文件库 适用于 .NET 的 AWS SDK

适用于.NET 的 Amazon SDK APIs 提供的高级版本和低级版本都提供了一种删除文件库的方法。

主题

- 使用的高级别 API 删除文件库 适用于 .NET 的 AWS SDK
- 使用中的低级 API 删除文件库 适用于 .NET 的 AWS SDK

使用的高级别 API 删除文件库 适用于 .NET 的 AWS SDK

该高级 API 的 `ArchiveTransferManager` 类提供了您可以用来删除文件库的 `DeleteVault` 方法。

示例：使用的高级别 API 删除文件库 适用于 .NET 的 AWS SDK

有关工作代码示例，请参阅“示例：使用高级 API 进行文件库操作 适用于 .NET 的 AWS SDK”。该 C# 代码示例显示了基本文件库操作，包括创建和删除文件库。

使用中的低级 API 删除文件库 适用于 .NET 的 AWS SDK

以下是使用 适用于 .NET 的 AWS SDK删除文件库的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定要从中删除文件库的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过创建一个 `DeleteVaultRequest` 类的实例提供请求信息。

您需要提供文件库名称和账户 ID。如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 .NET 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

3. 以参数形式提供请求对象，运行 `DeleteVault` 方法。

   Amazon S3 Glacier（S3 Glacier）只会删除空文件库。有关更多信息，请参阅 删除文件库。

以下 C# 代码段说明了前面的步骤。该代码段检索默认 AWS 区域中存在的文件库的元数据信息。

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

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

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

> **Note**
>
> 有关底层 REST API 的信息，请参阅删除文件库。

示例：使用文件库的低级 API 删除文件库 适用于 .NET 的 AWS SDK

有关工作代码示例，请参阅“示例：使用的低级 API 进行文件库操作 适用于 .NET 的 AWS SDK”。该 C# 代码示例显示了基本文件库操作，包括创建和删除文件库。

使用 REST API 在 S3 Glacier 中删除文件库

要使用 REST API 删除文件库，请参阅“删除文件库”。

使用 S3 Glacier 控制台删除空文件库

> **Note**
>
> 在删除文件库之前，必须先删除该文件库中的所有现有档案。为此，您可以使用 REST API、适用于 .NET 的 AWS SDK、或使用 AWS Command Line Interface (AWS CLI) 编写代码来

发出删除档案请求。 适用于 Java 的 AWS SDK有关删除存档的信息，请参阅步骤 5：从 S3 Glacier 中的文件库删除档案。

文件库为空后，可以使用以下步骤将其删除。

使用 Amazon S3 Glacier 控制台删除空文件库

1. 登录 AWS Management Console 并在 S3 Glacier 控制台上打开 S 3 Glacier 控制台。
2. 在 “选择区域” 下，选择文件库 AWS 区域 所在的位置。
3. 在左侧导航窗格中，选择文件库。
4. 在文件库列表中，选择要删除的文件库名称旁边的选项按钮，然后选择页面顶部的删除。
5. 在删除文件库对话框中，选择 删除以确认要删除文件库。

   > **Important**
   >
   > 删除文件库的操作无法撤消。

6. 要验证您是否已删除文件库，请打开文件库列表并输入您删除的文件库的名称。如果无法找到此文件库，则表示删除成功。

使用 Amazon S3 Glacier 中删除文件库 AWS Command Line Interface

可以使用 AWS Command Line Interface (AWS CLI) 删除 Amazon S3 Glacier（S3 Glacier）中的空文件库和非空文件库。

主题

- （先决条件）设置 AWS CLI
- 示例：使用删除空文件库 AWS CLI
- 示例：使用删除非空文件库 AWS CLI

（先决条件）设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   正在安装 AWS Command Line Interface

配置 AWS Command Line Interface

2. 在命令提示符下输入以下命令来验证您的 AWS CLI 设置。这些命令没有显式提供凭证，因此将使用默认配置文件的凭证。

   - 尝试使用 help 命令。

     ```
     aws help
     ```

   - 要获取已配置账户上 S3 Glacier 文件库的列表，请使用 `list-vaults` 命令。*123456789012*用您的 AWS 账户 身份证替换。

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

   - 要查看的当前配置数据 AWS CLI，请使用`aws configure list`命令。

     ```
     aws configure list
     ```

示例：使用删除空文件库 AWS CLI

- 使用 `delete-vault` 命令删除不包含存档的文件库。

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

示例：使用删除非空文件库 AWS CLI

只有在自上次计算的清单起文件库中没有任何档案，并且自上次清单盘点以来没有对文件库执行过任何写入操作时，S3 Glacier 才会删除文件库。删除非空文件库分为三个步骤： IDs从文件库的库存报告中检索档案，删除每个档案，然后删除文件库。

1. 使用 `initiate-job` 命令启动清单检索作业。

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

   预期输出：
   ```

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

2. 使用 `describe-job` 命令检查上一个检索作业的状态。

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

预期输出：

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

3. 等待作业完成。

   您必须等到任务输出已作好供您下载的准备。如果您在文件库中设置了通知配置，或者在启动任务时指定了 Amazon Simple Notification Service (Amazon SNS) 主题，则 S3 Glacier 会在完成任务后向该主题发送消息。

   您可以设置文件库的特定事件的通知配置。有关更多信息，请参阅 在 Amazon S3 Glacier 中配置文件库通知。只要发生特定事件，S3 Glacier 就会向指定的 SNS 主题发送消息。

4. 完成后，使用 `get-job-output` 命令将检索作业下载到文件 `output.json`。

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

此命令会生成一个包含以下字段的文件。

```
{
"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. 使用 `delete-archive` 命令从文件库中删除每个存档，直到不保留任何存档。

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

> **Note**
>
> 如果您的档案 ID 以连字符或其他特殊字符开头，则必须将档案 ID 用引号括起来才能运行此命令。

6. 使用 `initiate-job` 命令启动新的库存检索作业。

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

7. 完成后，使用 `delete-vault` 命令删除不带存档的文件库。

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

标记 S3 Glacier 文件库

您可以标签形式将自己的元数据分配给 Amazon S3 Glacier 文件库。标签 是您为文件库定义的键-值对。有关标记的基本信息（包括对标签的限制），请参阅标记 Amazon S3 Glacier 资源。

以下主题介绍了如何为文件库添加、列出和删除标签。

主题

- 使用 Amazon S3 Glacier 控制台标记文件库
- 使用标记文件库 AWS CLI
- 使用 Amazon S3 Glacier API 标记文件库
- 相关部分

使用 Amazon S3 Glacier 控制台标记文件库

可以使用 S3 Glacier 控制台添加、列出和删除标签，如下列步骤中所述。

查看文件库的标签

1. 登录 AWS Management Console 并在https://console.aws.amazon.com/glacier/家中打开 S3 Glacier 控制台。
2. 在 “选择区域” 下， AWS 区域 从 “区域” 选择器中选择。
3. 在左侧导航窗格中，选择文件库。
4. 在文件库列表中，选择一个文件库。
5. 选择文件库属性选项卡。滚动至标签部分，查看与文件库关联的标签。

向文件库添加标签

一个文件库最多可以关联 50 个标签。与文件库关联的标签必须具有唯一的标签键。

有关标签限制的更多信息，请参阅标记 Amazon S3 Glacier 资源。

1. 登录 AWS Management Console 并在https://console.aws.amazon.com/glacier/家中打开 S3 Glacier 控制台。
2. 在 “选择区域” 下， AWS 区域 从 “区域” 选择器中选择。
3. 在左侧导航窗格中，选择文件库。
4. 在文件库列表中，选择要将标签添加到的文件库的名称。
5. 选择文件库属性选项卡。
6. 在 Tags (标签) 部分中，选择 Add (添加)。此时将显示添加标签页面。

7. 在添加标签页面的键字段中指定标签键，然后也可以选择在值字段中指定标签值。
8. 选择 Save changes（保存更改）。

编辑标签

1. 登录 AWS Management Console 并在https://console.aws.amazon.com/glacier/家中打开 S3 Glacier 控制台。
2. 在 “选择区域” 下， AWS 区域 从 “区域” 选择器中选择。
3. 在左侧导航窗格中，选择文件库。
4. 在文件库列表中，选择文件库名称。
5. 选择文件库属性选项卡，然后向下滚动到标签部分。
6. 在标签下，选中要更改的标签旁边的复选框，然后选择编辑。将出现编辑标签页面。
7. 更新键字段中的标签，也可以选择更新值字段中的标签。
8. 选择 Save changes（保存更改）。

从文件库中删除标签

1. 登录 AWS Management Console 并在https://console.aws.amazon.com/glacier/家中打开 S3 Glacier 控制台。
2. 在 “选择区域” 下， AWS 区域 从 “区域” 选择器中选择。
3. 在左侧导航窗格中，选择文件库。
4. 在文件库列表中，选择要从中删除标签的文件库的名称。
5. 选择文件库属性选项卡。向下滚动到标签部分。
6. 在标签下，选中要删除的标签旁边的复选框，然后选择删除。
7. 此时将打开删除标签对话框。要确认删除已选择的标签，请选择删除。

使用标记文件库 AWS CLI

按照以下步骤使用 AWS Command Line Interface (AWS CLI) 添加、列出或移除标签。

每个标签由一个键和一个值组成。每个文件库可最多有 50 个标签。

1. 要将标签添加到文件库，请使用 `add-tags-to-vault` 命令。

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

有关此文件库操作的更多信息，请参阅向文件库添加标签。

2. 要列出已附加到文件库的所有标签，请使用 `list-tags-for-vault` 命令。

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

有关此文件库操作的更多信息，请参阅列出文件库的标签。

3. 要从已附加到文件库的标签集中删除一个或多个标签，请使用 `remove-tags-from-vault` 命令。

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

有关此文件库操作的更多信息，请参阅删除文件库中的标签。

使用 Amazon S3 Glacier API 标记文件库

您可以使用 S3 Glacier API 添加、列出和删除标签。有关示例，请参阅以下文档：

向文件库添加标签（POST 标签添加）

为指定文件库添加或更新标签。

列出文件库的标签（GET 标签）

列出指定文件库的标签。

从文件库删除标签（POST 标签删除）

从指定文件库中删除标签。

相关部分

- 标记 Amazon S3 Glacier 资源

S3 Glacier 文件库锁定

以下主题介绍了如何在 Amazon S3 Glacier 中锁定文件库以及如何使用文件库锁定策略。

主题

- 文件库锁定概述
- 使用 S3 Glacier API 锁定文件库
- 使用锁定文件库 AWS Command Line Interface
- 使用 S3 Glacier 控制台锁定文件库

文件库锁定概述

通过 S3 Glacier 文件库锁定，您可以轻松利用文件库锁定策略对单独的 S3 Glacier 文件库进行部署并实施合规性控制措施。您可以在一个文件库锁定策略中指定类似“一次写入，多次读取”(WORM) 这样的控制措施，并且可以锁定该策略以防止将来进行编辑。

> **Important**
>
> 文件库锁定策略被锁定后，将无法再更改或删除该策略。

S3 Glacier 实施文件库锁定策略中的控制集，以帮助实现合规性目标。例如，您可以使用文件库锁定策略来强制实施数据留存。您可以使用 AWS Identity and Access Management (IAM) 策略语言在文件库锁定策略中部署各种合规性控制。有关文件库锁定策略的更多信息，请参阅文件库锁定策略。

文件库锁定策略与文件库访问策略不同。但两者都管理对文件库的访问控制权。但是，文件库锁定策略可进行锁定以防止将来更改，从而强有力地实施您的合规性控制措施。您可以使用文件库锁定策略来部署法规和合规性控制措施，这通常需要对数据访问进行严密控制。

> **Important**
>
> 建议您先创建文件库，完成文件库锁定策略，然后将您的档案上传到文件库，以便将该策略应用于它们。

相反，您可使用文件库访问策略来实施与合规性无关、临时以及需要经常修改的访问控制。文件库锁定策略和文件库访问策略可一起使用。例如，您可在文件库锁定策略中实施基于时间的数据保留规则（拒

绝删除），并且在文件库访问策略中为指定的第三方或您的业务合作伙伴授予读取访问权限（允许读取）。

锁定文件库需要执行两个步骤：

1. 通过将文件库锁定策略附加到您的文件库来启动锁定，这会将锁定设置为正在进行状态并返回一个锁定 ID。当策略处于正在进行状态时，在锁定 ID 到期前，您有 24 个小时来验证文件库锁定策略。为防止您的文件库退出正在进行状态，必须在 24 小时内完成文件库锁定流程。否则，您的文件库锁定策略将被删除。
2. 使用锁定 ID 完成锁定过程。如果文件库锁定策略未按预期工作，可停止文件库锁定并从头开始重新启动。有关如何使用 S3 Glacier API 来锁定文件库的信息，请参阅使用 S3 Glacier API 锁定文件库。

使用 S3 Glacier API 锁定文件库

要使用 Amazon S3 Glacier API 锁定文件库，请先使用指定了待部署控件的文件库锁定策略调用 启动文件库锁定（POST 锁定策略）。`Initiate Vault Lock` 操作会将策略附加到您的文件库，将文件库锁定转换为进行中状态，并返回一个唯一的锁定 ID。在文件库锁定进入进行中状态后，您有 24 小时的时间来利用从 `Initiate Vault Lock` 调用返回的锁定 ID 调用 完成文件库锁定 (POST lockId)，以便完成锁定。

> **Important**
>
> - 建议您先创建文件库，完成文件库锁定策略，然后将您的档案上传到文件库，以便将该策略应用于它们。
> - 文件库锁定策略在被锁定后即不能更改或删除。

如果您在进入正在进行状态后的 24 个小时内未完成文件库锁定过程，则您的文件库会自动退出正在进行状态，并删除文件库锁定策略。您可以再次调用 `Initiate Vault Lock` 来安装新的文件库锁定策略并转换到正在进行状态。

利用正在进行状态，您有机会在锁定您的文件库锁定策略之前对其进行测试。您的文件库锁定策略在正在进行状态期间将会完全生效，就如同文件库已锁定一样，只不过您可以通过调用 中止文件库锁定（DELETE 锁定策略） 来删除该策略。要优化您的策略，可根据需要多次重复 `Abort Vault Lock`/`Initiate Vault Lock` 组合，验证您的文件库锁定策略更改。

在验证文件库锁定策略之后，您可使用最新的锁定 ID 调用 完成文件库锁定 (POST lockId)，以便完成文件库锁定过程。您的文件库会转换为锁定状态（此时，文件库锁定策略不可更改），而且无法再通过调用 `Abort Vault Lock` 进行删除。

相关部分

- 文件库锁定策略
- 中止文件库锁定（DELETE 锁定策略）
- 完成文件库锁定 (POST lockId)
- 获取文件库锁定（GET 锁定策略）
- 启动文件库锁定（POST 锁定策略）

使用锁定文件库 AWS Command Line Interface

您可以使用锁定您的保管库 AWS Command Line Interface。这将对指定的文件库实施文件库锁定策略并返回锁定 ID。您必须在 24 小时内完成文件库锁定过程，否则该文件库锁定策略将从文件库中删除。

（先决条件）设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   正在安装 AWS Command Line Interface

   正在配置 AWS Command Line Interface
2. 在命令提示符下输入以下命令来验证您的 AWS CLI 设置。这些命令没有显式提供凭证，因此将使用默认配置文件的凭证。

   - 尝试使用 help 命令。

     ```
     aws help
     ```

   - 要获取已配置账户上 S3 Glacier 文件库的列表，请使用 `list-vaults` 命令。*123456789012*用您的 AWS 账户 身份证替换。

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

   - 要查看的当前配置数据 AWS CLI，请使用`aws configure list`命令。

```
aws configure list
```

1. 使用 `initiate-vault-lock` 实施文件库锁定策略，并将文件库锁定的锁定状态设置为 `InProgress`。

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

2. 锁定配置是 JSON 文档，如以下示例所示。在使用此命令之前，请将*VAULT_ARN*和*Principal*替换为适合您的用例的值。

   要查找要锁定的文件库的 ARN，可以使用 `list-vaults` 命令。

```
{"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. 启动文件库锁定后，您应该会看到 `lockId` 返回的内容。

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

要完成文件库锁定，您必须在 24 小时内运行 `complete-vault-lock`，否则该文件库锁定策略将从文件库中删除。

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

相关部分

- 《AWS CLI Command Reference》中的 initiate-vault-lock。
- 《AWS CLI 命令参考》中的 list-vaults
- 《AWS CLI Command Reference》中的 complete-vault-lock。
- 文件库锁定策略

- 中止文件库锁定（DELETE 锁定策略）
- 完成文件库锁定 (POST lockId)
- 获取文件库锁定（GET 锁定策略）
- 启动文件库锁定（POST 锁定策略）

使用 S3 Glacier 控制台锁定文件库

通过 Amazon S3 Glacier 文件库锁定，您可以轻松利用文件库锁定策略对单独的 S3 Glacier 文件库进行部署并实施合规性控制。有关 S3 Glacier 文件库锁定的更多信息，请参阅使用文件库锁定策略进行 Amazon S3 Glacier 访问控制。

> **Important**
>
> - 建议您先创建文件库，完成文件库锁定策略，然后将您的档案上传到文件库，以便将该策略应用于它们。
> - 文件库锁定策略在被锁定后即不能更改或删除。

使用 S3 Glacier 控制台对您的文件库启动文件库锁定策略

通过将文件库锁定策略附加到您的文件库来启动锁定，这会将锁定设置为正在进行状态并返回一个锁定 ID。当策略处于正在进行状态时，在锁定 ID 到期前，您有 24 个小时来验证文件库锁定策略。

1. 登录 AWS Management Console 并在https://console.aws.amazon.com/glacier/家中打开 S3 Glacier 控制台。
2. 在 “选择区域” 下， AWS 区域 从 “区域” 选择器中选择。
3. 在左侧导航窗格中，选择文件库。
4. 在文件库页面上，选择创建文件库。
5. 创建新的文件库。

   > **Important**
   >
   > 建议您先创建文件库，完成文件库锁定策略，然后将您的档案上传到文件库，以便将该策略应用于它们。

6. 从文件库列表中选择您的新文件库。

7. 选择文件库策略选项卡。
8. 在文件库锁定策略部分，选择启动文件库锁定策略。
9. 在启动文件库锁定策略页面上，可以使用标准文本框在文本格式的文件库锁定策略中指定记录保留控制措施。

> **Note**
>
> 您可以在文本格式的文件库锁定策略中指定记录保留控制措施，并通过调用 `Initiate Vault Lock` API 操作或通过 S3 Glacier 控制台中的交互式用户界面来启动文件库锁定。有关设置文件库锁定策略的格式的信息，请参阅 Amazon S3 Glacier 文件库锁定策略示例。

10. 选择 Save changes（保存更改）。
11. 在记录文件库锁定 ID 对话框中，复制您的锁定 ID 并将其保存在安全的地方。

> **Important**
>
> 文件库锁定策略启动后，您有 24 小时的时间来验证该策略并完成锁定过程。要完成锁定过程，必须提供锁定 ID。如果未在 24 小时内提供锁定 ID，则锁定 ID 将过期，并且处于正在进行状态的策略将被删除。

12. 将您的锁定 ID 保存在安全的地方后，选择关闭。
13. 在接下来的 24 小时内测试您的文件库锁定策略。如果策略按预期运行，请选择完成文件库锁定策略。
14. 在完成文件库锁定对话框中，选中该复选框，以确认完成文件库锁定策略过程是不可逆的。
15. 在文本框中输入提供的锁定 ID。
16. 选择完成文件库锁定。

在 Amazon S3 Glacier 中处理档案

存档是您存储在文件库中的任何对象（例如，照片、视频或文档）。它是 Amazon S3 Glacier（S3 Glacier）的基本存储单位。每个档案都有唯一的 ID 和可选的描述。您在上传档案时，S3 Glacier 返回包括档案 ID 的响应。此档案 ID 在存储档案的 AWS 区域中是唯一的。以下是示例档案 ID。

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

存档长度 IDs 为 138 字节。在上传档案时，您可以提供可选的描述。您可以使用档案 ID（而不是档案描述）来检索档案。

> **Important**
>
> S3 Glacier 提供了一个管理控制台。您可以使用该控制台创建或删除文件库。但是，与 S3 Glacier 的所有其他交互活动要求您使用 AWS Command Line Interface (CLI) 或编写代码。例如，要上传照片、视频和其他文档等数据，您必须直接使用 REST API AWS CLI 或使用 Amazon 使用或编写代码来发出请求 SDKs。有关将 S3 Glacier 与配合使用的更多信息 AWS CLI，请转到 S3 Glacier AWS CLI 参考。要安装 AWS CLI，请转至AWS Command Line Interface。

主题

- Amazon S3 Glacier 中的归档操作
- 维护客户端档案元数据
- 在 Amazon S3 Glacier 中上传档案
- 在 S3 Glacier 中下载档案
- 删除 Amazon S3 Glacier 中的档案

Amazon S3 Glacier 中的归档操作

S3 Glacier 支持以下基本归档操作：上传、下载和删除。下载档案是一种异步操作。

在 Amazon S3 Glacier 中上传档案

您可以在单一操作中上传档案，也可以分段上传它。用于分段上传档案的 API 调用称为分段上传。有关更多信息，请参阅 在 Amazon S3 Glacier 中上传档案。

> **Important**
>
> S3 Glacier 提供了一个管理控制台。您可以使用该控制台创建或删除文件库。但是，与 S3 Glacier 的所有其他交互都需要您使用 AWS Command Line Interface (CLI) 或编写代码。例如，要上传照片、视频和其他文档等数据，您必须直接使用 REST API AWS CLI 或使用 Amazon 使用或编写代码来发出请求 SDKs。有关将 S3 Glacier 与配合使用的更多信息 AWS CLI，请转到 S3 Glacier AWS CLI 参考。要安装 AWS CLI，请转至AWS Command Line Interface。

在 Amazon S3 Glacier 中查找档案 ID

您可以通过下载包含下载档案的文件库的文件库清单来获取档案 ID。有关下载文件库清单的更多信息，请参阅在 Amazon S3 Glacier 中下载文件库清单。

在 Amazon S3 Glacier 中下载档案

下载档案是一种异步操作。您必须先启动下载特定档案的任务。收到任务请求后，S3 Glacier 会准备要供下载的档案。任务完成后，您可以下载您的档案数据。由于任务具有异步性，因此，您可以请求 S3 Glacier 在任务完成时向 Amazon Simple Notification Service (Amazon SNS) 主题发送通知。您可以为每个任务请求指定 SNS 主题，或者将您的文件库配置为在特定事件发生时发送通知。有关下载档案的更多信息，请参阅“在 S3 Glacier 中下载档案”。

删除 Amazon S3 Glacier 中的档案

S3 Glacier 提供了您可以用来一次删除一个档案的 API 调用。有关更多信息，请参阅 删除 Amazon S3 Glacier 中的档案。

更新 S3 Glacier 中的档案

上传档案后，您无法更新其内容或描述。可以更新档案内容或其描述的唯一方法是：删除档案，然后上传另一个档案。请注意，在您每次上传档案时，S3 Glacier 都会向您返回一个唯一的档案 ID。

维护客户端档案元数据

除了可选的档案描述以外，S3 Glacier 不支持档案的任何附加元数据。在您上传档案时，S3 Glacier 会分配一个 ID（字符的不透明序列），您无法从其推断有关档案的任何含意。您可能会在客户端维护有关档案的元数据。这些元数据可以包括档案名称以及有关档案的其他一些有意义的信息。

> **Note**
>
> 如果您是 Amazon Simple Storage Service (Amazon S3) 客户，则会知道，将数据元上传到存储段时，可以为数据元分配一个数据元密钥，例如 `MyDocument.txt` 或 `SomePhoto.jpg`。在 S3 Glacier 中，您无法为您上传的档案分配对象键。

如果您维护客户端档案元数据，请注意，S3 Glacier 会维护一份文件库清单，其中包括档案 IDs 以及您在档案上传期间提供的任何描述。有时候，您可能会下载文件库库存，以协调您为档案元数据维护的客户端数据库中存在的任何问题。但是，S3 Glacier 大约每天都会获取文件库清单。在您请求文件库清单时，S3 Glacier 会返回它准备的上一份清单（时间点快照）。

在 Amazon S3 Glacier 中上传档案

Amazon S3 Glacier（S3 Glacier）提供了一个管理控制台，您可以使用它来创建和删除文件库。但是，您无法通过使用管理控制台向 S3 Glacier 上传档案。要上传数据（例如照片、视频和其他文档），您必须使用 AWS CLI 或编写代码来提出请求，方法是直接使用 REST API 或使用 Amazon SDKs。

有关将 S3 Glacier 与配合使用的信息 AWS CLI，请转到 S3 Glacier AWS CLI 参考。要安装 AWS CLI，请转至AWS Command Line Interface。以下上传主题说明了如何使用适用于 Java 的 Amazon SDK、适用于 .NET 的 Amazon SDK 和 REST API 将档案上传到 S3 Glacier。

主题

- 向 Amazon S3 Glacier 上传档案的选项
- 在单一操作中上传档案
- 分段上传大型档案（分段上传）

向 Amazon S3 Glacier 上传档案的选项

根据上传的数据大小，S3 Glacier 提供以下选项：

- 在单一操作中上传档案 – 在单一操作中，您可以上传大小从 1 字节到最大 4 GB 的档案。但是，我们鼓励 S3 Glacier 客户使用分段上传来上传大于 100 MB 的档案。有关更多信息，请参阅 在单一操作中上传档案。
- 分段上传档案 – 利用分段上传 API，您可以上传大型档案，最大约为 40,000 GB (10,000 * 4 GB)。

  分段上传 API 调用旨在改进较大档案的上传体验。您可以分段上传档案。您可以独立地、以任何顺序以及并行地上传这些段。如果某段上传失败，则您只需重新上传该段，而无需重新上传整个档案。您可以对大小从 1 字节到大约 40 000 GB 的档案使用分段上传。有关更多信息，请参阅 分段上传大型档案（分段上传）。

> **Important**
>
> S3 Glacier 文件库清单一天仅更新一次。在上传档案时，您不会立即在控制台或您下载的文件库清单列表中看到添加到文件库的新档案，直至文件库清单完成更新。

使用 AWS Snowball Edge 服务

AWS Snowball Edge 绕过互联网，加速将大量数据移入和移出亚马逊自有设备。 AWS 有关更多信息，请参阅 AWS Snowball Edge 详细信息页面。

要将现有的数据上传到 Amazon S3 Glacier (S3 Glacier)，您可能会考虑使用 AWS Snowball Edge 设备类型之一将数据导入到 Amazon S3，然后使用生命周期规则将其移动到 S3 Glacier 存储类进行存档。当您将 Amazon S3 对象转换为 S3 Glacier 存储类时，Amazon S3 会在内部使用 S3 Glacier 以更低的成本实现持久存储。尽管对象存储在 S3 Glacier 中，它们仍是 Amazon S3 对象，需在 Amazon S3 中管理；并且您无法直接通过 S3 Glacier 访问它们。

有关 Amazon S3 生命周期配置以及将对象转换为 S3 Glacier 存储类的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的对象生命周期管理和转换对象。

在单一操作中上传档案

如在 Amazon S3 Glacier 中上传档案中所述，您可以在单一操作中上传较小的档案。但是，我们鼓励 Amazon S3 Glacier（S3 Glacier）客户使用分段上传来上传大于 100MB 的档案。

主题

- 使用单个操作上传档案 AWS Command Line Interface

- 使用单个操作上传档案 适用于 Java 的 AWS SDK
- 使用 Amazon S3 Glacier 适用于 .NET 的 AWS SDK 中的单一操作上传档案
- 使用 REST API 在单一操作中上传档案

使用单个操作上传档案 AWS Command Line Interface

您可以使用 () 在 Amazon S3 Glacier（S3 Glacier AWS CLI）中 AWS Command Line Interface 上传档案。

主题

- （先决条件）设置 AWS CLI
- 示例：使用上传档案 AWS CLI

（先决条件）设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   正在安装 AWS Command Line Interface

   正在配置 AWS Command Line Interface

2. 在命令提示符下输入以下命令来验证您的 AWS CLI 设置。这些命令没有显式提供凭证，因此将使用默认配置文件的凭证。

   - 尝试使用 help 命令。

     ```
     aws help
     ```

   - 要获取已配置账户上 S3 Glacier 文件库的列表，请使用 `list-vaults` 命令。*123456789012*用您的 AWS 账户 身份证替换。

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

   - 要查看的当前配置数据 AWS CLI，请使用`aws configure list`命令。

     ```
     aws configure list
     ```

示例：使用上传档案 AWS CLI

要上传档案，必须创建文件库。有关如何创建文件库的更多信息，请参阅在 Amazon S3 Glacier 中创建文件库。

1. 使用 `upload-archive` 命令将档案添加到现有文件库。在下面的示例中，替换 `vault name` 和 `account ID`。在 `body` 参数中指定要上传的文件的路径。

```
aws glacier upload-archive --vault-name awsexamplevault --account-id 123456789012
 --body archive.zip
```

2. 预期输出：

```
{
    "archiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
    "checksum": "969fb39823836d81f0cc028195fcdbcbbe76cdde932d4646fa7de5f21e18aa67",
    "location": "/123456789012/vaults/awsexamplevault/archives/
kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw"
}
```

完成后，该命令将输出档案 ID、校验和以及在 S3 Glacier 中的位置。有关 upload-archive 命令的更多信息，请参阅《AWS CLI 命令参考》中的 upload-archive。

使用单个操作上传档案 适用于 Java 的 AWS SDK

适用于 Java 的 Amazon SDK APIs 提供的高级版本和低级版本都提供了一种上传档案的方法。

主题

- 使用的高级别 API 上传档案 适用于 Java 的 AWS SDK
- 使用的低级 API 在单个操作中上传档案 适用于 Java 的 AWS SDK

使用的高级别 API 上传档案 适用于 Java 的 AWS SDK

该高级 API 的 `ArchiveTransferManager` 类提供了您可以用来将档案上传到文件库的 `upload` 方法。

> **Note**
>
> 您可以使用 upload 方法上传小型或大型档案。根据您要上传的档案大小，此方法会确定是在单一操作中上传档案，还是使用分段上传 API 分段上传档案。

示例：使用的高级别 API 上传档案 适用于 Java 的 AWS SDK

以下 Java 代码示例将档案上传到美国西部（俄勒冈州）区域 (us-west-2) 中的文件库 (examplevault)。有关支持的 AWS 区域和终端节点的列表，请参阅访问 Amazon S3 Glacier。

有关如何运行此示例的 step-by-step说明，请参阅使用 Eclipse 运行 Amazon S3 Glacier 的 Java 示例。您需要更新目标上传文件库名称和待上传文件名称旁显示的代码。

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);
        }
    }
}
```

使用的低级 API 在单个操作中上传档案 适用于 Java 的 AWS SDK

该低级 API 为所有归档操作提供了方法。下面是使用 适用于 Java 的 AWS SDK上传存档的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定要上传档案的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。
2. 通过创建一个 `UploadArchiveRequest` 类的实例提供请求信息。

   除了您要上传的数据以外，您还需要提供有效负载的校验和（SHA-256 树形哈希）、文件库名称、数据的内容长度和您的账户 ID。

   如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。
3. 以参数形式提供请求对象，运行 `uploadArchive` 方法。

   作为响应，Amazon S3 Glacier（S3 Glacier）返回新上传的档案的档案 ID。

以下 Java 代码段说明了前面的步骤。

```
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());
```

示例：使用的低级 API 在单个操作中上传档案 适用于 Java 的 AWS SDK

以下 Java 代码示例使用将档案上传 适用于 Java 的 AWS SDK 到文件库 (`examplevault`)。有关如何运行此示例的 step-by-step说明，请参阅使用 Eclipse 运行 Amazon S3 Glacier 的 Java 示例。您需要更新目标上传文件库名称和待上传文件名称旁显示的代码。

```
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);
        }
    }
}
```

使用 Amazon S3 Glacier 适用于 .NET 的 AWS SDK 中的单一操作上传档案

适用于.NET 的 Amazon SDK APIs 提供的高级版本和低级版本都提供了一种在单个操作中上传档案的方法。

主题

- 使用的高级别 API 上传档案 适用于 .NET 的 AWS SDK
- 使用的低级 API 在单个操作中上传档案 适用于 .NET 的 AWS SDK

使用的高级别 API 上传档案 适用于 .NET 的 AWS SDK

该高级 API 的 `ArchiveTransferManager` 类提供了您可以用来将档案上传到文件库的 `Upload` 方法。

> **Note**
>
> 您可以使用 `Upload` 方法上传小型或大型文件。根据您要上传的文件大小，此方法会确定是在单一操作中上传文件，还是使用分段上传 API 分段上传文件。

示例：使用的高级别 API 上传档案 适用于 .NET 的 AWS SDK

以下 C# 代码示例将档案上传到美国西部（俄勒冈州）区域中的文件库 (`examplevault`)。

有关如何运行此示例的 step-by-step说明，请参阅运行代码示例。您需要更新待上传文件名称旁所显示的代码。

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();
    }
  }
}
```

使用的低级 API 在单个操作中上传档案 适用于 .NET 的 AWS SDK

该低级 API 为所有归档操作提供了方法。下面是使用 适用于 .NET 的 AWS SDK上传存档的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定要上传档案的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。
2. 通过创建一个 `UploadArchiveRequest` 类的实例提供请求信息。

   除了您要上传的数据以外，您还需要提供有效负载的校验和（SHA-256 树形哈希）、文件库名称和您的账户 ID。

   如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 .NET 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。
3. 以参数形式提供请求对象，运行 `UploadArchive` 方法。

   作为响应，S3 Glacier 返回新上传的档案的档案 ID。

示例：使用的低级 API 在单个操作中上传档案 适用于 .NET 的 AWS SDK

以下 C# 代码示例说明了前面的步骤。该示例使用将档案上传 适用于 .NET 的 AWS SDK 到文件库 (`examplevault`)。

> **Note**
>
> 有关用于在单个请求中上传档案的底层 REST API 的信息，请参阅上传档案（发布档案）。

有关如何运行此示例的 step-by-step说明，请参阅运行代码示例。您需要更新待上传文件名称旁所显示的代码。

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;
      }
    }
  }
}
```

使用 REST API 在单一操作中上传档案

您可以使用上传档案 API 调用在单个操作中上传档案。有关更多信息，请参阅 上传档案（发布档案）。

分段上传大型档案（分段上传）

主题

- 分段上传流程
- 快讯
- 使用 AWS CLI上传大型档案
- 使用适用于 Java 的 Amazon SDK 分段上传大型档案
- 使用上传大型档案 适用于 .NET 的 AWS SDK
- 使用 REST API 分段上传大型档案

分段上传流程

如在 Amazon S3 Glacier 中上传档案中所述，我们鼓励 Amazon S3 Glacier（S3 Glacier）客户使用分段上传来上传大于 100 兆字节 (MiB) 的档案。

1. 开始分段上传

   当您发送请求以启动分段上传时，S3 Glacier 返回分段上传 ID，它是您分段上传的唯一标识符。后续的任何分段上传操作均需要此 ID。S3 Glacier 完成作业后，此 ID 至少在 24 小时内都不会过期。

   在您启动分段上传的请求中，请指定段大小（以字节数为单位）。除了最后一段以外，您上传的每一段都必须为此大小。

   > **Note**
   >
   > 使用分段上传时，您不需要知道整个档案大小。这意味着，在开始上传档案时，您可以在不知道档案大小的情况下使用分段上传。您只需在启动分段上传时决定段大小即可。

   此外，在启动分段上传请求中，您还可以提供可选的档案描述。

2. 上传段

对于每个段上传请求，您必须包括您在步骤 1 中获取的分段上传 ID。此外，在请求中，您还必须指定标识段在最终档案中的位置的内容范围（以字节为单位）。S3 Glacier 稍后会使用内容范围信息来以适当顺序拼组档案。由于您提供了上传的每一段的内容范围，因此，它会确定段在最终档案汇编中的位置，进而，您可以任何顺序上传段。此外，您还可以并行上传段。如果您使用与之前上传的段相同的内容范围上传新段，则之前上传的段会被覆盖。

3. 完成（或停止）分段上传

   上传所有档案段后，您可以使用完成操作。此外，您还必须在请求中指定上传 ID。S3 Glacier 将按您提供的内容范围以升序顺序拼组各个分段，从而创建档案。S3 Glacier 对“完成分段上传”请求的响应包括新创建的档案的档案 ID。如果您在“启动分段上传”请求中提供了可选的档案描述，则 S3 Glacier 会将它与拼组的档案相关联。成功完成分段上传后，您无法引用该分段上传 ID。这意味着，您无法访问与该分段上传 ID 相关联的段。

   如果停止分段上传，则您无法使用该分段上传 ID 上传其他任何段。与停止的分段上传相关联的任何段所占用的所有存储都会被释放。如果有任何分段上传正在进行，则即使在您停止后，它们仍然可能会成功或失败。

附加分段上传操作

Amazon S3 Glacier（S3 Glacier）提供了以下额外的分段上传 API 调用。

- 列出段—利用此操作，您可以列出特定分段上传的段。它会返回有关您为分段上传上传的段的信息。对于每个“列出段”请求，S3 Glacier 最多返回 1,000 段的信息。如果有更多段要为分段上传列出，则结果会分页，并且响应中会返回一个指示要从其所在位置继续列表的标记。您需要发送附加请求来检索后续的段。请注意，返回的段列表不包括未完成上传的段。
- 列出分段上传—利用此操作，您可以获取正在进行的分段上传的列表。正在进行的分段上传是已开始但还未完成或停止的上传。对于每个列出分段上传请求，S3 Glacier 最多会返回 1,000 个分段上传。如果有更多分段上传要列出，则结果会分页，并且响应中会返回一个指示要从其所在位置继续列表的标记。您需要发送附加请求来检索剩余的分段上传。

快讯

下表提供了分段上传的核心规范。

Item	规格
最大档案大小	10,000 x 4 吉字节 (GiB)
每次上传的分段的最大数量	10000
分段大小	1 MiB 到 4 GiB，最后一个分段可以小于 1 MiB。您可以指定大小值（以字节为单位）。 分段大小必须为兆字节 （1024 KiB）乘以 2 的幂，例如， 1048576 (1 MiB)、2097152 (2 MiB)、4194304 (4 MiB)、8388608 (8 MiB)。
列出分段请求返回的分段的最大数量	1000
在列出分段上传请求中返回的分段的最大数量	1000

使用 AWS CLI上传大型档案

您可以使用 () 在 Amazon S3 Glacier（S3 Glacier AWS CLI）中 AWS Command Line Interface 上传档案。为了改善大型档案的上传体验，S3 Glacier 提供了多个 API 操作来支持分段上传。通过这些 API 操作，您可以分段上传档案。您可以独立地、以任何顺序以及并行地上传这些段。如果某个分段上传失败，则您只需重新上传该分段，而无需重新上传整个档案。您可以对大小从 1 字节到大约 40,000 GiB 的档案使用分段上传。

有关 S3 Glacier 分段上传的更多信息，请参阅分段上传大型档案（分段上传）。

主题

- （先决条件）设置 AWS CLI
- （先决条件）安装 Python
- （先决条件）创建 S3 Glacier 文件库
- 示例：使用分段上传大型档案 AWS CLI

（先决条件）设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   正在安装 AWS Command Line Interface

   配置 AWS Command Line Interface

2. 在命令提示符下输入以下命令来验证您的 AWS CLI 设置。这些命令没有显式提供凭证，因此将使用默认配置文件的凭证。

   - 尝试使用 help 命令。

     ```
     aws help
     ```

   - 要获取已配置账户上 S3 Glacier 文件库的列表，请使用 `list-vaults` 命令。*123456789012*用您的 AWS 账户 身份证替换。

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

   - 要查看的当前配置数据 AWS CLI，请使用`aws configure list`命令。

     ```
     aws configure list
     ```

（先决条件）安装 Python

要完成分段上传，您必须计算所上传档案的 SHA256 树形哈希。这样做与计算要上传的文件的 SHA256 树形哈希值不同。要计算您正在上传的档案的 SHA256 树形哈希，可以使用 Java、C#（使用.NET），或 Python。 在本示例中，您将使用 Python。 有关使用说明 Java 或 C#，请参阅计算校验和。

有关安装的更多信息 Python，请参阅安装或更新 Python在 Boto3 开发者指南中。

（先决条件）创建 S3 Glacier 文件库

要使用下面的示例，必须至少创建一个 S3 Glacier 文件库。有关如何创建文件库的更多信息，请参阅在 Amazon S3 Glacier 中创建文件库。

示例：使用分段上传大型档案 AWS CLI

在此示例中，您将创建一个文件并使用分段上传 API 操作将此文件分段上传到 Amazon S3 Glacier。

> **Important**
>
> 在开始此过程之前，请确保您已执行所有先决条件的步骤。要上传档案，您必须创建并 AWS CLI 配置文件库，并做好使用准备 Java、C# 或 Python 计算 SHA256 树形哈希。

以下过程使用`initiate-multipart-uploadupload-multipart-part`、和`complete-multipart-upload` AWS CLI 命令。

有关每个命令的更多详细信息，请参见 initiate-multipart-upload, upload-multipart-part 和 complete-multipart-upload在 “AWS CLI 命令参考” 中。

1. 使用 initiate-multipart-upload命令来创建分段上传资源。在您的请求中，请指定分段大小（以字节数为单位）。除了最后一个分段以外，您上传的每个分段都必须为此大小。启动上传任务时，您不需要知道整个档案的大小。但是，在最后一步完成上传时，您将需要知道每个分段的总大小（以字节为单位）。

   在下面的命令中，将 `--vault-name` 和 `--account-ID` 参数的值替换为您自己的信息。此命令指定每个文件将上传分段大小为 1 兆字节 (MiB)（1024 x 1024 字节）的档案。如果需要，请替换此 `--part-size` 参数值。

   ```
   aws glacier initiate-multipart-upload --vault-name awsexamplevault --part-
   size 1048576 --account-id 123456789012
   ```

   预期输出：

   ```
   {
   "location": "/123456789012/vaults/awsexamplevault/multipart-uploads/uploadId",
   "uploadId": "uploadId"
   }
   ```

   完成后，该命令将输出分段上传资源的上传 ID 和在 S3 Glacier 中的位置。在后续步骤中，您将使用此上传 ID。

2. 在本示例中，您可以使用以下命令创建一个 4.4 MiB 的文件，将其拆分为 1 MiB 的块，然后上传每个块。要上传自己的文件，您可以按照类似的步骤将数据拆分为多个块并上传每个分段。

   Linux 或 macOS

以下命令将在 Linux 或 macOS 上创建一个名为 `file_to_upload` 的 4.4 MiB 的文件。

```
mkfile -n 9000b file_to_upload
```

Windows

以下命令将在 Windows 上创建一个名为 `file_to_upload` 的 4.4 MiB 的文件。

```
fsutil file createnew file_to_upload 4608000
```

3. 接下来，您将把此文件拆分为 1 MiB 的块。

```
split -b 1048576 file_to_upload chunk
```

您现在有以下五个块。前四个大小为 1 MiB，最后一个是 400 千字节 (KiB)。

```
chunkaa
chunkab
chunkac
chunkad
chunkae
```

4. 使用 upload-multipart-part命令上传档案的一部分。您可以按任何顺序上传档案分段。此外，您还可以并行上传分段。您最多可以为一个分段上传上传 10 000 段。

   在下面的命令中，将替换 `--vault-name`、`--account-ID` 和 `--upload-id` 参数的值。上传 ID 必须与 `initiate-multipart-upload` 命令输出中提供的 ID 相匹配。`--range` 参数指定您将上传大小为 1 MiB（1024 x 1024 字节）的分段。此大小必须与您在 `initiate-multipart-upload` 命令中指定的大小一致。如有必要，请调整此大小值。`--body` 参数指定您要上传的分段的名称。

```
aws glacier upload-multipart-part --body chunkaa --range='bytes 0-1048575/*' --
vault-name awsexamplevault --account-id 123456789012 --upload-id upload_ID
```

   如果成功，该命令将生成包含已上传分段校验和的输出。

5. 再次运行 `upload-multipart-part` 命令以上传分段上传的其余部分。更新每个命令的 `--range` 和 `--body` 参数值，使其与您要上传的分段相匹配。

```
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**
>
> 最终命令的 `--range` 参数值较小，因为上传的最后一个分段小于 1 MiB。如果成功，每个命令都将生成包含每个已上传分段校验和的输出。

6. 接下来，您将接组档案并完成上传。您必须包括档案的总大小和 SHA256 树形哈希。

   要计算档案的 SHA256 树形哈希，可以使用 Java、C# 或 Python。 在本示例中，您将使用 Python。 有关使用说明 Java 或 C#，请参阅计算校验和。

   创建 Python 文件`checksum.py`并插入以下代码。如果需要，请替换原始文件的名称。

```
from botocore.utils import calculate_tree_hash

checksum = calculate_tree_hash(open('file_to_upload', 'rb'))
print(checksum)
```

7. 运行`checksum.py`计算 SHA256 树形哈希。以下哈希值可能与您的输出不匹配。

```
$ python3 checksum.py
$ 3d760edb291bfc9d90d35809243de092aea4c47b308290ad12d084f69988ae0c
```

8. 使用 complete-multipart-upload命令完成档案上传。替换 `--vault-name`、`--account-ID`、`--upload-ID` 和 `--checksum` 参数的值。`--archive` 参数值以字节为单位指定档案的总大小。此值必须是为您上传的各段的所有大小之和。如果需要，请替换此值。

```
aws glacier complete-multipart-upload --archive-size 4608000 --vault-
name awsexamplevault --account-id 123456789012 --upload-id upload_ID --
checksum checksum
```

完成后，该命令将输出档案的 ID、校验和以及在 S3 Glacier 中的位置。

使用适用于 Java 的 Amazon SDK 分段上传大型档案

适用于 Java 的 Amazon SDK APIs 提供的高级和低级都提供了一种上传大型档案的方法（参见在 Amazon S3 Glacier 中上传档案）。

- 该高级 API 提供了您可以用来上传任何大小的档案的方法。根据您要上传的文件，该方法会在单一操作中上传档案或者使用 Amazon S3 Glacier（S3 Glacier）中的分段上传支持来分段上传档案。
- 该低级 API 紧密映射到底层 REST 实施。因此，它提供了一个在单一操作中上传较小档案的方法，以及一组支持较大档案的分段上传的方法。此部分说明了使用低级 API 分段上传大型档案的操作。

有关高级和低级别的更多信息 APIs，请参阅适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

主题

- 使用的高级别 API 分段上传大型档案 适用于 Java 的 AWS SDK
- 使用的低级 API 分段上传大型档案 适用于 Java 的 AWS SDK

使用的高级别 API 分段上传大型档案 适用于 Java 的 AWS SDK

您可以使用高级 API 的相同方法上传小型或大型档案。根据档案大小，高级 API 方法会决定是在单个操作中上传档案，还是使用 S3 Glacier 提供的分段上传 API 上传档案。有关更多信息，请参阅 使用的高级别 API 上传档案 适用于 Java 的 AWS SDK。

使用的低级 API 分段上传大型档案 适用于 Java 的 AWS SDK

对于上传的粒度控制，您可以使用低级 API（您可以在其中配置请求以及处理响应）。以下是使用 适用于 Java 的 AWS SDK分段上传大型档案的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

您需要指定要保存档案的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过调用 `initiateMultipartUpload` 方法启动分段上传。

   您需要提供要上传档案的文件库名称、要上传的档案段的大小，也可选择提供相关描述。您可以通过创建 `InitiateMultipartUploadRequest` 类的实例提供此信息。作为响应，S3 Glacier 会返回上传 ID。

3. 通过调用 `uploadMultipartPart` 方法上传段。

   对于要上传的每一段，您需要提供文件库名称、将在此段中上传的最终组合档案的字节范围、段数据的校验和，以及上传 ID。

4. 通过调用 `completeMultipartUpload` 方法完成分段上传。

   您需要提供上传 ID、整个档案的校验和、档案大小（您上传的所有段的组合大小）和文件库名称。S3 Glacier 从上传的分段构造档案并返回档案 ID。

示例：使用分段上传大型档案 适用于 Java 的 AWS SDK

以下 Java 代码示例使用将档案上传 适用于 Java 的 AWS SDK 到文件库 (`examplevault`)。有关如何运行此示例的 step-by-step说明，请参阅使用 Eclipse 运行 Amazon S3 Glacier 的 Java 示例。您需要更新待上传文件名称旁所显示的代码。

> **Note**
>
> 此示例对 1MB 到 1GB 的段大小有效。但是，S3 Glacier 支持最大为 4GB 的分段。

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();
    }
}
```

使用上传大型档案 适用于 .NET 的 AWS SDK

适用于.NET 的 Amazon SDK APIs 提供的高级和低级都提供了一种分段上传大型档案的方法（参见在 Amazon S3 Glacier 中上传档案）。

- 该高级 API 提供了您可以用来上传任何大小的档案的方法。根据您要上传的文件，该方法会在单一操作中上传档案或者使用 Amazon S3 Glacier（S3 Glacier）中的分段上传支持来分段上传档案。
- 该低级 API 紧密映射到底层 REST 实施。因此，它提供了一个在单一操作中上传较小档案的方法，以及一组支持较大档案的分段上传的方法。此部分说明了使用低级 API 分段上传大型档案的操作。

有关高级和低级别的更多信息 APIs，请参阅适用于 .NET 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

主题

- 使用的高级别 API 分段上传大型档案 适用于 .NET 的 AWS SDK
- 使用的低级 API 分段上传大型档案 适用于 .NET 的 AWS SDK

使用的高级别 API 分段上传大型档案 适用于 .NET 的 AWS SDK

您可以使用高级 API 的相同方法上传小型或大型档案。根据档案大小，高级 API 方法会决定是在单个操作中上传档案，还是使用 S3 Glacier 提供的分段上传 API 上传档案。有关更多信息，请参阅 使用的高级别 API 上传档案 适用于 .NET 的 AWS SDK。

使用的低级 API 分段上传大型档案 适用于 .NET 的 AWS SDK

对于上传的粒度控制，您可以使用低级 API（您可以在其中配置请求以及处理响应）。以下是使用 适用于 .NET 的 AWS SDK分段上传大型档案的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定要保存档案的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过调用 `InitiateMultipartUpload` 方法启动分段上传。

   您需要提供要上传档案的文件库名称、要上传的档案段的大小，也可选择提供相关描述。您可以通过创建 `InitiateMultipartUploadRequest` 类的实例提供此信息。作为响应，S3 Glacier 会返回上传 ID。

3. 通过调用 `UploadMultipartPart` 方法上传段。

   对于要上传的每一段，您需要提供文件库名称、将在此段中上传的最终组合档案的字节范围、段数据的校验和，以及上传 ID。

4. 通过调用 `CompleteMultipartUpload` 方法完成分段上传。

   您需要提供上传 ID、整个档案的校验和、档案大小（您上传的所有段的组合大小）和文件库名称。S3 Glacier 从上传的分段构造档案并返回档案 ID。

示例：使用适用于 .NET 的 Amazon SDK 分段上传大型档案

以下 C# 代码示例使用将档案上传 适用于 .NET 的 AWS SDK 到文件库 (`examplevault`)。有关如何运行此示例的 step-by-step说明，请参阅运行代码示例。您需要更新待上传文件名称旁显示的代码。

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;
    }
  }
}
```

使用 REST API 分段上传大型档案

如分段上传大型档案（分段上传）中所述，分段上传是指一组操作，这些操作可让您分段上传档案并执行相关操作。有关这些操作的更多信息，请参阅以下 API 参考主题：

- 启动分段上传（发布分段上传）
- 上传段（设置上传 ID）
- 完成分段上传（发布上传 ID）
- 中止分段上传（删除上传 ID）
- 列出段（获取上传 ID）
- 列出分段上传（获取分段上传）

在 S3 Glacier 中下载档案

Amazon S3 Glacier 提供了一个管理控制台，您可以使用它来创建和删除文件库。但是，您无法通过使用管理控制台从 S3 Glacier 下载档案。要下载数据（例如照片、视频和其他文档），必须使用 AWS Command Line Interface (AWS CLI) 或编写代码来发出请求，方法是直接使用 REST API 或使用 AWS SDKs。

有关将 S3 Glacier 与配合使用的信息 AWS CLI，请参阅 S3 Glacier AWS CLI 参考资料。要安装 AWS CLI，请参阅AWS Command Line Interface。以下主题介绍如何使用 适用于 Java 的 AWS SDK、和 Amazon S3 Glacier REST API 将 适用于 .NET 的 AWS SDK档案下载到 S3 Glacier。

主题

- 检索 S3 冰川档案
- 使用 Amazon S3 Glacier 下载档案 适用于 Java 的 AWS SDK
- 使用 适用于 .NET 的 AWS SDK在 Amazon S3 Glacier 中下载档案
- 使用 Python 并行处理下载大型档案
- 使用 REST API 下载档案
- 使用 Amazon S3 Glacier 下载档案 AWS CLI

检索 S3 冰川档案

从 Amazon S3 Glacier 检索档案是一个异步操作，您首先需要启动任务，然后在任务完成后下载输出。要启动档案取回任务，您可以使用 启动作业（POST 作业） REST API 操作或中的等效操作 AWS CLI、或 AWS SDKs。

主题

- 档案检索选项
- 关于限范围的档案检索

从 S3 Glacier 检索档案是一个分为两个步骤的过程。以下是此过程的概述。

检索档案

1. 启动档案检索任务。

a. 获得您要检索的档案的 ID。您可以从文件库清单获取档案 ID。您可以通过 REST API 获取档案 ID AWS CLI、或 AWS SDKs。有关更多信息，请参阅 在 Amazon S3 Glacier 中下载文件库清单。

b. 使用启动作业（POST 作业）操作启动任务，请求 S3 Glacier 为后续下载准备整个档案或档案的一部分。

当您启动任务时，S3 Glacier 会在响应中返回任务 ID 并异步运行任务。（如步骤 2 所述，在任务完成之前，您不能下载任务输出。）

> **Important**
>
> 数据检索策略可能导致您的 `Initiate Job` 请求失败，并发生 `PolicyEnforcedException` 异常，但这仅限于标准检索。有关数据检索策略的更多信息，请参阅S3 Glacier 数据检索策略。有关 `PolicyEnforcedException` 异常的更多信息，请参阅错误响应。

如果需要，您可以还原存储在 S3 Glacier 中的大型数据段。有关从 S3 Glacier 存储类恢复数据的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的用于归档对象的存储类。

2. 在任务完成后，使用获取任务输出（获取输出）操作下载字节。

   您可以下载所有字节，或者指定字节范围，只下载任务输出的一部分。对于较大的输出，以区块下载输出的方式在下载失败（例如，由于网络发生故障而失败）时对您有所帮助。如果您在单一请求中获取任务输出，并且网络发生故障，则您不得不从头重新开始下载输出。但是，如果您以区块下载输出，万一发生任何故障，则您只需重新开始下载较小的部分，而不是整个输出。

S3 Glacier 必须先完成任务，然后，您才能获取其输出。任务在完成后的至少 24 小时内都不会过期，这意味着，您可以在任务完成后的 24 小时期限内下载输出。恢复可以在任务完成后 24 小时后随时过期。要确定您的任务是否已完成，请使用以下选项之一检查其状态：

- 等待任务完成通知 –您可以指定 S3 Glacier 在完成任务后可以向其发布通知的 Amazon Simple Notification Service (Amazon SNS) 主题。S3 Glacier 只有在完成任务后才会发送通知。

  启动任务时，您可以为该任务指定 Amazon SNS 主题。除了在您的任务请求中指定 Amazon SNS 主题以外，如果您的文件库已为档案检索事件设置了通知配置，S3 Glacier 也会向该 SNS 主题发布通知。有关更多信息，请参阅 在 Amazon S3 Glacier 中配置文件库通知。

- 显式请求任务信息 –您也可以使用 S3 Glacier `Describe Job` API operation (描述任务（获取任务 ID）)，以定期轮询任务信息。但是，建议使用 Amazon SNS 通知。

> **Note**
>
> 使用 Amazon SNS 通知获取的信息与调用 `Describe Job` API 操作所获取的信息相同。

档案检索选项

在启动检索档案的任务时，您可以根据访问时间和成本需求指定以下检索选项之一。有关检索定价的信息，请参阅 Amazon S3 Glacier 定价。

- 加速 - 加速检索允许您在偶尔需要紧急请求还原档案时快速访问存储在 S3 Glacier Flexible Retrieval 存储类或 S3 Intelligent-Tiering 归档访问层中的数据。对于除了最大型档案（250 MB 以上）之外的所有其他档案，使用加速检索访问的数据通常在 1 到 5 分钟内可用。预配置容量确保在您需要时，可以使用针对加速检索的检索容量。有关更多信息，请参阅 预配置容量。
- 标准 –标准检索允许您在数小时内访问您的任意档案。标准检索通常在 3 到 5 小时内完成。“标准”是未指定检索选项的检索请求的原定设置选项。
- 批量 – 批量检索是 S3 Glacier 最低成本的检索选项，使您可以在一天内以较低的成本检索大量（甚至是 PB 级）的数据。批量检索通常在 5 到 12 小时内完成。

下表总结了归档检索选项。有关定价的信息，请参阅 Amazon S3 Glacier 定价。

要进行`ExpeditedStandard`、或`Bulk`检索，`Tier`请将 RestoreObjectREST API 操作请求中的请求元素设置为所需的选项，或 AWS Command Line Interface (AWS CLI) 或中的等效选项 AWS SDKs。如果您购买了预配置容量，则所有加速检索都会通过您的预配置容量自动获得处理。

预配置容量

预配置容量帮助确保在您需要时，可以使用针对加速检索的检索容量。每个容量单位允许每 5 分钟至少执行三次加急检索，并提供高达每秒 150 兆字节的检索吞吐量 () MBps。

如果您的工作负载需要极高的稳定性和对数据子集可预测的访问性能（以分钟为单位），建议您购买预调配检索容量。没有预配置容量的加速检索通常也可以接受，但是在极少情况下会出现不寻常的高需求。不过，如果您需要随时可以访问加速检索，您必须购买预配置检索容量。

购买预配置容量

您可以使用 S3 Glacier 控制台、购买预配置容量 (POST provisioned-capacity) REST API 操作或购买预配置的 AWS SDKs容量单位。 AWS CLI有关预配置容量的定价信息，请参阅 Amazon S3 Glacier 定价。

预配置容量单位将持续一个月，从购买日期和时间开始计算。

如果开始日期为一个月的第 31 天，过期日期为下个月的最后一天。例如，如果开始日期为 8 月 31 日，则过期日期为 9 月 30 日。如果开始日期为 1 月 31 日，则过期日期为 2 月 28 日。

使用 Amazon S3 Glacier 控制台购买预配置容量

1. 登录 AWS Management Console 并在https://console.aws.amazon.com/glacier/家中打开 S3 Glacier 控制台。
2. 在左侧的导航窗格中，选择数据检索设置。
3. 在预配置容量单位 (PCUs) 下，选择购买 PC U。此时将显示购买 PCU 对话框。
4. 如果要购买预配置容量，请在确认购买框中输入 **`confirm`**。
5. 选择购买 PCU。

关于限范围的档案检索

当您从 S3 Glacier 检索档案时，您可以选择性地指定要检索的档案范围（部分）。默认为检索整个档案。如果您要执行以下操作，指定字节范围会很有用：

- 管理您的数据下载 – S3 Glacier 允许您在检索请求完成后的 24 小时内下载检索的数据。因此，您可能只想要检索档案的某些部分，以便在给定的下载时间窗内管理下载时间表。
- 检索大型档案的目标段 – 例如，假设您之前聚合了许多文件并以单一档案的形式上传了这些文件，您现在想检索这些文件中的一些文件。在这种情况下，您可以通过使用一个检索请求指定档案的范围，该范围包含您感兴趣的文件。或者，您可以启动多个检索请求，每个请求均具有一个针对一个或多个文件的范围。

当使用范围检索启动检索任务时，您必须提供以兆字节对齐的范围。也就是说，字节范围可以从零（档案的开头）开始，或者从其后的任何 1-MB 间隔（1MB、2MB、3MB，依此类推）处开始。

该范围的结尾可以是您档案的结尾或大于范围开头的任何 1MB 间隔处。此外，如果您要在（检索任务完成后）下载数据时获取校验和值，则您在任务启动中请求的范围还必须以树形哈希对齐。可以使用校

验和来确保数据在传输过程中没有损坏。有关兆字节对齐和树形哈希对齐的更多信息，请参阅“下载数据时接收校验和”。

使用 Amazon S3 Glacier 下载档案 适用于 Java 的 AWS SDK

适用于 Java 的 Amazon SDK APIs 提供的高级版本和低级版本都提供了一种下载档案的方法。

主题

- 使用的高级别 API 下载档案 适用于 Java 的 AWS SDK
- 使用的低级 API 下载档案 适用于 Java 的 AWS SDK

使用的高级别 API 下载档案 适用于 Java 的 AWS SDK

该高级 API 的 `ArchiveTransferManager` 类提供了您可以用来下载档案的 `download` 方法。

> **Important**
>
> `ArchiveTransferManager` 类将创建 Amazon Simple Notification Service (Amazon SNS) 主题，以及该主题订阅的 Amazon Simple Queue Service (Amazon SQS) 队列。然后，它启动了档案检索任务，并对队列进行轮询以便找到可用档案。如果存档可用，则开始下载。有关检索时间的详细信息，请参阅 档案检索选项。

示例：使用高级 API 下载档案 适用于 Java 的 AWS SDK

以下 Java 代码示例将从美国西部（俄勒冈州）区域 (`us-west-2`) 中的文件库 (`examplevault`) 中下载档案。

有关运行此示例的 step-by-step说明，请参阅使用 Eclipse 运行 Amazon S3 Glacier 的 Java 示例。您需要更新现有档案 ID 和已下载档案的本地文件保存路径旁显示的代码

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);
        }
    }
}
```

使用的低级 API 下载档案 适用于 Java 的 AWS SDK

以下是使用 适用于 Java 的 AWS SDK 低级 API 检索文件库清单的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定要从中下载档案的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过执行 `archive-retrieval` 方法启动 `initiateJob` 任务。

   您可以通过创建一个 `InitiateJobRequest` 类的实例提供任务信息，例如，您要下载的档案的档案 ID，以及您希望 Amazon S3 Glacier（S3 Glacier）向其发布任务完成消息的可选 Amazon SNS 主题。作为响应，S3 Glacier 返回任务 ID。该响应位于一个 `InitiateJobResult` 类的实例中。

```
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();
```

您可以选择指定字节范围，以请求 S3 Glacier 只准备档案的一部分。例如，您可以通过添加以下语句更新前面的请求，以请求 S3 Glacier 只准备档案的 1MB 到 2MB 部分。

```
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. 等待作业完成。

   您必须等到任务输出已作好供您下载的准备。如果您在文件库中设置了标识 Amazon Simple Notification Service (Amazon SNS) 主题的通知配置，或者在启动任务时指定了Amazon SNS 主题，则 S3 Glacier 会在完成任务后向该主题发送消息。

   此外，您还可以通过调用 `describeJob` 方法轮询 S3 Glacier 来确定任务完成状态。尽管如此，使用 Amazon SNS 主题进行通知才是推荐的方法。

4. 通过执行 `getJobOutput` 方法下载任务输出（档案数据）。

   您可以通过创建一个 `GetJobOutputRequest` 类的实例来提供请求信息，例如，任务 ID 和文件库名称。S3 Glacier 返回的输出位于 `GetJobOutputResult` 对象中。

```
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.
```

   前面的代码段会下载整个任务输出。您可以选择性地只检索输出的一部分，或者通过在您的 `GetJobOutputRequest` 中指定字节范围以较小的区块下载整个输出。

```
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 ****");
```

   在响应您的 `GetJobOutput` 调用时，S3 Glacier 返回您下载的数据部分的校验和（如果满足特定条件）。有关更多信息，请参阅 下载数据时接收校验和。

   要确认下载中没有错误，您随后可以在客户端计算校验和，并将它与 S3 Glacier 在响应中发送的校验和相比较。

   对于指定了可选范围的档案取回任务，当你获得任务描述时，它会包括你正在检索的范围的校验和 () SHA256 TreeHash。您可以使用此值进一步确认您稍后下载的整个字节范围的准确性。例如，如果您启动任务以检索以树形哈希对齐的档案范围，然后以区块下载输出，使得您的每个

`GetJobOutput` 请求均返回一个校验和，则您可以在客户端计算您下载的每个部分的校验和，然后计算树形哈希。您可以将它与 S3 Glacier 为响应您的描述任务请求而返回的校验和相比较，以确认您下载的整个字节范围与 S3 Glacier 中存储的字节范围相同。

要了解可工作的示例，请参阅 示例 2：使用分块 适用于 Java 的 AWS SDK下载输出的低级 API 检索档案 。

示例 1：使用低级 API 检索档案 适用于 Java 的 AWS SDK

以下 Java 代码示例会从指定的文件库下载档案。任务完成后，该示例会在单一 `getJobOutput` 调用中下载整个输出。有关以区块下载输出的示例，请参阅“示例 2：使用分块 适用于 Java 的 AWS SDK下载输出的低级 API 检索档案 ”。

该示例执行以下任务：

- 创建 Amazon Simple Notification Service (Amazon SNS) 主题。

  完成任务后，S3 Glacier 会向此主题发送通知。
- 创建 Amazon Simple Queue Service (Amazon SQS) 队列。

  该示例会向该队列附加策略，以使 Amazon SNS 主题能够向该队列发布消息。
- 启动任务以下载指定的档案。

  在任务请求中，指定了创建的 Amazon SNS 主题，以便 S3 Glacier 可以在完成任务后向该主题发布通知。
- 定期检查 Amazon SQS 队列是否有包含该任务 ID 的消息。

  如果有消息，则分析 JSON，并检查任务是否已成功完成。如果已成功完成，则下载档案。
- 通过删除它创建的 Amazon SNS 主题和 Amazon SQS 队列清除相关数据。

```
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));
    }
}
```

示例 2：使用分块 适用于 Java 的 AWS SDK下载输出的低级 API 检索档案

以下 Java 代码示例从 S3 Glacier 检索档案。该代码示例会通过在 `GetJobOutputRequest` 数据元中指定字节范围来以区块下载任务输出。

```
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));
    }
}
```

使用 适用于 .NET 的 AWS SDK在 Amazon S3 Glacier 中下载档案

适用于.NET 的 Amazon SDK APIs 提供的高级版本和低级版本都提供了一种下载档案的方法。

主题

- 使用的高级别 API 下载档案 适用于 .NET 的 AWS SDK
- 使用的低级 API 下载档案 适用于 .NET 的 AWS SDK

使用的高级别 API 下载档案 适用于 .NET 的 AWS SDK

该高级 API 的 `ArchiveTransferManager` 类提供了您可以用来下载档案的 `Download` 方法。

> **Important**
>
> `ArchiveTransferManager` 类将创建 Amazon Simple Notification Service (Amazon SNS) 主题，以及该主题订阅的 Amazon Simple Queue Service (Amazon SQS) 队列。然后，它启动了档案检索任务，并对队列进行轮询以便找到可用档案。如果存档可用，则开始下载。有关检索时间的详细信息，请参阅 档案检索选项

示例：使用的高级别 API 下载档案 适用于 .NET 的 AWS SDK

以下 C# 代码示例将从美国西部（俄勒冈州）区域中的文件库 (`examplevault`) 中下载档案。

有关如何运行此示例的 step-by-step说明，请参阅运行代码示例。您需要更新现有档案 ID 和已下载档案的本地文件保存路径旁显示的代码

```
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);
      }
    }
  }
}
```

使用的低级 API 下载档案 适用于 .NET 的 AWS SDK

以下是使用 适用于 .NET 的 AWS SDK的低级 API 下载 Amazon S3 Glacier（S3 Glacier）档案的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定要从中下载档案的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过执行 `archive-retrieval` 方法启动 `InitiateJob` 任务。

   您可以通过创建一个 `InitiateJobRequest` 类的实例提供任务信息，例如，您要下载的档案的档案 ID，以及您希望 S3 Glacier 向其发布任务完成消息的可选 Amazon SNS 主题。作为响应，S3 Glacier 返回任务 ID。该响应位于一个 `InitiateJobResponse` 类的实例中。

```
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;
```

   您可以选择指定字节范围，以请求 S3 Glacier 只准备档案的一部分，如以下请求所示。该请求指定 S3 Glacier 只准备档案的 1 MB 到 2 MB 部分。

```
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. 等待作业完成。

   您必须等到任务输出已作好供您下载的准备。如果您在文件库中设置了标识 Amazon Simple Notification Service (Amazon SNS) 主题的通知配置，或者在启动任务时指定了Amazon SNS 主题，则 S3 Glacier 会在完成任务后向该主题发送消息。以下部分给出的代码示例使用适用于 S3 Glacier 的 Amazon SNS 来发布消息。

   此外，您还可以通过调用 `DescribeJob` 方法轮询 S3 Glacier 来确定任务完成状态。尽管如此，使用 Amazon SNS 主题进行通知才是推荐的方法。

4. 通过执行 `GetJobOutput` 方法下载任务输出（档案数据）。

   您可以通过创建一个 `GetJobOutputRequest` 类的实例来提供请求信息，例如，任务 ID 和文件库名称。S3 Glacier 返回的输出位于 `GetJobOutputResponse` 对象中。

```
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);
  }
```

```
}
```

前面的代码段会下载整个任务输出。您可以选择性地只检索输出的一部分，或者通过在您的 `GetJobOutputRequest` 中指定字节范围以较小的区块下载整个输出。

```
GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
{
  JobId = jobId,
  VaultName = vaultName
};
getJobOutputRequest.SetRange(0, 1048575); // Download only the first 1 MB chunk of
 the output.
```

在响应您的 `GetJobOutput` 调用时，S3 Glacier 返回您下载的数据部分的校验和（如果满足特定条件）。有关更多信息，请参阅 下载数据时接收校验和。

要确认下载中没有错误，您随后可以在客户端计算校验和，并将它与 S3 Glacier 在响应中发送的校验和相比较。

对于指定了可选范围的档案取回作业，当你获得任务描述时，它会包括你要检索的范围的校验和 (SHA256TreeHash)。你可以使用这个值来进一步验证你以后下载的整个字节范围的准确性。例如，如果您启动任务以检索以树形哈希对齐的档案范围，然后以区块下载输出，使得您的每个 `GetJobOutput` 请求均返回一个校验和，则您可以在客户端计算您下载的每个部分的校验和，然后计算树形哈希。您可以将它与 S3 Glacier 为响应您的描述任务请求而返回的校验和相比较，以确认您下载的整个字节范围与 S3 Glacier 中存储的字节范围相同。

要了解可工作的示例，请参阅 示例 2：使用分块 适用于 .NET 的 AWS SDK下载输出的低级 API 检索档案。

示例 1：使用低级 API 检索档案 适用于 .NET 的 AWS SDK

以下 C# 代码示例会从指定的文件库下载档案。任务完成后，该示例会在单一 `GetJobOutput` 调用中下载整个输出。有关以区块下载输出的示例，请参阅“示例 2：使用分块 适用于 .NET 的 AWS SDK下载输出的低级 API 检索档案”。

该示例执行以下任务：

- 设置 Amazon Simple Notification Service (Amazon SNS) 主题

完成任务后，S3 Glacier 会向此主题发送通知。

- 设置 Amazon Simple Queue Service (Amazon SQS) 队列

  该示例会向该队列附加策略，以使 Amazon SNS 主题能够发布消息。

- 启动任务以下载指定的档案。

  在任务请求中，该示例会指定 Amazon SNS 主题，以便 S3 Glacier 可以在完成任务后发送消息。

- 定期检查 Amazon SQS 队列是否有消息。

  如果有消息，则分析 JSON，并检查任务是否已成功完成。如果已成功完成，则下载档案。该代码示例使用 JSON.NET 库（请参阅 JSON.NET）来分析 JSON。

- 通过删除它创建的 Amazon SNS 主题和 Amazon SQS 队列清除相关数据。

```
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);
      }
    }
  }
}
```

示例 2：使用分块 适用于 .NET 的 AWS SDK下载输出的低级 API 检索档案

以下 C# 代码示例从 S3 Glacier 检索档案。该代码示例会通过在 `GetJobOutputRequest` 数据元中指定字节范围来以区块下载任务输出。

```
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);
      }
    }
  }
}
```

使用 Python 并行处理下载大型档案

本主题介绍如何使用 Python 的并行处理从 Amazon S3 Glacier (S3 Glacier) 下载大型档案。这种方法允许您通过将任何大小的档案分解成可以独立处理的较小片段来可靠地下载它们。

概览

此示例中提供的 Python 脚本执行以下任务：

1. 为通知设置必要的 AWS 资源（亚马逊 SNS 主题和亚马逊 SQS 队列）

2. 使用 S3 Glacier 启动档案检索任务
3. 监控 Amazon SQS 队列中是否有任务完成通知
4. 将大型存档拆分为可管理的块
5. 使用多个工作线程并行下载区块
6. 将每个区块保存到磁盘以备日后重新组装

先决条件

在开始之前，请确保你有：

- 已安装 Python 3.6 或更高版本
- AWS 已安装适用于 Python 的 SDK (Boto3)
- AWS 凭证配置了适用于 S3 Glacier、Amazon SNS 和 Amazon SQS 的相应权限
- 足够的磁盘空间来存储下载的存档块

示例：使用 Python 并行处理下载档案

以下 Python 脚本演示了如何使用并行处理从 S3 Glacier 下载大型档案：

```
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)
```

使用脚本

要使用此脚本，请执行以下步骤：

1. 用您的特定信息替换脚本中的占位符值：
   - *output_file_path*: 用于保存区块文件的目录
   - *vault_name*: 你的 S3 Glacier 保管库的名称
   - *notify_queue_name*: 作业通知队列的名称
   - *chunk_download_queue_name*: 区块下载队列的名称

- *sns_topic_name*: SNS 主题的名称
- *region*: AWS 您的保管库所在的地区
- *archive_id*: 要检索的档案的 ID

2. 运行 脚本：

```
python download_large_archive.py
```

3. 下载完所有区块后，你可以使用如下命令将它们合并为一个文件：

```
cat /path/to/chunks/*.chunk > complete_archive.file
```

重要注意事项

使用此脚本时，请记住以下几点：

- 从 S3 Glacier 检索档案可能需要几个小时才能完成，具体取决于所选的检索层。
- 该脚本无限期运行，持续轮询队列。您可能需要根据自己的具体要求添加终止条件。
- 确保您有足够的磁盘空间来存储存档的所有区块。
- 如果脚本中断，您可以重新启动脚本`retrieve_archive=False`以继续下载区块，而无需启动新的检索任务。
- 根据您的网络带宽*chunk_size*和系统资源调整和*workers*参数。
- 亚马逊 S3 检索、亚马逊 SNS 和亚马逊 SQS 的使用收取标准 AWS 费用。

使用 REST API 下载档案

使用 REST API 下载档案

下载档案是一个分为两个步骤的流程。

1. 启动 `archive-retrieval` 类型的任务。有关更多信息，请参阅 启动作业（POST 作业）。
2. 任务完成后，下载档案数据。有关更多信息，请参阅 获取任务输出（获取输出）。

使用 Amazon S3 Glacier 下载档案 AWS CLI

您可以使用 () 在 Amazon S3 Glacier（S3 Glacier AWS CLI）中 AWS Command Line Interface 下载档案。

主题

- （先决条件）设置 AWS CLI
- 示例：使用下载档案 AWS CLI

（先决条件）设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   正在安装 AWS Command Line Interface

   正在配置 AWS Command Line Interface

2. 在命令提示符下输入以下命令来验证您的 AWS CLI 设置。这些命令没有显式提供凭证，因此将使用默认配置文件的凭证。

   - 尝试使用 help 命令。

     ```
     aws help
     ```

   - 要获取已配置账户上 S3 Glacier 文件库的列表，请使用 `list-vaults` 命令。*123456789012*用您的 AWS 账户 身份证替换。

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

   - 要查看的当前配置数据 AWS CLI，请使用`aws configure list`命令。

     ```
     aws configure list
     ```

示例：使用下载档案 AWS CLI

> **Note**
>
> 要下载档案，您必须知道档案 ID。步骤 1-4 将检索档案 ID。如果您已经知道要下载的档案 ID，请跳至第 5 步。

1. 使用 `initiate-job` 命令启动清单检索任务。清单报告将列出您的档案 ID。

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

预期输出：

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

2. 使用 `describe-job` 命令检查上一个 任务的状态。

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

预期输出：

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

3. 等待作业完成。

   您必须等到任务输出已作好供您下载的准备。如果您在文件库中设置了通知配置，或者在启动任务时指定了 Amazon Simple Notification Service (Amazon SNS) 主题，则 S3 Glacier 会在完成任务后向该主题发送消息。

   您可以设置文件库的特定事件的通知配置。有关更多信息，请参阅 在 Amazon S3 Glacier 中配置文件库通知。只要发生特定事件，S3 Glacier 就会向指定的 SNS 主题发送消息。

4. 完成后，使用 `get-job-output` 命令将检索作业下载到文件 `output.json`。此文件将包含您的档案 ID。

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

   此命令会生成一个包含以下字段的文件。

```
{
"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. 使用 `initiate-job` 命令启动检索文件库中每个档案的过程。您需要指定任务参数，如下面所示的 `archive-retrieval`。

```
aws glacier initiate-job --vault-name awsexamplevault --account-id 111122223333
 --job-parameters="{\"Type\":\"archive-retrieval\",\"ArchiveId\":\"*** archiveId
 ***\"}"
```

6. 等待 `archive-retrieval` 任务完成。使用 `describe-job` 命令检查上一个命令的状态。

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

7. 完成上述任务后，使用 `get-job-output` 命令下载您的档案。

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

删除 Amazon S3 Glacier 中的档案

无法使用 Amazon S3 Glacier（S3 Glacier）管理控制台删除档案。要删除档案，必须使用 AWS Command Line Interface (CLI) 或编写代码，直接使用 REST API 或使用 适用于 Java 的 AWS SDK 和.NET 包装器库发出删除请求。以下主题说明了如何使用 适用于 Java 的 AWS SDK 和.NET 包装器库、REST API 和。 AWS CLI

主题

- 使用 适用于 Java 的 AWS SDK
- 使用 Amazon S3 Glacier 删除档案 适用于 .NET 的 AWS SDK
- 使用 REST API 删除 Amazon S3 Glacier 档案
- 使用 AWS Command Line Interface在 Amazon S3 Glacier 中删除档案

您可以从文件库一次删除一个档案。要删除档案，您必须在删除请求中提供档案 ID。您可以通过下载包含下载档案的文件库的文件库清单来获取档案 ID。有关下载文件库清单的更多信息，请参阅在 Amazon S3 Glacier 中下载文件库清单。

在删除档案后，您仍可能成功请求启动对已删除档案的检索任务，但档案检索任务会失败。

在您删除档案时，对相应档案 ID 正在进行的档案检索可能成功，也可能不成功，具体取决于下面的场景：

- 如果 S3 Glacier 收到删除档案请求时，档案检索任务正在积极地为下载准备数据，则档案检索操作可能会失败。
- 如果 S3 Glacier 收到删除档案请求时，档案检索任务已成功地为下载准备好档案，则您将能够下载输出。

有关档案检索的更多信息，请参阅“在 S3 Glacier 中下载档案”。

此操作是幂等的。删除已删除的档案不会导致错误。

删除档案后，如果您立即下载文件库清单，则它可能会在列表中包括已删除的档案，因为 S3 Glacier 每天大约只准备一次文件库清单。

> **Note**
>
> 有关自动删除文件库存档的信息，请参阅在 Amazon S3 Glacier 中自动删除文件库档案。

使用 适用于 Java 的 AWS SDK

以下是使用 适用于 Java 的 AWS SDK 低级 API 删除档案的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定存储要删除的档案的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过创建一个 `DeleteArchiveRequest` 类的实例提供请求信息。

   您需要提供档案 ID、文件库名称和您的账户 ID。如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

3. 以参数形式提供请求对象，运行 `deleteArchive` 方法。

以下 Java 代码段说明了前面的步骤。

```
AmazonGlacierClient client;

DeleteArchiveRequest request = new DeleteArchiveRequest()
    .withVaultName("*** provide a vault name ***")
    .withArchiveId("*** provide an archive ID ***");

client.deleteArchive(request);
```

> **Note**
>
> 有关底层 REST API 的信息，请参阅删除档案。

示例：使用删除档案 适用于 Java 的 AWS SDK

以下 Java 代码示例 适用于 Java 的 AWS SDK 使用删除档案。有关如何运行此示例的 step-by-step说明，请参阅使用 Eclipse 运行 Amazon S3 Glacier 的 Java 示例。您需要更新文件库名称和待删除档案 ID 旁显示的代码。

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);
        }
    }
}
```

使用 Amazon S3 Glacier 删除档案 适用于 .NET 的 AWS SDK

适用于.NET 的 Amazon SDK APIs 提供的高级版本和低级版本都提供了一种删除档案的方法。

主题

- 使用的高级别 API 删除档案 适用于 .NET 的 AWS SDK
- 使用低级 API 删除档案 适用于 .NET 的 AWS SDK

使用的高级别 API 删除档案 适用于 .NET 的 AWS SDK

该高级 API 的 `ArchiveTransferManager` 类提供了您可以用来删除档案的 `DeleteArchive` 方法。

示例：使用的高级别 API 删除档案 适用于 .NET 的 AWS SDK

以下 C# 代码示例使用的高级别 API 适用于 .NET 的 AWS SDK 来删除档案。有关如何运行此示例的 step-by-step说明，请参阅运行代码示例。您需要更新待删除档案 ID 旁显示的代码。

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();
    }
  }
}
```

使用低级 API 删除档案 适用于 .NET 的 AWS SDK

以下是使用 适用于 .NET 的 AWS SDK删除档案的步骤。

1. 创建 `AmazonGlacierClient` 类（客户端）的实例。

   您需要指定存储要删除的档案的 AWS 区域。您使用此客户端执行的所有操作都适用于该 AWS 区域。

2. 通过创建一个 `DeleteArchiveRequest` 类的实例提供请求信息。

   您需要提供档案 ID、文件库名称和您的账户 ID。如果您不提供账户 ID，则系统会使用与您提供来对请求签名的证书相关联的账户 ID。有关更多信息，请参阅 AWS SDKs 与 Amazon S3 Glacier 搭配使用。

3. 以参数形式提供请求对象，运行 `DeleteArchive` 方法。

示例：使用的低级 API 删除档案 适用于 .NET 的 AWS SDK

以下 C# 示例说明了前面的步骤。该示例使用的低级 API 适用于 .NET 的 AWS SDK 来删除档案。

> **Note**
>
> 有关底层 REST API 的信息，请参阅删除档案。

有关如何运行此示例的 step-by-step说明，请参阅运行代码示例。您需要更新待删除档案 ID 旁显示的代码。

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);
      }
    }
  }
```

使用 REST API 删除 Amazon S3 Glacier 档案

您可以使用删除档案 API 来删除档案。

- 有关删除档案 API 的更多信息，请参阅 删除档案。
- 有关使用 REST API 的信息，请参阅 Amazon S3 Glacier 的 API 参考。

使用 AWS Command Line Interface在 Amazon S3 Glacier 中删除档案

您可以使用 () 删除亚马逊 S3 Glacier (S3 Glacier AWS CLI) 中的 AWS Command Line Interface 档案。

主题

- （先决条件）设置 AWS CLI
- 示例：使用删除档案 AWS CLI

（先决条件）设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   正在安装 AWS Command Line Interface

   正在配置 AWS Command Line Interface

2. 在命令提示符下输入以下命令来验证您的 AWS CLI 设置。这些命令没有显式提供凭证，因此将使用默认配置文件的凭证。

   - 尝试使用 help 命令。

     ```
     aws help
     ```

   - 要获取已配置账户上 S3 Glacier 文件库的列表，请使用 `list-vaults` 命令。*123456789012*用您的 AWS 账户 身份证替换。

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

- 要查看的当前配置数据 AWS CLI，请使用`aws configure list`命令。

```
aws configure list
```

示例：使用删除档案 AWS CLI

1. 使用 initiate-job命令启动库存取回任务。

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

预期输出：

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

2. 使用 describe-job命令来检查上一个检索任务的状态。

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

预期输出：

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

```
    "StatusCode": "InProgress"
}
```

3. 等待作业完成。

   您必须等到任务输出已作好供您下载的准备。如果您在文件库中设置了通知配置，或者在启动任务时指定了 Amazon Simple Notification Service (Amazon SNS) 主题，则 S3 Glacier 会在完成任务后向该主题发送消息。

   您可以设置文件库的特定事件的通知配置。有关更多信息，请参阅 在 Amazon S3 Glacier 中配置文件库通知。只要发生特定事件，S3 Glacier 就会向指定的 SNS 主题发送消息。

4. 完成后，使用 get-job-output命令将检索任务下载到文件中`output.json`。

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

   此命令会生成一个包含以下字段的文件。

```
{
"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. 使用 `delete-archive` 命令从文件库中删除每个存档，直到不保留任何存档。

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

AWS SDKs 与 Amazon S3 Glacier 搭配使用

AWS 允许您 SDKs 为 Amazon S3 Glacier 开发应用程序。SDK 库包含底层 S3 Glacier API，可以简化您的编程任务。例如，对于发送到 S3 Glacier 的每个请求，必须包括签名才能验证您的请求。使用 SDK 库时，您只需要在代码中提供 AWS 安全证书，库会计算必要的签名并将其包含在发送到 S3 Glacier 的请求中。它们 AWS SDKs 提供了映射到底层 REST API 的库，并提供了可用于轻松构造请求和处理响应的对象。

主题

- AWS Java 和.NET 的 SDK 库
- 将 S3 Glacier 与 S AWS DK 配
- 适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用
- 适用于 .NET 的 AWS SDK 与 Amazon S3 Glacier 搭配使用

AWS Command Line Interface (AWS CLI) 是一款用于管理您 AWS 服务的（包括 S3 Glacier）的统一工具。有关下载的信息 AWS CLI，请参阅AWS Command Line Interface。有关 S3 Glacier CLI 命令的列表，请参阅《AWS CLI 命令参考》https://docs.aws.amazon.com/cli/latest/reference/glacier/index.html。

AWS Java 和.NET 的 SDK 库

AWS SDKs 适用于 Java 和.NET 的提供了高级和低级封装库。

您可以使用本开发者指南中的 适用于 Java 的 AWS SDK 和来查找使用 Amazon S3 Glacier 的 适用于 .NET 的 AWS SDK 示例。

什么是低级 API？

低级包装程序库紧密映射 S3 Glacier 支持的底层 REST API（Amazon S3 Glacier 的 API 参考）。对于每个 S3 Glacier REST 操作，低级 API 均提供了相应的方法、用于提供请求信息的请求对象，以及用于处理 S3 Glacier 响应的响应对象。低级包装程序库是底层 S3 Glacier 操作的最完整实现。

有关这些软件开发工具包库的信息，请参阅适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用和适用于 .NET 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

什么是高级 API？

为了进一步简化应用程序开发，这些库为某些操作提供了更高级的抽象。例如：

- 上传档案—要使用低级 API 上传档案，则除了文件名以及您要在其中保存档案的文件库名称以外，您还需要提供有效负载的校验和（SHA-256 树形哈希）。但是，高级 API 会为您计算校验和。
- 下载档案或文件库清单—要使用低级 API 下载档案，您首先要启动任务，等待任务完成，然后获取任务输出。您需要编写附加代码来设置 Amazon Simple Notification Service (Amazon SNS) 主题，以便 S3 Glacier 在作业完成时通知您。此外，您还需要使用某个轮询机制来检查任务完成消息是否已发布到该主题。该高级 API 提供了用于下载档案的方法，该方法会处理所有的这些步骤。您只需指定档案 ID 以及您要保存下载的数据的文件夹路径。

有关这些软件开发工具包库的信息，请参阅适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用和适用于 .NET 的 AWS SDK 与 Amazon S3 Glacier 搭配使用。

何时使用高级和低级 API？

通常，如果高级 API 提供了您执行操作所需的方法，则您应使用高级 API，因为这样比较简单。但是，如果高级 API 没有提供该功能，则您可以使用低级 API。此外，低级 API 还允许对操作进行粒度控制（例如，在操作失败时执行重试逻辑）。例如，上传档案时，高级 API 会使用文件大小来确定是在单一操作中上传档案，还是使用分段上传 API。此外，该 API 还具有内置的重试逻辑，可以在上传失败时执行。但是，您的应用程序可能需要对这些决定进行粒度控制，在这种情况下，您可以使用低级 API。

将 S3 Glacier 与 S AWS DK 配

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

SDK 文档	代码示例
适用于 C++ 的 AWS SDK	适用于 C++ 的 AWS SDK 代码示例
AWS CLI	AWS CLI 代码示例
适用于 Go 的 AWS SDK	适用于 Go 的 AWS SDK 代码示例
适用于 Java 的 AWS SDK	适用于 Java 的 AWS SDK 代码示例

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

有关特定于 S3 Glacier 的示例，请参阅使用 S3 Glacier 的代码示例 AWS SDKs。

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

适用于 Java 的 AWS SDK 与 Amazon S3 Glacier 搭配使用

为亚马逊 S3 Glacier（S3 Glacier） 适用于 Java 的 AWS SDK 提供了高级和低级 APIs ，如中所述。AWS SDKs 与 Amazon S3 Glacier 搭配使用有关下载的信息 适用于 Java 的 AWS SDK，请参阅适用于 Java 的亚马逊 SDK。

> **Note**
>
> 为访问 S3 Glacier 适用于 Java 的 AWS SDK 提供了线程安全的客户端。应用程序应创建一个客户端并在线程之间重复使用此客户端，您应将此作为一项最佳实践。

主题

- 使用低级 API
- 使用高级 API
- 使用 Eclipse 运行 Amazon S3 Glacier 的 Java 示例
- 设置端点

使用低级 API

低级 `AmazonGlacierClient` 类提供了映射到 S3 Glacier (Amazon S3 Glacier 的 API 参考) 的底层 REST 操作的所有方法。调用其中任何一种方法时，您都必须创建相应的请求对象，并提供响应对象；在该响应对象中，方法可以向操作返回 S3 Glacier 响应。

例如，`AmazonGlacierClient` 类提供了 `createVault` 方法来创建文件库。此方法会映射到底层的创建文件库 REST 操作（请参阅“创建文件库（设置文件库）”）。要使用此方法，您必须创建接收 S3 Glacier 响应的 `CreateVaultResult` 对象的实例，如以下 Java 代码段所示：

```
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);
```

指南中的所有低级示例都使用了此模式。

> **Note**
>
> 在创建请求时，前面的代码段会指定 `AccountID`。但是，使用时 适用于 Java 的 AWS SDK，请求`AccountId`中的是可选的，因此本指南中的所有低级示例都没有设置此值。`AccountId`就是 AWS 账户 身份证。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定 AWS 账户 ID，也可以选择指定 '-'，在这种情况下，S3 Glacier 使用与用于签署请求的凭据关联的 AWS 账户 ID。如果您指定您的账户 ID，请勿在其中包括连字符。使用时 适用于 Java 的 AWS SDK，如果您不提供账户 ID，则库会将账户 ID 设置为 '-'。

使用高级 API

为了进一步简化应用程序开发， 适用于 Java 的 AWS SDK 提供了为低级 API 中的某些方法实现更高级别抽象的`ArchiveTransferManager`类。它为归档操作提供了有用的方法，例如 `upload` 和 `download` 方法。

例如，以下 Java 代码段使用了 `upload` 高级方法来上传档案。

```
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();
```

请注意，您执行的任何操作都适用于您在创建`ArchiveTransferManager`对象时指定的 AWS 区域。如果您未指定任何 AWS 区域，则将 适用于 Java 的 AWS SDK 设置`us-east-1`为默认 AWS 区域。

此指南中的所有高级示例都使用了此模式。

> **Note**
>
> 高级 `ArchiveTransferManager` 类可以通过 `AmazonGlacierClient` 实例或 `AWSCredentials` 实例来构造。

使用 Eclipse 运行 Amazon S3 Glacier 的 Java 示例

入门 Java 代码示例的最简单方式是安装最新的 AWS Toolkit for Eclipse。有关安装或更新至工具包最新版本的信息，请转到 http://aws.amazon.com/eclipse。以下任务将引导您创建和测试本节中提供的 Java 代码示例。

创建 Java 代码示例的常规过程

1	按照在 A mazon SDK fo AWS r Java 中提供 AWS 凭证 适用于 Java 的 AWS SDK 主题中所述，为您的证书创建默认凭证配置文件。

2	在 Eclipse 中创建一个新的 AWS Java 项目。此项目使用 适用于 Java 的 AWS SDK进行了预先配置。
3	将代码从您正在阅读的部分复制到您的项目。
4	通过提供任意所需数据更新代码。例如，如果上传某个文件，则提供文件路径和存储桶名称。
5	运行该代码。验证是否使用 AWS Management Console创建了对象。有关该的更多信息 AWS Management Console，请访问 http://aws.amazon.com/console/。

设置端点

默认情况下， 适用于 Java 的 AWS SDK 使用终端节点`https://glacier.us-east-1.amazonaws.com`。您可以显式设置端点，如以下 Java 代码段所示。

以下代码段说明如何使用低级 API 将端点设置为美国西部（俄勒冈州）区域 (`us-west-2`)。

Example

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

以下代码段说明如何使用高级 API 将端点设置为美国西部（俄勒冈州）区域。

```
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);
```

有关支持的 AWS 区域和终端节点的列表，请参阅访问 Amazon S3 Glacier。

适用于 .NET 的 AWS SDK 与 Amazon S3 Glacier 搭配使用

适用于 .NET 的 AWS SDK API 在中可用`AWSSDK.dll`。有关下载的信息 适用于 .NET 的 AWS SDK，请转到示例代码库。如中所述AWS SDKs 与 Amazon S3 Glacier 搭配使用， 适用于 .NET 的 AWS SDK 提供了高级和低级别 APIs。

> **Note**
>
> 低级 API 和高级 API 提供线程安全客户端来访问 S3 Glacier。应用程序应创建一个客户端并在线程之间重复使用此客户端，您应将此作为一项最佳实践。

主题

- 使用低级 API
- 使用高级 API
- 运行代码示例
- 设置端点

使用低级 API

低级 `AmazonGlacierClient` 类提供了映射到 Amazon S3 Glacier（S3 Glacier）(Amazon S3 Glacier 的 API 参考) 的底层 REST 操作的所有方法。调用其中任何一种方法时，您都必须创建相应的请求对象，并提供响应对象，所调用的方法可以在响应对象中向操作返回 S3 Glacier 响应。

例如，`AmazonGlacierClient` 类提供了 `CreateVault` 方法来创建文件库。此方法会映射到底层的创建文件库 REST 操作（请参阅“创建文件库（设置文件库）”）。要使用此方法，您必须创建 `CreateVaultRequest` 和 `CreateVaultResponse` 类的实例，以提供请求信息和接收 S3 Glacier 响应，如以下 C# 代码段所示：

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

CreateVaultRequest request = new CreateVaultRequest()
{
  AccountId = "-",
  VaultName = "*** Provide vault name ***"
};
```

```
CreateVaultResponse response = client.CreateVault(request);
```

指南中的所有低级示例都使用了此模式。

> **Note**
>
> 在创建请求时，前面的代码段会指定 `AccountId`。但是，使用时 适用于 .NET 的 AWS SDK，请求`AccountId`中的是可选的，因此本指南中的所有低级示例都没有设置此值。`AccountId`就是 AWS 账户 身份证。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定 AWS 账户 ID，也可以选择指定 '-'，在这种情况下，S3 Glacier 使用与用于签署请求的凭据关联的 AWS 账户 ID。如果您指定您的账户 ID，请勿在其中包括连字符。使用时 适用于 .NET 的 AWS SDK，如果您不提供账户 ID，则库会将账户 ID 设置为 '-'。

使用高级 API

为了进一步简化应用程序开发， 适用于 .NET 的 AWS SDK 提供了为低`ArchiveTransferManager`级 API 中的某些方法实现更高级别抽象的类。它为归档操作提供了有用的方法，例如 `Upload` 和 `Download`。

例如，以下 C# 代码段使用了 `Upload` 高级方法来上传档案。

```
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;
```

请注意，您执行的任何操作都适用于您在创建`ArchiveTransferManager`对象时指定的 AWS 区域。此指南中的所有高级示例都使用了此模式。

> **Note**
>
> 高级 `ArchiveTransferManager` 类仍然需要低级 `AmazonGlacierClient` 客户端，您可以显式传递该客户端或者让 `ArchiveTransferManager` 创建该客户端。

运行代码示例

入门 .NET 代码示例的最简单方式是安装 适用于 .NET 的 AWS SDK。有关更多信息，请转到适用于 .NET 的 Amazon SDK。

以下过程为您概述了测试此指南中提供的代码示例的步骤。

创建 .NET 代码示例的一般流程（使用 Visual Studio）

1	按照适用于.NET 的 Amazon SDK 主题配置 AWS 凭证中所述，为您的证书创建 AWS 凭证配置文件。
2	使用 AWS 空项目模板创建新的 Visual Studio 项目。
3	将项目文件 `Program.cs` 中的代码替换为您正在阅读的部分中的代码。
4	运行该代码。验证是否使用 AWS Management Console创建了对象。有关的更多信息 AWS Management Console，请访问 http://aws.amazon.com/console/。

设置端点

默认情况下，将终端节点 适用于 .NET 的 AWS SDK 设置为美国西部（俄勒冈）区域 (`https://glacier.us-west-2.amazonaws.com`)。您可以将终端节点设置为其他 AWS 区域，如以下 C# 片段所示。

以下代码段说明如何使用低级 API 将端点设置为美国西部（俄勒冈州）区域 (`us-west-2`)。

Example

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

以下代码段说明如何使用高级 API 将端点设置为美国西部（俄勒冈州）区域。

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

有关支持的 AWS 区域和终端节点的最新列表，请参阅访问 Amazon S3 Glacier。

使用 S3 Glacier 的代码示例 AWS SDKs

以下代码示例展示了如何将 S3 Glacier 与 AWS 软件开发套件 (SDK) 一起使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务服务结合来完成特定任务的代码示例。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用

Hello Amazon S3 Glacier

以下代码示例展示了如何开始使用 Amazon S3 Glacier。

.NET

适用于 .NET 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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}");
        }
    }
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考ListVaults中的。

代码示例

- 使用 S3 Glacier 的基本示例 AWS SDKs
  - Hello Amazon S3 Glacier
  - 使用 S3 Glacier 执行的操作 AWS SDKs
    - AddTagsToVault与 AWS SDK 或 CLI 配合使用
    - CreateVault与 AWS SDK 或 CLI 配合使用
    - DeleteArchive与 AWS SDK 或 CLI 配合使用
    - DeleteVault与 AWS SDK 或 CLI 配合使用
    - DeleteVaultNotifications与 AWS SDK 或 CLI 配合使用
    - DescribeJob与 AWS SDK 或 CLI 配合使用
    - DescribeVault与 AWS SDK 或 CLI 配合使用
    - GetJobOutput与 AWS SDK 或 CLI 配合使用
    - GetVaultNotifications与 AWS SDK 或 CLI 配合使用
    - InitiateJob与 AWS SDK 或 CLI 配合使用
    - ListJobs与 AWS SDK 或 CLI 配合使用
    - ListTagsForVault与 AWS SDK 或 CLI 配合使用
    - ListVaults与 AWS SDK 或 CLI 配合使用

    - SetVaultNotifications与 AWS SDK 或 CLI 配合使用
    - UploadArchive与 AWS SDK 或 CLI 配合使用
    - UploadMultipartPart与 AWS SDK 或 CLI 配合使用
- 使用 S3 Glacier 的场景 AWS SDKs
  - 使用 AWS 软件开发工具包将文件存档到 Amazon S3 Glacier，获取通知并启动任务
  - 获取 Amazon S3 Glacier 档案内容并使用 AWS 软件开发工具包删除档案

使用 S3 Glacier 的基本示例 AWS SDKs

以下代码示例展示了如何使用 Amazon S3 Glacier 的基础知识 AWS SDKs。

示例

- Hello Amazon S3 Glacier
- 使用 S3 Glacier 执行的操作 AWS SDKs
  - AddTagsToVault与 AWS SDK 或 CLI 配合使用
  - CreateVault与 AWS SDK 或 CLI 配合使用
  - DeleteArchive与 AWS SDK 或 CLI 配合使用
  - DeleteVault与 AWS SDK 或 CLI 配合使用
  - DeleteVaultNotifications与 AWS SDK 或 CLI 配合使用
  - DescribeJob与 AWS SDK 或 CLI 配合使用
  - DescribeVault与 AWS SDK 或 CLI 配合使用
  - GetJobOutput与 AWS SDK 或 CLI 配合使用
  - GetVaultNotifications与 AWS SDK 或 CLI 配合使用
  - InitiateJob与 AWS SDK 或 CLI 配合使用
  - ListJobs与 AWS SDK 或 CLI 配合使用
  - ListTagsForVault与 AWS SDK 或 CLI 配合使用
  - ListVaults与 AWS SDK 或 CLI 配合使用
  - SetVaultNotifications与 AWS SDK 或 CLI 配合使用
  - UploadArchive与 AWS SDK 或 CLI 配合使用
  - UploadMultipartPart与 AWS SDK 或 CLI 配合使用

Hello Amazon S3 Glacier

以下代码示例展示了如何开始使用 Amazon S3 Glacier。

.NET

适用于 .NET 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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}");
        }
    }
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考ListVaults中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 S3 Glacier 执行的操作 AWS SDKs

以下代码示例演示了如何使用执行单个 S3 Glacier 操作 AWS SDKs。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 S3 Glacier API，是必须在上下文中运行的大型程序的代码节选。您可以在使用 S3 Glacier 的场景 AWS SDKs 中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon S3 Glacier API 参考。

示例

- AddTagsToVault与 AWS SDK 或 CLI 配合使用
- CreateVault与 AWS SDK 或 CLI 配合使用
- DeleteArchive与 AWS SDK 或 CLI 配合使用
- DeleteVault与 AWS SDK 或 CLI 配合使用
- DeleteVaultNotifications与 AWS SDK 或 CLI 配合使用
- DescribeJob与 AWS SDK 或 CLI 配合使用
- DescribeVault与 AWS SDK 或 CLI 配合使用
- GetJobOutput与 AWS SDK 或 CLI 配合使用
- GetVaultNotifications与 AWS SDK 或 CLI 配合使用
- InitiateJob与 AWS SDK 或 CLI 配合使用
- ListJobs与 AWS SDK 或 CLI 配合使用
- ListTagsForVault与 AWS SDK 或 CLI 配合使用
- ListVaults与 AWS SDK 或 CLI 配合使用
- SetVaultNotifications与 AWS SDK 或 CLI 配合使用
- UploadArchive与 AWS SDK 或 CLI 配合使用
- UploadMultipartPart与 AWS SDK 或 CLI 配合使用

AddTagsToVault与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 AddTagsToVault。

.NET

适用于 .NET 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// 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;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考AddTagsToVault中的。

CLI

AWS CLI

以下命令向名为 `my-vault` 的文件库中添加两个标签：

```
aws glacier add-tags-to-vault --account-id - --vault-name my-vault --
tags id=1234,date=july2015
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AddTagsToVault中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateVault`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateVault`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 归档文件，获取通知并启动任务

.NET

适用于 .NET 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Create an 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;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考CreateVault中的。

CLI

AWS CLI

以下命令创建名为 `my-vault` 的新文件库：

```
aws glacier create-vault --vault-name my-vault --account-id -
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateVault中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateVault中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建客户端。

```
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 };
```

创建文件库。

```
// 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();
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考CreateVault中的。

适用于 JavaScript (v2) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
// 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!");
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考CreateVault中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：为用户账户创建新文件库。由于未向-AccountId 参数提供任何值，因此 cmdlet 使用默认值 “-” 来表示当前账户。

```
New-GLCVault -VaultName myvault
```

输出：

```
/01234567812/vaults/myvault
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) CreateVault中的。

适用于 PowerShell V5 的工具

示例 1：为用户账户创建新文件库。由于未向-AccountId 参数提供任何值，因此 cmdlet 使用默认值 “-” 来表示当前账户。

```
New-GLCVault -VaultName myvault
```

输出：

```
/01234567812/vaults/myvault
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) CreateVault中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class 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
```

- 有关 API 的详细信息，请参阅适用CreateVault于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteArchive`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteArchive`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 获取档案内容并删除档案

CLI

AWS CLI

从文件库中删除存档

以下 `delete-archive` 示例从 `example_vault` 中删除指定存档。

```
aws glacier delete-archive \
    --account-id 111122223333  \
    --vault-name example_vault \
    --archive-id Sc0u9ZP8yaWkmh-XGlIvAVprtLhaLCGnNwNl5I5x9HqPIkX5mjc0DrId3Ln-
Gi_k2HzmlIDZUz117KSdVMdMXLuFWi9PJUitxWO73edQ43eTlMWkH0pd9zVSAuV_XXZBVhKhyGhJ7w
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteArchive中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteArchive中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class 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
```

- 有关 API 的详细信息，请参阅适用DeleteArchive于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteVault`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteVault`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 获取档案内容并删除档案

CLI

AWS CLI

以下命令删除名为 `my-vault` 的文件库：

```
aws glacier delete-vault --vault-name my-vault --account-id -
```

此命令不生成任何输出。Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteVault中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteVault中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class 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
```

- 有关 API 的详细信息，请参阅适用DeleteVault于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteVaultNotifications`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteVaultNotifications`。

CLI

AWS CLI

删除文件库的 SNS 通知

以下 `delete-vault-notifications` 示例演示了如何删除 Amazon Simple Notification Service (Amazon SNS) 针对指定文件库发送的通知。

```
aws glacier delete-vault-notifications \
    --account-id 111122223333 \
    --vault-name example_vault
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteVaultNotifications中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class 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
```

- 有关 API 的详细信息，请参阅适用DeleteVaultNotifications于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeJob`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeJob`。

CLI

AWS CLI

以下命令检索名为 `my-vault` 的文件库中有关库存检索任务的信息：

```
aws glacier describe-job --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW
```

输出：

```
{
    "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"
}
```

作业 ID 可以在 `aws glacier initiate-job` 和 `aws glacier list-jobs` 的输出中找到。Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeJob中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回指定任务的详细信息。任务成功完成后，可以使用 Read-O GCJob utput cmdlet 将任务的内容（档案或清单列表）检索到本地文件系统。

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) DescribeJob中的。

适用于 PowerShell V5 的工具

示例 1：返回指定任务的详细信息。任务成功完成后，可以使用 Read-O GCJob utput cmdlet 将任务的内容（档案或清单列表）检索到本地文件系统。

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) DescribeJob中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class 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
```

- 有关 API 的详细信息，请参阅适用DescribeJob于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeVault`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeVault`。

.NET

适用于 .NET 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// 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;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考DescribeVault中的。

CLI

AWS CLI

以下命令检索名为 `my-vault` 的文件库的相关数据：

```
aws glacier describe-vault --vault-name my-vault --account-id -
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVault中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetJobOutput`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetJobOutput`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 获取档案内容并删除档案

CLI

AWS CLI

以下命令将文件库清单任务的输出保存到名为 `output.json` 的当前目录中的某个文件中：

```
aws glacier get-job-output --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW output.json
```

可在 `aws glacier list-jobs` 输出中找到 `job-id`。请注意，输出文件名是一个位置参数，不以选项名称作为前缀。Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

输出：

```
{
    "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
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考GetJobOutput中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：下载计划在指定任务中检索的档案内容，并将这些内容存储到磁盘上的文件中。如果有校验和，则下载会为您验证校验和。如果需要，可以通过指定 **`-Select '*'`** 来返回包括校验和在内的整个响应。

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) GetJobOutput中的。

适用于 PowerShell V5 的工具

示例 1：下载计划在指定任务中检索的档案内容，并将这些内容存储到磁盘上的文件中。如果有校验和，则下载会为您验证校验和。如果需要，可以通过指定 **-Select '*'** 来返回包括校验和在内的整个响应。

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) GetJobOutput中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class 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
```

- 有关 API 的详细信息，请参阅适用GetJobOutput于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetVaultNotifications`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetVaultNotifications`。

CLI

AWS CLI

以下命令获取名为 `my-vault` 的文件库的通知配置的描述：

```
aws glacier get-vault-notifications --account-id - --vault-name my-vault
```

输出：

```
{
    "vaultNotificationConfig": {
        "Events": [
            "InventoryRetrievalCompleted",
            "ArchiveRetrievalCompleted"
        ],
        "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault"
    }
}
```

如果没有为该文件库配置任何通知，则会返回错误。Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考GetVaultNotifications中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class 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
```

- 有关 API 的详细信息，请参阅适用GetVaultNotifications于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`InitiateJob`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `InitiateJob`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 归档文件，获取通知并启动任务

.NET

适用于 .NET 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

从文件库中检索档案。此示例使用 ArchiveTransferManager 类。有关 API 的详细信息，请参阅ArchiveTransferManager。

```
    /// <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}%");
        }
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考InitiateJob中的。

CLI

AWS CLI

以下命令启动作业以获取文件库 `my-vault` 的清单：

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

输出：

```
{
```

```
    "location": "/0123456789012/vaults/my-vault/jobs/
zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
    "jobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW"
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

以下命令启动一个作业以从文件库 `my-vault` 检索归档：

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-
parameters file://job-archive-retrieval.json
```

`job-archive-retrieval.json` 是本地文件夹中的一个 JSON 文件，用于指定作业类型、归档 ID 和一些可选参数：

```
{
  "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"
}
```

存档文件 IDs 可在`aws glacier upload-archive`和的输出中找到`aws glacier get-job-output`。

输出：

```
{
    "location": "/011685312445/vaults/mwunderl/jobs/l7IL5-
EkXyEY9Ws95fClzIbk2O5uLYaFdAYOi-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav",
    "jobId": "l7IL5-EkXy2O5uLYaFdAYOiEY9Ws95fClzIbk-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav"
}
```

有关作业参数格式的详细信息，请参阅《Amazon Glacier API 参考》中的“启动作业”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考InitiateJob中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

检索文件库清单。

```
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);

        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考InitiateJob中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：启动一项任务，从用户拥有的指定文件库中检索档案。可以使用 Get-GLCJob cmdlet 检查作业的状态。任务成功完成后，可以使用读取-GCJob 输出 cmdlet 将存档内容检索到本地文件系统。

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

输出：

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) InitiateJob中的。

适用于 PowerShell V5 的工具

示例 1：启动一项任务，从用户拥有的指定文件库中检索档案。可以使用 Get-GLCJob cmdlet 检查作业的状态。任务成功完成后，可以使用读取-GCJob 输出 cmdlet 将存档内容检索到本地文件系统。

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

输出：

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) InitiateJob中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

检索文件库清单。

```
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
```

从文件库中检索档案。

```
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
```

- 有关 API 的详细信息，请参阅适用InitiateJob于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListJobs`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListJobs`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 归档文件，获取通知并启动任务
- 获取档案内容并删除档案

.NET

适用于 .NET 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// List 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;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考ListJobs中的。

CLI

AWS CLI

以下命令列出了名为 `my-vault` 的文件库的正在进行和最近完成的作业：

```
aws glacier list-jobs --account-id - --vault-name my-vault
```

输出：

```
{
    "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 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListJobs中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class 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
```

- 有关 API 的详细信息，请参阅适用ListJobs于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListTagsForVault`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListTagsForVault`。

.NET

适用于 .NET 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// List 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;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考ListTagsForVault中的。

CLI

AWS CLI

以下命令列出应用于名为 `my-vault` 的文件库的标签：

```
aws glacier list-tags-for-vault --account-id - --vault-name my-vault
```

输出：

```
{
    "Tags": {
        "date": "july2015",
        "id": "1234"
    }
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListTagsForVault中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListVaults`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListVaults`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 归档文件，获取通知并启动任务

.NET

适用于 .NET 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
/// <summary>
/// List 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;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考ListVaults中的。

CLI

AWS CLI

以下命令列出默认账户和区域中的文件库：

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

输出：

```
{
    "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 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListVaults中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListVaults中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class 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
```

- 有关 API 的详细信息，请参阅适用ListVaults于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`SetVaultNotifications`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `SetVaultNotifications`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 归档文件，获取通知并启动任务

CLI

AWS CLI

以下命令为名为 `my-vault` 的文件库配置 SNS 通知：

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

`notificationconfig.json` 是当前文件夹中的一个 JSON 文件，用于指定 SNS 主题和要发布的事件：

```
{
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
  "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考SetVaultNotifications中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class 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
```

- 有关 API 的详细信息，请参阅适用SetVaultNotifications于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`UploadArchive`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `UploadArchive`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 归档文件，获取通知并启动任务

.NET

适用于 .NET 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// 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;
        }
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 AWS SDK API 参考UploadArchive中的。

CLI

AWS CLI

以下命令将名为 `archive.zip` 的当前文件夹中的存档上传到名为 `my-vault` 的文件库：

```
aws glacier upload-archive --account-id - --vault-name my-vault --
body archive.zip
```

输出：

```
{
    "archiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--
zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
    "checksum":
 "969fb39823836d81f0cc028195fcdbcbbe76cdde932d4646fa7de5f21e18aa67",
    "location": "/0123456789012/vaults/my-vault/archives/
kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw"
}
```

Amazon Glacier 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

要检索上传的存档，可使用 aws glacier initiate-job 命令启动检索作业。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考UploadArchive中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.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();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UploadArchive中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建客户端。

```
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 };
```

上传档案。

```
// 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();
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考UploadArchive中的。

适用于 JavaScript (v2) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
// 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);
```

```
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考UploadArchive中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：将单个文件上传到指定文件库，返回档案 ID 和计算出的校验和。

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

输出：

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

示例 2：将文件夹层次结构的内容上传到用户账户中的指定文件库。对于上传的每个文件，cmdlet 都会发出文件名、相应的档案 ID 和计算出的档案校验和。

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) UploadArchive中的。

适用于 PowerShell V5 的工具

示例 1：将单个文件上传到指定文件库，返回档案 ID 和计算出的校验和。

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

输出：

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

示例 2：将文件夹层次结构的内容上传到用户账户中的指定文件库。对于上传的每个文件，cmdlet 都会发出文件名、相应的档案 ID 和计算出的档案校验和。

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) UploadArchive中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class 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
```

- 有关 API 的详细信息，请参阅适用UploadArchive于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`UploadMultipartPart`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `UploadMultipartPart`。

CLI

AWS CLI

以下命令上传存档的前 1 MiB（1024 x 1024 字节）部分：

```
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 在执行操作时需要一个账户 ID 参数，但您可以使用连字符来指定正在使用的账户。

正文参数采用本地文件系统上分段文件的路径。范围参数采用 HTTP 内容范围，指示该分段在已完成的存档中占用的字节。上传 ID 由 `aws glacier initiate-multipart-upload` 命令返回，也可以使用 `aws glacier list-multipart-uploads` 获取它。

有关使用 CLI 分段上传到 Amazon Glacier 的更多信息，请参阅 AWS CL AWS I 用户指南中的使用亚马逊 Glacier。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考UploadMultipartPart中的。

JavaScript

适用于 JavaScript (v2) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 Buffer 对象的 1 兆字节区块的分段上传。

```
// 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);
        }
      });
    });
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 AWS SDK 开发人员指南》。
- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 AWS SDK API 参考UploadMultipartPart中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 S3 Glacier 的场景 AWS SDKs

以下代码示例向您展示了如何在 S3 Glacier 中使用实现常见场景 AWS SDKs。这些场景向您展示了如何通过调用 S3 Glacier 中的多个函数或与其他 AWS 服务结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 使用 AWS 软件开发工具包将文件存档到 Amazon S3 Glacier，获取通知并启动任务
- 获取 Amazon S3 Glacier 档案内容并使用 AWS 软件开发工具包删除档案

使用 AWS 软件开发工具包将文件存档到 Amazon S3 Glacier，获取通知并启动任务

以下代码示例展示了如何：

- 创建 Amazon S3 Glacier 文件库。
- 配置文件库，将通知发布到 Amazon SNS 主题。
- 将档案文件上传到文件库。
- 启动档案检索任务。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建一个包装 S3 Glacier 操作的类。

```
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
```

调用包装器类上的函数，以创建文件库并上传文件，然后将文件库配置为发布通知并启动检索档案的任务。

```
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.")
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的以下主题。
  - CreateVault
  - InitiateJob
  - ListJobs
  - ListVaults

- SetVaultNotifications
- UploadArchive

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

获取 Amazon S3 Glacier 档案内容并使用 AWS 软件开发工具包删除档案

以下代码示例展示了如何：

- 列出 Amazon S3 Glacier 文件库的任务并获取任务状态。
- 获取已完成的档案检索任务的输出。
- 删除档案。
- 删除文件库。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建一个包装 S3 Glacier 操作的类。

```
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
```

调用包装器类上的函数，以从已完成的任务中获取档案内容，然后删除档案。

```
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)
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的以下主题。
  - DeleteArchive
  - DeleteVault
  - GetJobOutput
  - ListJobs

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将 S3 Glacier 与 S AWS DK 配。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon S3 Glacier 的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以从专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构中受益。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的 安全性和云中 的安全性：

- 云安全 — AWS 负责保护在云中运行 AWS 服务的基础架构 AWS Cloud。 AWS 还为您提供可以安全使用的服务。作为 AWS 合规性计划的一部分，我们的安全措施的有效性定期由第三方审计员进行测试和验证。要了解适用于 Amazon S3 Glacier（S3 Glacier）的合规性计划，请参阅合规性计划范围内的AWS 服务。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您组织的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 S3 Glacier 时应用责任共担模型。以下主题说明如何配置 S3 Glacier 以实现您的安全性和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 S3 Glacier 资源。

主题

- Amazon S3 Glacier 中的数据保护
- 适用于 Amazon S3 Glacier 的 Identity and Access Management
- Amazon S3 Glacier 中的日志记录和监控
- Amazon S3 Glacier 的合规性验证
- Amazon S3 Glacier 中的故障恢复能力
- Amazon S3 Glacier 中的基础设施安全性

Amazon S3 Glacier 中的数据保护

Amazon S3 Glacier（S3 Glacier）为数据归档和长期备份提供高度持久的云存储。S3 Glacier 旨在提供 99.999999999% 的持久性，并提供全面的安全和合规功能，帮助您满足严格的监管要求。S3 Glacier 以冗余方式将数据存储在多个 AWS 可用区 (AZ) 和每个可用区内的多台设备上。为了提高持久性，在确认成功上传 AZs 之前，S3 Glacier 会将您的数据同步存储在多个位置。

有关 AWS 全球云基础设施的更多信息，请参阅全球基础架构。

出于数据保护目的，我们建议您保护 AWS 账户 凭证，仅向个人用户、群组或角色提供履行其工作职责所需的权限。

如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-2 版》。

主题

- 数据加密
- 密钥管理
- 互联网络流量保密性

数据加密

数据保护是指保护传输中（往返于 Amazon S3 Glacier 时）和静态数据（存储在 AWS 数据中心时）的数据。您可以使用安全套接字层 (SSL) 或客户端加密保护直接上传到 S3 Glacier 的传输中数据。

您也可以通过 Amazon S3 访问 S3 Glacier。Amazon S3 支持 Amazon S3 存储桶的生命周期配置，让您可以将对象转换为 S3 Glacier 存储类（用于档案）。通过生命周期策略在 Amazon S3 和 S3 Glacier 之间传输的数据使用 SSL 加密。

存储在 S3 Glacier 中的静态数据使用 256 位高级加密标准 (AES-256) 以及 AWS维护的密钥，自动进行服务器端加密。如果您希望管理自己的密钥，还可以在将数据存储到 S3 Glacier 之前使用客户端加密。有关为 Amazon S3 设置默认加密的信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的 Amazon S3 默认加密。

密钥管理

服务器端加密是静态数据加密，即，Amazon S3 Glacier 在将数据写入其数据中心时对数据进行加密，并在您访问时进行解密。只要您验证了您的请求并且拥有访问权限，您访问加密和未加密数据的方式就没有区别。

存储在 S3 Glacier 中的静态数据使用 AES-256 以及 AWS维护的密钥，自动进行服务器端加密。作为额外的保护措施， AWS 使用我们定期轮换的根密钥对密钥本身进行加密。

互联网络流量保密性

通过网络访问 Amazon S3 Glacier 是通过 AWS 发布的 APIs。客户端必须支持传输层安全性（TLS）1.2。我们建议使用 TLS 1.3 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码

套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Diffie-Hellman Ephemeral (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。此外，必须使用与 IAM 主体关联的访问密钥 ID 和秘密访问密钥来对请求进行签名，也可以使用 AWS Security Token Service (AWS STS) 生成临时安全证书来对请求进行签名。

VPC 端点

通过虚拟私有云 (VPC) 终端节点，您 AWS PrivateLink 无需互联网网关、NAT 设备、VPN 连接或连接，即可将您的 VPC 与支持的 AWS 服务和由其提供支持的 VPC 终端节点服务进行私密 AWS Direct Connect 连接。虽然 S3 Glacier 不直接支持 VPC 端点，但如果您将 S3 Glacier 作为与 Amazon S3 集成的存储层进行访问，则可以利用 Amazon Simple Storage Service (Amazon S3) VPC 端点。

有关 Amazon S3 生命周期配置以及将对象转换为 S3 Glacier 存储类的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的对象生命周期管理和转换对象。有关 VPC 端点的更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 端点。

适用于 Amazon S3 Glacier 的 Identity and Access Management

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制可以通过身份验证（登录）和授权（具有权限）使用 S3 Glacier 资源的人员。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

- 受众
- 使用身份进行身份验证
- 使用策略管理访问
- Amazon S3 Glacier 如何与 IAM 配合使用
- 适用于 Amazon S3 Glacier 的基于身份的策略示例
- 适用于 Amazon S3 Glacier 的基于资源的策略示例
- Amazon S3 Glacier 身份和访问管理问题排查
- API 权限参考

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您在 S3 Glacier 中所做的工作。

服务用户 – 如果使用 S3 Glacier 服务来完成任务，则您的管理员会为您提供所需的凭证和权限。当您使用更多 S3 Glacier 功能来完成工作时，您可能需要额外权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 S3 Glacier 中的功能，请参阅Amazon S3 Glacier 身份和访问管理问题排查。

服务管理员 – 如果您在公司负责管理 S3 Glacier 资源，则您可能具有 S3 Glacier 的完全访问权限。您有责任确定您的服务用户应访问哪些 S3 Glacier 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 S3 Glacier 搭配使用的更多信息，请参阅Amazon S3 Glacier 如何与 IAM 配合使用。

IAM 管理员 – 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 S3 Glacier 的访问的详细信息。要查看您可在 IAM 中使用的 S3 Glacier 基于身份的策略示例，请参阅适用于 Amazon S3 Glacier 的基于身份的策略示例。

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM 用户指南》中的用于签署 API 请求的AWS 签名版本 4。

无论使用何种身份验证方法，您都可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的多重身份验证和《IAM 用户指南》中的 IAM 中的AWS 多重身份验证。

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅 IAM 用户指南中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户 和应用程序中使用。有关 IAM Identity Center 的信息，请参阅 AWS IAM Identity Center 用户指南中的什么是 IAM Identity Center？。

IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的用例，应在需要时更新访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的群组，IAMAdmins并向该群组授予管理 IAM 资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的 IAM 用户的使用案例。

IAM 角色

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。要在中临时担任 IAM 角色 AWS Management Console，您可以从用户切换到 IAM 角色（控制台）。您可

以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的代入角色的方法。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问：要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅《IAM 用户指南》中的针对第三方身份提供商创建角色（联合身份验证）。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限：IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取：您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南中的 IAM 中的跨账户资源访问。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序 EC2 或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两项操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
  - 服务相关角色-服务相关角色是一种与服务相关联的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 A@@ mazon 上运行的应用程序 EC2 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这比在 EC2 实例中存储访问密钥更可取。要为 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建一个附加到该实例的实例配置文件。实例配置文件包含该角色，并允许在 EC2 实例上运行的程序获得临时证书。有关更多信息，请参阅 IAM 用户指南中的使用 IAM 角色向在 A mazon EC2 实例上运行的应用程序授予权限。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户托管策略定义自定义 IAM 权限。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持的服务示例 ACLs。 AWS WAF要了解更多信息 ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界：权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs) — SCPs 是 JSON 策略，用于指定中组织或组织单位 (OU) 的最大权限 AWS Organizations。 AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有帐户。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organization SCPs s 和的更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs) — RCPs 是 JSON 策略，您可以使用它来设置账户中资源的最大可用权限，而无需更新附加到您拥有的每个资源的 IAM 策略。RCP 限制成员账户中资源的权限，并可能影响身份（包括身份）的有效权限 AWS 账户根用户，无论这些身份是否属于您的组织。有关 Organizations 的更多信息 RCPs，包括 AWS 服务 该支持的列表 RCPs，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略：会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon S3 Glacier 如何与 IAM 配合使用

在使用 IAM 管理对 S3 Glacier 的访问之前，您应该了解哪些 IAM 功能可用于 S3 Glacier。

可用于 Amazon S3 Glacier 的 IAM 功能

IAM 特征	S3 Glacier 支持
基于身份的策略	是
基于资源的策略	是
策略操作	是
策略资源	是
策略条件键（特定于服务）	是
ACLs	否
ABAC（策略中的标签）	否
临时凭证	是
主体权限	否
服务角色	否
服务相关角色	否

要全面了解 S3 Glacier 和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中与 IAM 配合使用的AWS 服务。

适用于 S3 Glacier 的基于身份的策略

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

适用于 S3 Glacier 的基于身份的策略示例

要查看 S3 Glacier 基于身份的策略的示例，请参阅适用于 Amazon S3 Glacier 的基于身份的策略示例。

S3 Glacier 内基于资源的策略

支持基于资源的策略：是

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时 AWS 账户，可信账户中的 IAM 管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

S3 Glacier 服务仅支持一种基于资源的策略（称为文件库策略），这种策略附加到文件库。此策略定义了哪些主体可以对该文件库执行操作。

S3 Glacier 文件库策略按以下方式管理权限：

- 使用单个文件库策略而不是多个单用户策略管理账户中的用户权限。
- 作为使用 IAM 角色的替代方案，可以管理跨账户权限。

S3 Glacier 内基于资源的策略示例

要查看 S3 Glacier 基于资源的策略的示例，请参阅适用于 Amazon S3 Glacier 的基于资源的策略示例。

S3 Glacier 策略操作

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

要查看 S3 Glacier 操作的列表，请参阅《服务授权参考》中的 Amazon S3 Glacier 定义的操作。

S3 Glacier 中的策略操作在操作前使用以下前缀：

```
glacier
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
            "glacier:CreateVault",
            "glacier:DescribeVault",
            "glacier:ListVaults"
         ]
```

您也可以使用通配符（*）指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，包括以下操作：

```
"Action": "glacier:GetVault*"
```

要查看 S3 Glacier 基于身份的策略的示例，请参阅适用于 Amazon S3 Glacier 的基于身份的策略示例。

S3 Glacier 策略资源

支持策略资源：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符（*）指示语句应用于所有资源。

```
"Resource": "*"
```

要查看 S3 Glacier 资源类型及其列表 ARNs，请参阅《服务授权参考》中的 Amazon S3 Glacier 定义的资源。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 Amazon S3 Glacier 定义的操作。

在 S3 Glacier 中，主要资源为文件库。S3 Glacier 仅支持文件库级别的策略。也就是说，在 IAM 策略中，您指定的`Resource`值可以是特定文件库或特定 AWS 区域中的一组文件库。S3 Glacier 不支持档案级权限。

对于所有 S3 Glacier 操作，`Resource` 指定要授予其权限的文件库。这些资源具有与之关联的唯一 Amazon 资源名称 (ARNs)，如下表所示，您可以在 ARN 中使用通配符 (*) 来匹配以相同前缀开头的文件库名称。

S3 Glacier 提供一组操作用来处理 S3 Glacier 资源。有关可用操作的信息，请参阅 Amazon S3 Glacier 的 API 参考。

某些 S3 Glacier API 操作支持多个资源。例如，`glacier:AddTagsToVault` 访问 examplevault1 和 examplevault2，因此主体必须具有访问这两个资源的权限。要在单个语句中指定多个资源，请 ARNs 用逗号分隔。

```
        "Resource": [
        ""arn:aws:glacier:us-west-2:123456789012:vaults/examplevault1",",
        ""arn:aws:glacier:us-west-2:123456789012:vaults/examplevault2","
        ]
```

S3 Glacier 的策略条件键

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

有关 S3 Glacier 条件键的列表，请参阅《服务授权参考》中的 Amazon S3 Glacier 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon S3 Glacier 定义的操作。

有关使用 Glacier 特定的条件键的示例，请参阅文件库锁定策略。

ACLs 在 S3 冰川中

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

ABAC 和 S3 Glacier

支持 ABAC（策略中的标签）：否

基于属性的访问控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在中 AWS，这些属性称为标签。您可以向 IAM 实体（用户或角色）和许多 AWS 资源附加标签。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 `aws:ResourceTag/`*`key-name`*、`aws:RequestTag/`*`key-name`* 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时凭证用于 S3 Glacier

支持临时凭证：是

当你使用临时凭证登录时，有些 AWS 服务 不起作用。有关更多信息，包括哪些 AWS 服务 适用于临时证书，请参阅 IAM 用户指南中的AWS 服务 与 IA M 配合使用的信息。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的从用户切换到 IAM 角色（控制台）。

您可以使用 AWS CLI 或 AWS API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 AWS。 AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

S3 Glacier 的跨服务主体权限

支持转发访问会话（FAS）：否

当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两项操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

S3 Glacier 的服务角色

支持服务角色：否

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会破坏 S3 Glacier 的功能。仅当 S3 Glacier 提供相关指导时才编辑服务角色。

S3 Glacier 的服务相关角色

支持服务相关角色：否

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

适用于 Amazon S3 Glacier 的基于身份的策略示例

默认情况下，用户和角色没有创建或修改 S3 Glacier 资源的权限。他们也无法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 执行任务。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略（控制台）。

有关 S3 Glacier 定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》中的 Amazon S3 Glacier 的操作、资源和条件密钥。 ARNs

以下是一个策略示例，该策略使用标识该地区所有文件库的 Amazon 资源名称 (`glacier:CreateVaultARNglacier:ListVaults`) 授予对资源执行三个 S3 Glacier 文件库相关操作（、`glacier:DescribeVault`和）的权限。`us-west-2` AWS ARNs 唯一标识 AWS 资源。有关与 S3 Glacier 配合 ARNs 使用的更多信息，请参阅S3 Glacier 策略资源。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
         "glacier:CreateVault",
         "glacier:DescribeVault",
         "glacier:ListVaults"
         ],
```

```
                "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/*"
            }
        ]
    }
```

此策略授予在 us-west-2 区域中创建、列出和获取文件库描述的权限。ARN 结尾处的通配符 (*) 表示此语句可匹配任何文件库名称。

> **Important**
>
> 在授予使用 glacier:CreateVault 操作创建文件库的权限时，您必须指定通配符 (*)，因为您在创建文件库之前不知道文件库的名称。

主题

- 策略最佳实践
- 使用 S3 Glacier 控制台
- 允许用户查看他们自己的权限
- 客户托管策略示例

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 S3 Glacier 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略或工作职能的AWS 托管式策略。
- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。

- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 S3 Glacier 控制台

要访问 Amazon S3 Glacier 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您的 AWS 账户中的 S3 Glacier 资源的详细信息。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

S3 Glacier 控制台为您提供了一个创建和管理 S3 Glacier 文件库的集成环境。至少要向您创建的 IAM 身份授予执行 `glacier:ListVaults` 操作的权限才能查看 S3 Glacier 控制台，如以下示例所示。

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

AWS 通过提供由创建和管理的独立 IAM 策略来解决许多常见用例 AWS。托管策略可针对常见使用案例授予必要权限，因此，您无需自行调查具体需要哪些权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

以下 AWS 托管策略仅适用于 S3 Glacier，您可以将其附加到账户中的用户：

- AmazonGlacierReadOnlyAccess— 通过授予对 S3 Glacier 的只读访问权限 AWS Management Console。
- AmazonGlacierFullAccess— 授予通过 S3 Glacier 的完全访问权限 AWS Management Console。

此外，您还可以创建自定义 IAM 策略，以授予 S3 Glacier API 操作和资源的相关权限。您可以将这些自定义策略附加到您为 S3 Glacier 文件库创建的自定义 IAM 角色。

下一节中讨论的两个 S3 Glacier AWS 托管策略都为授予权限`glacier:ListVaults`。

有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "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": "*"
        }
    ]
}
```

客户托管策略示例

本节的用户策略示例介绍如何授予各 S3 Glacier 操作的权限。当你使用 S3 Glacier REST API、Amazon SDKs AWS CLI、或 S3 Glacier 管理控制台（如果适用）时，这些策略会起作用。

> **Note**
>
> 所有示例都使用美国西部（俄勒冈）区域 (`us-west-2`)，并包含虚构账户。 IDs

示例

- 示例 1：允许用户从文件库下载存档
- 示例 2：允许用户创建文件库和配置通知
- 示例 3：允许用户将存档上传到特定文件库
- 示例 4：授予用户对特定文件库的完全权限

示例 1：允许用户从文件库下载存档

要下载存档，您首先要启动作业以检索存档。在检索作业完成后，您可以下载数据。以下示例策略授予执行 `glacier:InitiateJob` 操作以启动作业（这将允许用户从文件库中检索存档或文件库清单）的权限和执行 `glacier:GetJobOutput` 操作以下载检索到的数据的权限。此策略还授予执行 `glacier:DescribeJob` 操作的权限，以便用户能够获取作业状态。有关更多信息，请参阅 启动作业（POST 作业）。

此策略授予对名为 `examplevault` 的文件库的这些权限。您可以从 Amazon S3 Glacier 控制台获取文件库 ARN，也可以通过调用描述文件库（获取文件库）或列出文件库（获取文件库） API 操作以编程方式获取文件库 ARN。

```
{
                "Version":"2012-10-17",
                "Statement": [
                   {
                      "Effect": "Allow",
                      "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/
examplevault",
                      "Action":["glacier:InitiateJob",
                               "glacier:GetJobOutput",
                               "glacier:DescribeJob"]
                   }
                ]
                }
```

示例 2：允许用户创建文件库和配置通知

以下示例策略授予在 `Resource` 元素所指定的 us-west-2 区域中创建文件库以及配置通知的权限。有关使用通知的更多信息，请参阅在 Amazon S3 Glacier 中配置文件库通知。该政策还授予 AWS 在该地区列出文件库并获取特定文件库描述的权限。

> **Important**
>
> 在授予使用 `glacier:CreateVault` 操作创建文件库的权限时，您必须在 `Resource` 值中指定通配符 (*)，因为您在创建文件库之前不知道文件库的名称。

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

示例 3：允许用户将存档上传到特定文件库

以下示例策略授予将档案上传到 us-west-2 区域中的特定文件库的权限。借助这些权限，用户可以使用上传档案（发布档案）API 操作一次性上传整个存档，或者使用启动分段上传（发布分段上传）API 操作分段上传。

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

示例 4：授予用户对特定文件库的完全权限

以下示例策略授予对名为 `examplevault` 的文件库执行所有 S3 Glacier 操作的权限。

```
{
            "Version":"2012-10-17",
            "Statement": [
               {
                  "Effect": "Allow",
                  "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/
examplevault",
                  "Action":["glacier:*"]
               }
            ]
            }
```

适用于 Amazon S3 Glacier 的基于资源的策略示例

S3 Glacier 文件库可以有一个文件库访问策略和一个文件库锁定策略与之关联。Amazon S3 Glacier 文件库访问策略是一种基于资源的策略，可用于管理对文件库的权限。文件库锁定策略 是可锁定的文件库访问策略。在锁定文件库锁定策略后，无法更改策略。可以使用文件库锁定策略来实施合规性控制。

主题

- 库访问策略
- 文件库锁定策略

库访问策略

Amazon S3 Glacier 文件库访问策略是一种基于资源的策略，可用于管理对文件库的权限。

您可以为每个文件库创建一个文件库访问策略来管理权限。您可以随时修改文件库访问策略中的权限。S3 Glacier 还支持对每个文件库设置文件库锁定策略，文件库被锁定后即无法更改。有关使用文件库锁定策略的更多信息，请参阅文件库锁定策略。

示例

- 示例 1：授予特定的 Amazon S3 Glacier 操作的跨账户权限
- 示例 2：授予 MFA 删除操作的跨账户权限

示例 1：授予特定的 Amazon S3 Glacier 操作的跨账户权限

以下示例策略向两个 AWS 账户 授予对名为 `examplevault` 的文件库执行一组 S3 Glacier 操作的跨账户权限。

> **Note**
>
> 拥有该文件库的账户需要支付与该文件库关联的所有费用。由允许的外部账户产生的所有请求、数据传输和检索费用均由拥有该文件库的账户支付。

```
{
   "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"

                       ]
                    }
                 ]
              }
```

示例 2：授予 MFA 删除操作的跨账户权限

您可以使用多重身份验证 (MFA) 来保护您的 S3 Glacier 资源。为了提供额外的安全级别，MFA 要求用户通过提供有效的 MFA 代码来证明其实际拥有 MFA 设备。有关配置 MFA 访问权限的更多信息，请参阅《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

示例策略向 AWS 账户 具有临时证书的用户授予从名为 examplevault 的文件库中删除档案的权限，前提是该请求已使用 MFA 设备进行身份验证。此策略使用 `aws:MultiFactorAuthPresent` 条件键指定这一附加要求。有关更多信息，请参阅《IAM 用户指南》中的可用的条件键。

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

文件库锁定策略

可以为 Amazon S3 Glacier（S3 Glacier）文件库附加一个基于资源的文件库访问策略和一个文件库锁定策略。文件库锁定策略 是您可以锁定的文件库访问策略。使用文件库锁定策略可以帮助您强制执行监管和合规性要求。Amazon S3 Glacier 提供了一组用于管理文件库锁策略的 API 操作，请参阅使用 S3 Glacier API 锁定文件库。

我们举例来说明文件库锁定策略，假设您需要先将存档保留一年，然后才能删除存档。要实施此要求，您可以创建一个文件库锁定策略，以便在存档保留时间达到 1 年之前拒绝用户删除存档的权限。在锁定一个策略之前，可以先测试该策略。锁定策略后，策略将是不可变的。有关锁定过程的更多信息，请参阅 文件库锁定策略。如果您要管理可更改的其他用户权限，可以使用文件库访问策略（请参阅 库访问策略）。

您可以使用 S3 Glacier API SDKs AWS CLI、Amazon 或 S3 Glacier 控制台来创建和管理文件库锁定策略。有关基于文件库资源的策略所允许的 S3 Glacier 操作的列表，请参阅API 权限参考。

示例

- 示例 1：拒绝针对保留时间不到 365 天的存档的删除权限
- 示例 2：根据标签来拒绝删除权限

示例 1：拒绝针对保留时间不到 365 天的存档的删除权限

假定您需要符合法规要求，只能删除保留时间达到 1 年的存档。可通过实施以下文件库锁定策略来实施此要求。如果要删除的存档的保留时间不到 1 年，则该策略将拒绝对 examplevault 文件库执行 `glacier:DeleteArchive` 操作。该策略使用特定于 S3 Glacier 的条件键 `ArchiveAgeInDays` 来实施 1 年保留要求。

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

示例 2：根据标签来拒绝删除权限

假定您实施了一个基于时间的保留规则，即允许在存档保留时间不到 1 年的情况下删除存档。同时，假定您需要对您的存档实施法定保留策略，以在法律调查期间无限期地禁止删除或修改操作。在这种情况下，法定保留优先于文件库锁定策略中指定的基于时间的保留规则。

为了实施这两个规则，以下示例策略包含两条语句：

- 第一条语句拒绝任何人删除存档（锁定文件库）。可通过使用 `LegalHold` 标签执行此锁定。
- 第二条语句授予在存档保留时间不到 365 天的情况下删除存档的权限。但是，如果满足第一条语句中的条件，即使在存档保留时间不到 365 天的情况下，任何人也无法删除存档。

```
{
"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 身份和访问管理问题排查

使用以下信息可帮助您诊断和修复在使用 S3 Glacier 和 IAM 时可能遇到的常见问题。

主题

- 我无权在 S3 Glacier 中执行操作
- 我无权执行 iam：PassRole
- 我想允许我以外的人访问我 AWS 账户 的 S3 Glacier 资源

我无权在 S3 Glacier 中执行操作

如果您收到错误提示，指明您无权执行某个操作，则必须更新策略以允许执行该操作。

当 `mateojackson` IAM 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `glacier:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 glacier:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 `mateojackson` 用户的策略，以允许使用 `glacier:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam：PassRole

如果收到错误，则表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 S3 Glacier。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 S3 Glacier 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人访问我 AWS 账户 的 S3 Glacier 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 S3 Glacier 是否支持这些功能，请参阅Amazon S3 Glacier 如何与 IAM 配合使用。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

API 权限参考

在设置Amazon S3 Glacier 如何与 IAM 配合使用和编写可挂载到 IAM 身份的权限策略（基于身份的策略）或可挂载到资源的权限策略（基于资源的权限策略）时，可以使用下表作为参考。该包括每个 S3 Glacier API 操作、您可以为其授予执行该操作的权限的相应操作以及您可以为其授予权限的 AWS 资源。

您可以在策略的 `Action` 元素中指定这些操作，并在策略的 `Resource` 元素中指定资源值。另外，您可以使用 IAM 策略语言 `Condition` 元素指定策略将生效的时间。

要指定操作，请在 API 操作名称之前使用 `glacier:` 前缀（例如，`glacier:CreateVault`）。对于大多数 S3 Glacier 操作而言，`Resource` 是要授予其权限的文件库。您可以通过使用文件库 ARN，将文件库指定为 `Resource` 值。要表示条件，您可以使用预定义的条件键。有关更多信息，请参阅 S3 Glacier 内基于资源的策略。

下表列出了可用于基于身份的策略和基于资源的策略的操作。

> **Note**
> 一些操作只能用于基于身份的策略。这些操作在第一列的 API 操作名称后面用星号 (*) 标记。

S3 Glacier API 和所需的操作权限

中止分段上传（删除上传 ID）

所需权限（API 操作）：`glacier:AbortMultipartUpload`

资源：`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 条件键：

中止文件库锁定（DELETE 锁定策略）

所需权限（API 操作）：`glacier:AbortVaultLock`

资源：

S3 Glacier 条件键：

向文件库添加标签（POST 标签添加）

所需权限（API 操作）：`glacier:AddTagsToVault`

资源：`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 条件键：`glacier:ResourceTag/TagKey`

完成分段上传（发布上传 ID）

所需权限（API 操作）：`glacier:CompleteMultipartUpload`

资源：`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 条件键：`glacier:ResourceTag/TagKey`

完成文件库锁定 (POST lockId)

所需权限（API 操作）：glacier:CompleteVaultLock

资源：

S3 Glacier 条件键：glacier:ResourceTag/*TagKey*

创建文件库（设置文件库）*

所需权限（API 操作）：glacier:CreateVault

资源：

S3 Glacier 条件键：

删除档案

所需权限（API 操作）：glacier:DeleteArchive

资源：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 条件键：glacier:ArchiveAgeInDays、glacier:ResourceTag/*TagKey*

删除文件库

所需权限（API 操作）：glacier:DeleteVault

资源：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 条件键：glacier:ResourceTag/*TagKey*

删除文件库访问策略 (DELETE access-policy)

所需权限（API 操作）：glacier:DeleteVaultAccessPolicy

资源：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 条件键：glacier:ResourceTag/*TagKey*

删除文件库通知（删除通知配置）

所需权限（API 操作）：glacier:DeleteVaultNotifications

资源：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 条件键：glacier:ResourceTag/*TagKey*

描述任务（获取任务 ID）

所需权限（API 操作）：glacier:DescribeJob

资源：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 条件键：

描述文件库（获取文件库）

所需权限（API 操作）：glacier:DescribeVault

资源：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 条件键：

获取数据检索策略（GET 策略）*

所需权限（API 操作）：glacier:GetDataRetrievalPolicy

资源：arn:aws:glacier:*region*:*account-id*:policies/retrieval-limit-policy

S3 Glacier 条件键：

获取任务输出（获取输出）

所需权限（API 操作）：glacier:GetJobOutput

资源：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 条件键：

获取文件库访问策略 (GET access-policy)

所需权限（API 操作）：glacier:GetVaultAccessPolicy

资源：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 条件键：

获取文件库锁定（GET 锁定策略）

所需权限（API 操作）：glacier:GetVaultLock

资源：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 条件键：

获取文件库通知（获取通知配置）

所需权限（API 操作）：glacier:GetVaultNotifications

资源：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 条件键：

启动作业（POST 作业）

所需权限（API 操作）：glacier:InitiateJob

资源：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 条件键：glacier:ArchiveAgeInDays、glacier:ResourceTag/*TagKey*

启动分段上传（发布分段上传）

所需权限（API 操作）：glacier:InitiateMultipartUpload

资源：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 条件键：glacier:ResourceTag/*TagKey*

启动文件库锁定（POST 锁定策略）

所需权限（API 操作）：glacier:InitiateVaultLock

资源：

S3 Glacier 条件键：glacier:ResourceTag/*TagKey*

列出任务（获取任务）

所需权限（API 操作）：glacier:ListJobs

资源：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 条件键：

列出分段上传（获取分段上传）

所需权限（API 操作）：glacier:ListMultipartUploads

资源：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 条件键：

列出段（获取上传 ID）

所需权限（API 操作）：glacier:ListParts

资源：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 条件键：

列出文件库的标签（GET 标签）

所需权限（API 操作）：glacier:ListTagsForVault

资源：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 条件键：

列出文件库（获取文件库）

所需权限（API 操作）：glacier:ListVaults

资源：

S3 Glacier 条件键：

从文件库删除标签（POST 标签删除）

所需权限（API 操作）：glacier:RemoveTagsFromVault

资源：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 条件键：glacier:ResourceTag/*TagKey*

设置数据检索策略（PUT 策略）*

所需权限（API 操作）：glacier:SetDataRetrievalPolicy

资源：arn:aws:glacier:*region*:*account-id*:policies/retrieval-limit-policy

S3 Glacier 条件键：

设置文件库访问策略 (PUT access-policy)

所需权限（API 操作）：glacier:SetVaultAccessPolicy

资源：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 条件键：glacier:ResourceTag/*TagKey*

设置文件库通知配置（设置通知配置）

所需权限（API 操作）：glacier:SetVaultNotifications

资源：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 条件键：glacier:ResourceTag/*TagKey*

上传档案（发布档案）

所需权限（API 操作）：glacier:UploadArchive

资源：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 条件键：glacier:ResourceTag/*TagKey*

上传段（设置上传 ID）

所需权限（API 操作）：glacier:UploadMultipartPart

资源：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 条件键：glacier:ResourceTag/*TagKey*

Amazon S3 Glacier 中的日志记录和监控

监控是维护 Amazon S3 Glacier（S3 Glacier）和您的 AWS 解决方案的可靠性、可用性和性能的重要组成部分。您应该从 AWS 解决方案的所有部分收集监控数据，以便在出现故障时可以更轻松地识别和调试故障来源。 AWS 提供了以下用于监控您的 S3 Glacier 资源和响应潜在事件的工具：

亚马逊 CloudWatch 警报

通过 Amazon S3 使用 S3 Glacier 时，您可以使用亚马逊 CloudWatch 警报在您指定的时间段内监控单个指标。如果指标超过给定阈值，则会向 Amazon SNS 主题或 AWS Auto Scaling 政策发送通知。 CloudWatch 警报不会调用操作，因为它们处于特定状态。而是必须在状态已改变并在指定的若干个时间段内保持不变后才调用。有关更多信息，请参阅使用 Amazon 监控指标 CloudWatch。

AWS CloudTrail 日志

CloudTrail 提供用户、角色或 AWS 服务在 S3 Glacier 中执行的操作的记录。 CloudTrail 将 S3 Glacier 的所有 API 调用捕获为事件，包括来自 S3 Glacier 控制台的调用和对 S3 Glacier 的代码调用 APIs。有关更多信息，请参阅 使用记录亚马逊 S3 Glacier API 调用 AWS CloudTrail。

AWS Trusted Advisor

Trusted Advisor 借鉴了从为成千上万的 AWS 客户提供服务中学到的最佳实践。 Trusted Advisor 检查您的 AWS 环境，然后在有机会节省资金、提高系统可用性和性能或帮助填补安全漏洞时提出建议。所有 AWS 客户均可使用五张 Trusted Advisor 支票。拥有商业或企业支持计划的客户可以查看所有 Trusted Advisor 支票。

有关更多信息，请参阅《支持 用户指南》中的 AWS Trusted Advisor。

Amazon S3 Glacier 的合规性验证

Amazon S3 Glacier（S3 Glacier）的安全与合规性由第三方审计机构作为多项 AWS 合规计划的一部分进行评估，包括：

- 系统和组织控制 (SOC)
- 支付卡行业数据安全标准 (PCI DSS)
- 联邦风险与授权管理项目 (FedRAMP)
- 健康保险流通与责任法案 (HIPAA)

AWS 在 “按合规计划划分的范围内的 AWS 服务” 中提供了经常更新的特定合规计划范围内的AWS 服务列表。

您可以使用第三方审计报告进行下载 AWS Artifact。有关更多信息，请参阅《AWS Artifact 用户指南》中的 “在 A AWS rtifact 中下载报告”。

有关 AWS 合规计划的更多信息，请参阅AWS 合规计划。

您在使用 S3 Glacier 时的合规性责任由您数据的敏感性、您组织的合规性目标以及适用的法律法规决定。如果您对 S3 Glacier 的使用需遵守 HIPAA、PCI 或 FedRAMP 等标准， AWS 提供了以下实用资源：

- 通过 S3 Glacier 文件库锁定，您可轻松利用文件库锁定策略对单独的 S3 Glacier 文件库进行部署并实施合规性控制。您可以在一个文件库锁定策略中指定类似“一次写入，多次读取”(WORM) 这样的控制，并且可以锁定该策略以防止将来进行编辑。策略在锁定后不能再更改。文件库锁定策略可以帮助您遵守监管框架，例如 SEC17a -4 和 HIPAA。
- 《安全与合规性快速入门指南》讨论了架构注意事项和在上部署以安全性和合规性为重点的基准环境的步骤。 AWS
- HIPAA 安全与合规架构概述了公司 AWS 如何帮助他们满足 HIPAA 要求。
- W@@ el AWS l-Architected 工具AWS （WA 工具）是一项云端服务，可为您提供一致的流程，供您使用最佳实践来审查和衡量架构 AWS 。 AWS WA 工具为提高工作负载的可靠性、安全性、效率和成本效益提供了建议。
- AWS 合规资源提供了几种不同的工作簿和指南，可能适用于您的行业和所在地。
- AWS Config 可帮助您评估您的资源配置对内部实践、行业指南和法规的遵守情况。
- AWS Sec@@ urity Hub 为您提供内部安全状态的全面视图， AWS 并帮助您检查自己是否符合安全行业标准和最佳实践。

Amazon S3 Glacier 中的故障恢复能力

AWS 全球基础设施是围绕区域和可用区构建的。 AWS 区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。这些可用区为您提供了高效的方法来设计和操作应用程序和数据库。与传统的单个数据中心基础设施或多个数据中心基础设施相比，它们具有更高的可用性、容错性和可扩展性。S3 Glacier 以冗余方式将数据存储在跨越至少三个可用区的多台设备中。为了提高持久性，在确认成功上传 AZs 之前，S3 Glacier 会将您的数据同步存储在多个位置。

有关 AWS 区域和可用区的更多信息，请参阅AWS 全球基础设施。

Amazon S3 Glacier 中的基础设施安全性

作为一项托管服务，Amazon S3 Glacier（S3 Glacier）受亚马逊网络服务：安全流程概述中描述的 AWS 全球网络安全程序的保护。

通过网络访问 S3 Glacier 是通过 AWS 发布的 APIs。客户端必须支持传输层安全性（TLS）1.2。我们建议使用 TLS 1.3 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Diffie-Hellman Ephemeral (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。此外，必须使用与 IAM 主体关联的访问密钥 ID 和秘密访问密钥对请求进行签名，也可以使用AWS Security Token Service (AWS STS) 生成临时安全证书来对请求进行签名。

VPC 端点

通过虚拟私有云 (VPC) 终端节点，您 AWS PrivateLink 无需互联网网关、NAT 设备、VPN 连接或连接，即可将您的 VPC 与支持的 AWS 服务和由其提供支持的 VPC 终端节点服务进行私密 AWS Direct Connect 连接。虽然 S3 Glacier 不直接支持 VPC 端点，但如果您将 S3 Glacier 作为与 Amazon S3 集成的存储层进行访问，则可以利用 Amazon S3 VPC 端点。

有关 Amazon S3 生命周期配置以及将对象转换为 S3 Glacier 存储类的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的对象生命周期管理和转换对象。有关 VPC 端点的更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 端点。

S3 Glacier 数据检索策略

借助 Amazon S3 Glacier 数据检索策略，您可以轻松设置数据检索配额并管理每个配额 AWS 账户 中的数据检索活动 AWS 区域。有关 S3 Glacier 数据检索费用的更多信息，请参阅 S3 Glacier 定价。

> **Important**
>
> 数据检索策略仅适用于标准检索并管理直接对 S3 Glacier 发出的检索请求。
> 有关 S3 Glacier 存储类的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的用于归档对象的存储类和转换对象。

主题

- 选择 S3 Glacier 数据检索策略
- 使用 S3 Glacier 控制台设置数据检索策略
- 使用 Amazon S3 Glacier API 设置数据检索策略

选择 S3 Glacier 数据检索策略

您可以从三种 S3 Glacier 数据检索策略中进行选择：No Retrieval Limit (无检索限制)、Free Tier Only (仅免费套餐) 和 Max Retrieval Rate (最大检索速率)。

为检索使用的默认数据检索策略是“No Retrieval Limit (无检索限制)”。如果您使用“No Retrieval Limit (无检索限制)”策略，则不设置检索配额，接受所有有效的数据检索请求。

通过使用仅限免费套餐的政策，您可以将取回的次数控制在每日 AWS 免费套餐限额之内，而不会产生任何数据检索费用。如果您想要检索的数据量超过 AWS 免费套餐限额，则可以使用最高检索率策略来设置检 bytes-per-hour索率配额。“最大检索率” 策略可确保您的账户中所有检索任务的峰值检索率 AWS 区域 不会超过您设置的 bytes-per-hour配额。

对于“Free Tier Only (仅免费套餐)”和“Max Retrieval Rate (最大检索速率)”策略，超出您所指定的检索配额的数据检索请求都不会被接受。如果您使用“Free Tier Only (仅免费套餐)”策略，则 S3 Glacier 将同步拒绝超出 AWS 免费套餐限额的检索请求。如果您使用最大检索率策略，S3 Glacier 会拒绝导致正在进行的任务的峰值检索率超过该策略设置的bytes-per-hour 配额的检索请求。这些策略可帮助您简化数据检索费用管理。

以下是关于数据检索策略的一些有用事实：

- 设置数据检索策略并不会改变使用标准检索从 S3 Glacier 中检索数据所需要的时长：3 到 5 小时。
- 设置新的数据检索策略并不影响之前接受且已在进行的检索作业。
- 如果某个检索任务请求由于数据检索策略而被拒绝，我们不会就该任务或请求向您收费。
- 您可以为每个策略设置一个数据检索策略 AWS 区域，该策略将管理您账户 AWS 区域 下的所有数据检索活动。数据检索策略是特定的， AWS 区域 因为数据检索成本各不相同 AWS 区域。有关更多信息，请参阅 Amazon S3 Glacier 定价。

“Free Tier Only (仅免费套餐)”策略

您可以将数据检索政策设置为 “仅限免费套餐”，以确保您的检索始终保持在 AWS 免费套餐限额之内，这样您就不会产生数据检索费用。如果某个检索请求被拒绝，您会收到一条错误消息，指出请求已被当前数据检索策略拒绝。

您可以基于区域将数据检索策略设置为“Free Tier Only (仅免费套餐)”。一旦设置好策略，您每天能够检索的数据量就会限制在针对该 AWS 区域按比例计算的每日 AWS 免费检索限额以内。您也不会产生数据检索费用。

在您产生数据检索费用后一个月内，您还可以切换到“Free Tier Only (仅免费套餐)”策略。在这种情况下，“Free Tier Only (仅免费套餐)”策略将对新的检索请求生效，但不会影响过去的请求。您将需要支付之前产生的费用。

“Max Retrieval Rate (最大检索速率)”策略

您可以将数据检索策略设置为 “最大检索速率”，通过指定具有 bytes-per-hour最大值的数据检索配额来控制峰值检索速率。当您将数据检索策略设置为 “最大检索率” 时，如果新的检索请求会导致正在进行的任务的峰值检索率超过策略指定的 bytes-per-hour配额，则该请求将被拒绝。如果某个检索任务请求被拒绝，您将收到一条错误消息，指出请求已被当前数据检索策略拒绝。

将数据检索策略设置为 “最大检索率” 策略可能会影响您一天内可以使用的 AWS 免费套餐限额。例如，假定您将最大检索速率设置为每小时 1 MB。这低于 AWS 免费套餐政策费率。为确保充分利用每日 AWS 免费套餐限额，您可以先将政策设置为 “仅限免费套餐”，然后根据需要稍后切换到最高检索率政策。有关如何计算取回限额的更多信息，请访问 Amazon S3 Glacier FAQs。

“No Retrieval Limit (无检索限制)”策略

如果您的数据检索策略设置为“No Retrieval Limit (无检索限制)”，则将接受所有有效的数据检索请求，且您的数据检索费用将根据您的使用量变化。

使用 S3 Glacier 控制台设置数据检索策略

使用 Amazon S3 Glacier 控制台创建数据检索策略

1. 登录 AWS Management Console 并在https://console.aws.amazon.com/glacier/家中打开 S3 Glacier 控制台。
2. 在 “选择区域” 下， AWS 区域 从下拉菜单中选择一个。您可以为每个策略配置数据检索策略 AWS 区域。
3. 在左侧的导航窗格中，选择数据检索设置。
4. 选择编辑。此时将出现编辑数据检索策略页面。
5. 在数据检索策略下，选择一个策略。

   您可以从三种数据检索策略中选择一种：No Retrieval Limit (无检索限制)、Free Tier Only (仅免费套餐)或 Specify a max retrieval rate (指定最大检索速率)。

   - 如果选择 无检索限制，则接受所有有效的数据检索请求。
   - 如果您仅选择免费套餐，则不接受超过 AWS 免费套餐的数据检索请求。
   - 如果选择指定最大检索率，当数据检索请求会导致正在进行的任务的峰值检索速率超过您指定的最大检索速率，则数据检索请求将被拒绝。您必须在最大检索速率下的每小时 GB 数框中指定每小时千兆字节 (GB) 的值。在每小时 GB 数中输入值以后，控制台会为您计算估计的费用。
6. 选择保存更改。

使用 Amazon S3 Glacier API 设置数据检索策略

您可以使用 Amazon S3 Glacier REST API 或使用来查看和设置数据检索策略 AWS SDKs。

使用 Amazon S3 Glacier REST API 设置数据检索策略

您可以使用 Amazon S3 Glacier REST API 查看和设置数据检索策略。您可以使用获取数据检索策略（GET 策略）操作查看现有数据检索策略。您可以使用设置数据检索策略（PUT 策略）操作设置数据检索策略。

使用 PUT 策略操作时，您可通过将 JSON `Strategy` 字段值设置为 `BytesPerHour`、`FreeTier` 或 None 来选择数据检索策略类型。`BytesPerHour` 等同于在控制台中选择指定最大检索速率，`FreeTier` 等同于选择仅免费套餐，None 等同于选择无检索限制。

当您使用启动作业（POST 作业）操作启动数据检索任务，而该任务将超出数据检索策略中设置的最大检索速率时，`Initiate Job`操作将停止并引发异常。

使用 AWS SDKs 来设置数据检索策略

AWS 允许您 SDKs 为 Amazon S3 Glacier 开发应用程序。它们 SDKs 提供了映射到底层 REST API 的库，并提供了使您能够轻松构造请求和处理响应的对象。有关更多信息，请参阅 AWS SDKs 与 Amazon S3 Glacier 搭配使用。

标记 Amazon S3 Glacier 资源

标签是您分配给 AWS 资源的标签。每个标签都由键 和值组成，这两个参数都由您定义。您可将您定义的标签分配给 Amazon S3 Glacier（S3 Glacier）文件库资源。使用标签是一种管理 AWS 资源和整理数据（包括账单数据）的简单而强大的方法。

主题

- 标签基本知识
- 标签限制
- 使用标签跟踪成本
- 使用标签管理访问控制
- 相关部分

标签基本知识

您可以使用 S3 Glacier 控制台、 AWS Command Line Interface (AWS CLI) 或 S3 Glacier API 来完成以下任务：

- 向文件库添加标签
- 列出文件库的标签
- 从文件库中删除标签

有关如何添加、列出和删除标签的信息，请参阅 标记 S3 Glacier 文件库。

您可使用标签对您的文件库进行分类。例如，您可以按用途、所有者或环境对文件库进行分类。由于您定义每个标签的键和值，因此您可以创建一组自定义类别来满足您的特定需求。例如，您可定义一组标签，帮助您按文件库的所有者和用途跟踪文件库。以下是一些标签的示例：

- 所有者：名称
- 用途：视频存档
- 环境：生产

标签限制

基本标签限制如下：

- 一个资源（文件库）的最大标签数为 50。
- 标签键和值区分大小写。

标签键限制如下：

- 在文件库的一组标签内，每个标签键必须是唯一的。如果您添加的标签具有已使用的键，则您的新标签将覆盖现有键值对。
- 标签密钥不能以开头，`aws:`因为此前缀已保留供使用 AWS。 AWS 可以代表您创建以此前缀开头的标签，但您无法对其进行编辑或删除。
- 标签键的长度必须介于 1 和 128 个 Unicode 字符之间。
- 标签键必须包含以下字符：Unicode 字母、数字、空格和以下特殊字符：`_ . / = + - @`。

标签值限制如下：

- 标签值的长度必须介于 0 和 255 个 Unicode 字符之间。
- 标签值可以为空。另外，它们必须包含以下字符：Unicode 字母、数字、空格和以下任意特殊字符：`_ . / = + - @`。

使用标签跟踪成本

您可以使用标签对 AWS 费用进行分类和跟踪。当您将标签应用于任何 AWS 资源（包括文件库）时，您的 AWS 成本分配报告将包括按标签汇总的使用量和成本。您可以应用代表业务类别（如成本中心、应用程序名称或所有者）的标签，以便在多项服务中组织您的成本。有关更多信息，请参阅 AWS Billing 用户指南中的对自定义账单报告使用成本分配标签。

使用标签管理访问控制

您可使用标签作为访问策略语句中的条件。例如，您可设置一个法定保留标签并将其作为数据保留策略中的一个条件包含，该策略声明“如果法定保留标签值设置为 `True`，则将拒绝每个人的存档删除操作。”您可部署数据保留策略并在正常条件下将法定保留标签设置为 `False`。如果必须保留您的数据来协助调查，则可通过将标签值设置为 `True` 并在稍后以类似方式解除保留来轻松打开此法定保留。有关更多信息，请参阅《IAM 用户指南》中的使用标签控制访问。

相关部分

- 标记 S3 Glacier 文件库

使用记录亚马逊 S3 Glacier API 调用 AWS CloudTrail

Amazon S3 Glacier（S3 Glacier）与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在 S3 Glacier 中采取的操作的记录。 CloudTrail 将 S3 Glacier 的所有 API 调用捕获为事件，包括来自 S3 Glacier 控制台的调用和对 S3 Glacier 的代码调用 APIs。如果您创建了跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括 S3 Glacier 的事件。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台的 Event history（事件历史记录）中查看最新事件。使用收集的信息 CloudTrail，您可以确定向 S3 Glacier 发出的请求、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅AWS CloudTrail 用户指南。

中的亚马逊 S3 Glacier 信息 CloudTrail

CloudTrail 在您创建账户 AWS 账户 时已在您的账户上启用。当 S3 Glacier 中发生活动时，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在中查看、搜索和下载最近发生的事件 AWS 账户。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您的事件 AWS 账户，包括 S3 Glacier 的事件，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，当您在控制台中创建跟踪时，该跟踪将应用于所有 AWS 区域。跟踪记录 AWS 分区中所有 AWS 区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

所有 S3 Glacier 操作均由记录 CloudTrail 并记录在中Amazon S3 Glacier 的 API 参考。例如，对创建文件库（设置文件库）删除文件库、和列出文件库（获取文件库）操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是其他凭证发出的。

- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 Amazon S3 Glacier 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序出现。

以下示例显示了一个演示创建文件库（设置文件库）、删除文件库列出文件库（获取文件库）、和描述文件库（获取文件库）操作的 CloudTrail 日志条目。

```
{
    "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 的 API 参考

Amazon S3 Glacier 支持一系列操作，特别是一组 RESTful API 调用，使您能够与服务进行交互。

您可以使用能够发送 HTTP 请求的任何编程库，将您的 REST 请求发送到 S3 Glacier。发送 REST 请求时，S3 Glacier 会要求您通过对请求签名来验证每个请求。此外，上传档案时，您还必须计算有效负载的校验和，并将它包括在您的请求中。有关更多信息，请参阅 对请求进行签名。

如果发生错误，您需要知道 S3 Glacier 在错误响应中发送的内容，以便进行处理。除了介绍 REST 操作以外，此部分还提供了所有的此类信息，以便您直接进行 REST API 调用。

您可以直接使用 REST API 调用，也可以使用提供包装器库 SDKs 的 Amazon。这些库会对您发送的每个请求签名，并对您请求中的有效负载计算校验和。因此，使用 Amazon SDKs 可以简化您的编码任务。本开发人员指南提供了使用 适用于 Java 的 AWS SDK 和.NET 进行基本 S3 Glacier 操作的工作示例。有关更多信息，请参阅AWS SDKs 与 Amazon S3 Glacier 搭配使用。

主题

- 常见请求标头
- 常用响应标头
- 对请求进行签名
- 计算校验和
- 错误响应
- 文件库操作
- 归档操作
- 分段上传操作
- 任务操作
- 在任务操作中使用的数据类型
- 数据检索操作

常见请求标头

Amazon S3 Glacier（S3 Glacier）REST 请求采用包含此请求基本说明的标头。下表描述了所有 S3 Glacier REST 请求均可使用的标头。

标头名称	描述	必填
Authorization	对请求签名所需的标头。S3 Glacier 需要签名版本 4。有关更多信息，请参阅 对请求进行签名。 类型：字符串	是
Content-Length	请求正文（不带标头）的长度。 类型：字符串 条件：只有 上传档案（发布档案） API 才需要。	条件
Date	可以用于创建 Authorization 标头中包含的签名的日期。如果要将 Date 标头用于签名，则必须使用 ISO 8601 基本格式来指定它。在这种情况下，不需要 x-amz-date 标头。请注意，存在 x-amz-date 时，它始终会覆盖 Date 标头的值。 如果 Date 标头不用于签名，则可以为 RFC 2616 第 3.3 部分指定的完整日期格式之一。例如，以下日期/时间 Wed, 10 Feb 2017 12:00:00 GMT 是用于 S3 Glacier 的有效日期/时间标头。 如果您要将 Date 标头用于签名，则它必须使用 ISO 8601 基本 YYYYMMDD'T'HHMMSS'Z' 格式。 类型：字符串 条件：如果指定了 Date，但它没有使用 ISO 8601 基本格式，则您还必须包括 x-amz-date 标头。如果使用 ISO 8601 基本格式指定了 Date，则它足够用于对请求签名，您无需 x-amz-date 标头。有关更多信息，请参阅《Amazon Web Services Glossary》中的“Handling Dates in Signature Version 4”。	条件

标头名称	描述	必填
Host	此标头指定您要将请求发送到的服务端点。该值必须采用 glacier.*region*.amazonaws.com “” 格式，其中替换*region*为 AWS 区域标识，例如us-west-2 。 类型：字符串	是
x-amz-content-sha256	使用上传档案（发布档案）或上传段（设置上传 ID）上传的整个有效载荷的计算 SHA256 校验和。虽然此标头与 x-amz-sha256-tree-hash 标头不同，但是，对于某些小型有效负载，值是相同的。如果需要 x-amz-content-sha256 ，则必须指定 x-amz-content-sha256 和 x-amz-sha256-tree-hash 。 类型：字符串 条件：流式处理 API、上传档案（发布档案）和 上传段（设置上传 ID）需要。	条件
x-amz-date	用于在 Authorization 标头中创建签名的日期。格式必须为使用 YYYYMMDD'T'HHMMSS'Z' 格式的 ISO 8601 基本格式。例如，以下日期/时间 20170210T120000Z 是用于 S3 Glacier 的有效 x-amz-date 。 类型：字符串 条件：x-amz-date 对所有请求而言是可选的；它可以用于覆盖对请求签名所使用的日期。如果使用 ISO 8601 基本格式指定了 Date 标头，则无需 x-amz-date 。存在 x-amz-date 时，它始终会覆盖 Date 标头的值。有关更多信息，请参阅《Amazon Web Services Glossary》中的“Handling Dates in Signature Version 4”。	条件

标头名称	描述	必填
x-amz-glacier-version	要使用的 S3 Glacier API 版本。当前版本为 2012-06-01 。 类型：字符串	是
x-amz-sha256-tree-hash	已上传档案 (上传档案（发布档案）) 或档案部分 () 的计算出的 SHA256 树形哈希校验和。上传段（设置上传 ID）有关计算此校验和的更多信息，请参阅“计算校验和”。 类型：字符串 默认值：无 条件：上传档案（发布档案） 和 上传段（设置上传 ID） 需要。	条件

常用响应标头

下表描述了大多数 API 响应中的常用响应标头。

名称	描述
Content-Length	响应正文的长度（以字节为单位）。 类型：字符串
Date	Amazon S3 Glacier（S3 Glacier）的响应日期和时间，例如 Wed, 10 Feb 2017 12:00:00 GMT。日期的格式必须是 RFC 2616 第 3.3 节中指定的完整日期格式之一。请注意，返回的 Date 可能与其他日期略有偏差，因此，例如，从 上传档案（发布档案） 请求返回的日期可能与文件库库存列表中的档案所显示的日期不匹配。 类型：字符串
x-amzn-RequestId	由 S3 Glacier 创建的唯一标识您请求的值。如果您在使用 S3 Glacier 时遇到问题， AWS 可以使用此值来解决问题。建议您记录这些值。

名称	描述
	类型：字符串
x-amz-sha256-tree-hash	档案或清单正文的 SHA256 树形哈希校验和。有关计算此校验和的更多信息，请参阅“计算校验和”。 类型：字符串

对请求进行签名

S3 Glacier 要求通过对请求进行签名，验证所发送的每个请求的身份。您使用加密哈希函数计算数字签名，从而对请求签名。加密哈西是根据输入内容返回唯一哈希值的函数。对哈希函数的输入内容包括您的请求文本和秘密访问密钥。哈希函数返回哈希值，您将该值包含在请求中，作为签名。该签名是您的请求的 `Authorization` 标头的一部分。

收到您的请求后，S3 Glacier 使用与您用于对该请求进行签名的相同哈希函数和输入重新计算签名。如果所得签名与该请求中的签名相匹配，则 S3 Glacier 处理该请求。否则，请求将被拒绝。

S3 Glacier 支持使用AWS 签名版本 4 进行身份验证。计算签名的过程可分为三个任务：

- 任务 1：创建规范请求

  将您的 HTTP 请求重新排列为规范格式。必须使用规范格式，因为 S3 Glacier 在重新计算签名以与您发送的签名进行比较时使用同一规范格式。
- 任务 2：创建待签字符串

  创建一个字符串，将该字符串用作您的加密哈希函数输入值中的一项。该字符串称为待签字符串，是哈希算法名称、请求日期、凭证范围字符串以及来自上一任务的规范化请求的结合。凭证范围字符串本身就是日期、 AWS 区域和服务信息的串联。
- 任务 3：创建签名

  使用加密哈希函数为您的请求创建签名，该函数接受两种输入字符串：待签字符串和派生密钥。派生密钥的计算方法是从您的私有访问密钥开始，然后使用凭证范围字符串创建一系列基于哈希的消息身份验证代码 () HMACs。请注意，此签名步骤中使用的哈希函数不是 S3 Glacier APIs 中用于上传数据的树形哈希算法。

主题

- 实例签名计算
- 为流式处理操作计算签名

实例签名计算

以下示例引导您了解为 创建文件库（设置文件库） 创建签名的详细信息。该示例可用作核查您的签名计算方法的参考。有关更多信息，请参阅 I AM 用户指南中的签署 AWS API 请求。

示例假定以下各项：

- 请求的时间戳为 `Fri, 25 May 2012 00:24:53 GMT`。
- 端点为美国东部（弗吉尼亚州北部）区域 `us-east-1`。

通用请求语法 (包括 JSON 正文) 为：

```
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
```

为任务 1：创建规范请求计算的规范请求格式为：

```
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
```

规范请求的最后一行是请求正文的哈希值。另外，请注意规范请求的第三行是空的。这是因为此 API 没有查询参数。

任务 2：创建待签字符串的待签字符串为：

```
AWS4-HMAC-SHA256
20120525T002453Z
20120525/us-east-1/glacier/aws4_request
5f1da1a2d0feb614dd03d71e87928b8e449ac87614479332aced3a701f916743
```

待签字符串的第一行是算法，第二行是时间戳，第三行是凭证范围，最后一行是来自任务 1：创建规范请求的规范请求的哈希。要在凭证范围中使用的服务名称为 glacier。

对于任务 3：创建签名，派生密钥可以表示为：

```
derived key = HMAC(HMAC(HMAC(HMAC("AWS4" + YourSecretAccessKey,"20120525"),"us-
east-1"),"glacier"),"aws4_request")
```

如果使用秘密访问密钥 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY，则计算出的签名为：

```
3ce5b2f2fffac9262b4da9256f8d086b4aaf42eba5f111c21681a65a127b7c2a
```

最终步骤是构造 Authorization 标头。对于示例访问密钥 AKIAIOSFODNN7EXAMPLE，标头（为了便于阅读，添加了换行符）为：

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-east-1/
glacier/aws4_request,
SignedHeaders=host;x-amz-date;x-amz-glacier-version,
Signature=3ce5b2f2fffac9262b4da9256f8d086b4aaf42eba5f111c21681a65a127b7c2a
```

为流式处理操作计算签名

上传档案（发布档案）和上传段（设置上传 ID）为流式处理操作，这些操作要求您在对请求签名以及发送请求时包括一个附加标头 x-amz-content-sha256。流式处理操作的签名步骤与其他操作的签名步骤完全相同，只是要另外添加流式处理标头。

直播标头的计算基于x-amz-content-sha256要上传的全部内容（有效负载）的SHA256 哈希值。请注意，此计算与 SHA256 树形哈希 (计算校验和) 不同。除了微不足道的情况外，有效载荷数据的 SHA 256 哈希值将与有效载荷数据的 SHA256 树形哈希值不同。

如果将有效载荷数据指定为字节数组，则可以使用以下 Java 代码片段来计算 SHA256 哈希。

```
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();
}
```

同样，在 C# 中，您可以计算有效载荷数据的 SHA256 哈希值，如以下代码片段所示。

```
public static byte[] CalculateSHA256Hash(byte[] payload)
{
    SHA256 sha256 = System.Security.Cryptography.SHA256.Create();
    byte[] hash = sha256.ComputeHash(payload);

    return hash;
}
```

流式处理 API 的示例签名计算

以下示例将引导您详细了解为 上传档案（发布档案） S3 Glacier 中的两个流媒体 APIs 中的一个创建签名。示例假定以下各项：

- 请求的时间戳为 `Mon, 07 May 2012 00:00:00 GMT`。
- 端点为美国东部（弗吉尼亚州北部）区域 。
- 内容有效负载为字符串“Welcome to S3 Glacier”。

通用请求语法（包括 JSON 正文）显示在以下示例中。请注意，其中包括了 `x-amz-content-sha256` 标头。在此简化示例中，`x-amz-sha256-tree-hash` 和 `x-amz-content-sha256` 是相同的值。但是，对于大于 1MB 的档案上传，情况就不是这样。

```
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
```

为任务 1：创建规范请求计算的规范请求格式显示如下。请注意，其中包括了流式处理标头 x-amz-content-sha256 及其值。这意味着您必须先读取有效负载并计算 SHA256 哈希值，然后再计算签名。

```
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
```

签名计算的其余操作遵循实例签名计算中所述的步骤。使用秘密访问密钥 Authorization 和访问密钥 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY 的 AKIAIOSFODNN7EXAMPLE 标头显示如下（为了便于阅读，添加了换行符）：

```
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
```

计算校验和

上传档案时，您必须包括 x-amz-sha256-tree-hash 和 x-amz-content-sha256 标头。x-amz-sha256-tree-hash 标头是您的请求正文中有效负载的校验和。此主题描述了如何计算 x-amz-sha256-tree-hash 标头。x-amz-content-sha256 标头是整个有效负载的哈希，并且是授权所必需的项目。有关更多信息，请参阅 流式处理 API 的示例签名计算。

您的请求的有效负载可以是：

- 整个档案 — 在单一请求中使用上传档案 API 上传档案时，您可以在请求正文中发送整个档案。在这种情况下，您必须包括整个档案的校验和。
- 档案段 — 使用分段上传 API 分段上传档案时，您可以在请求正文中只发送档案的一段。在这种情况下，您可以包括档案段的校验和。上传所有段后，您可以发送完成分段上传请求，该请求必须包括整个档案的校验和。

有效负载的校验和为 SHA256 树形哈希。它被称为树形哈希是因为，在计算校验和的过程中，您会计算 SHA256 哈希值树。根部的哈希值为整个档案的校验和。

> **Note**
>
> 此部分描述了一种计算 SHA256 树形哈希的方法。但是，只要能得出相同的结果，您可以使用任何方法。

您可以按以下方法计算 SHA256 树形哈希：

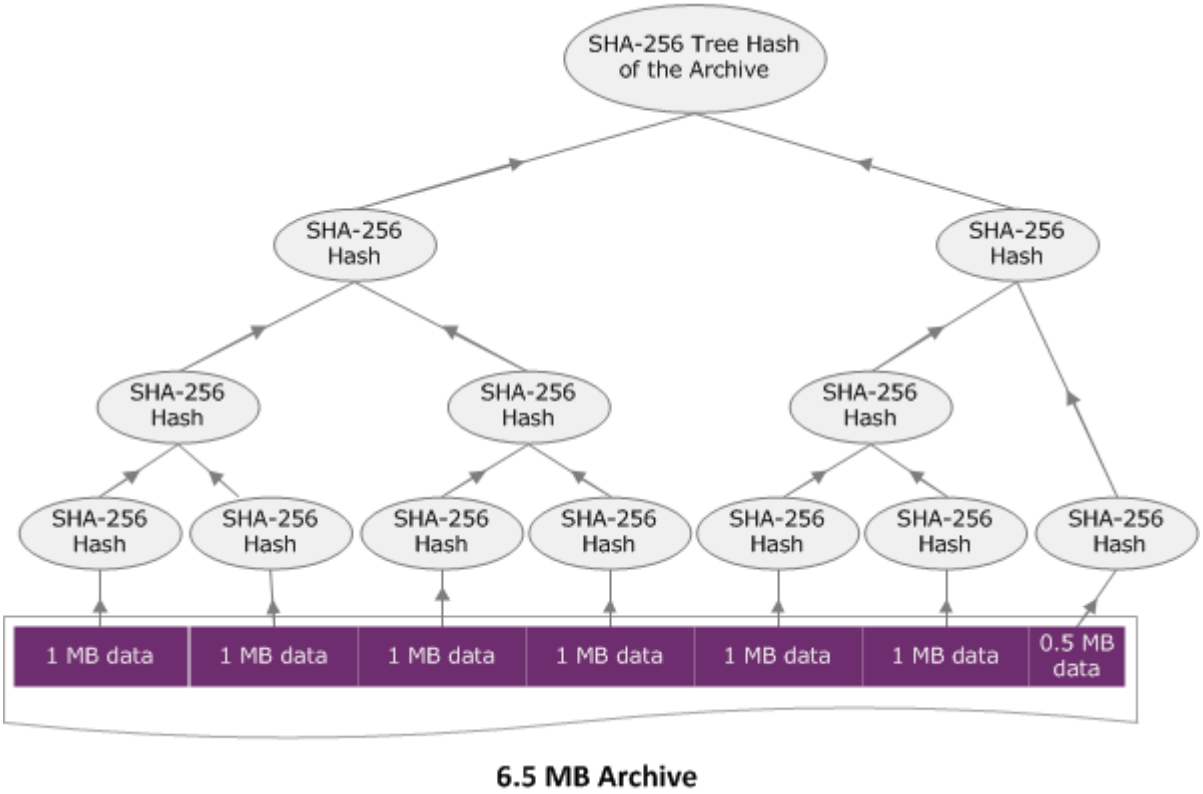
1. 针对有效负载数据的每个 1MB 区块，计算 SHA256 哈希。数据的最后一个区块可以小于 1MB。例如，如果您要上传一个 3.2MB 的档案，则您需要计算数据的前三个 1MB 区块中每一个区块的 SHA256 哈希值，然后计算剩余 0.2MB 数据的 SHA256 哈希。这些哈希值构成了树的叶节点。
2. 构建树的下一层。
   a. 连接两个连续子节点的哈希值，然后计算连接的哈希值的 SHA256 哈希。此连接和 SHA256 哈希的生成会产生这两个子节点的父节点。
   b. 如果只剩下一个子节点，您可以将该哈希值提升到树的下一层。
3. 重复步骤 2，直到结果树具有根为止。树根提供了整个档案的哈希，相应的子树根提供了分段上传中的段的哈希。

主题

- 树形哈希示例 1：在单一请求中上传档案
- 树形哈希示例 2：使用分段上传来上传档案
- 计算文件的树形哈希
- 下载数据时接收校验和

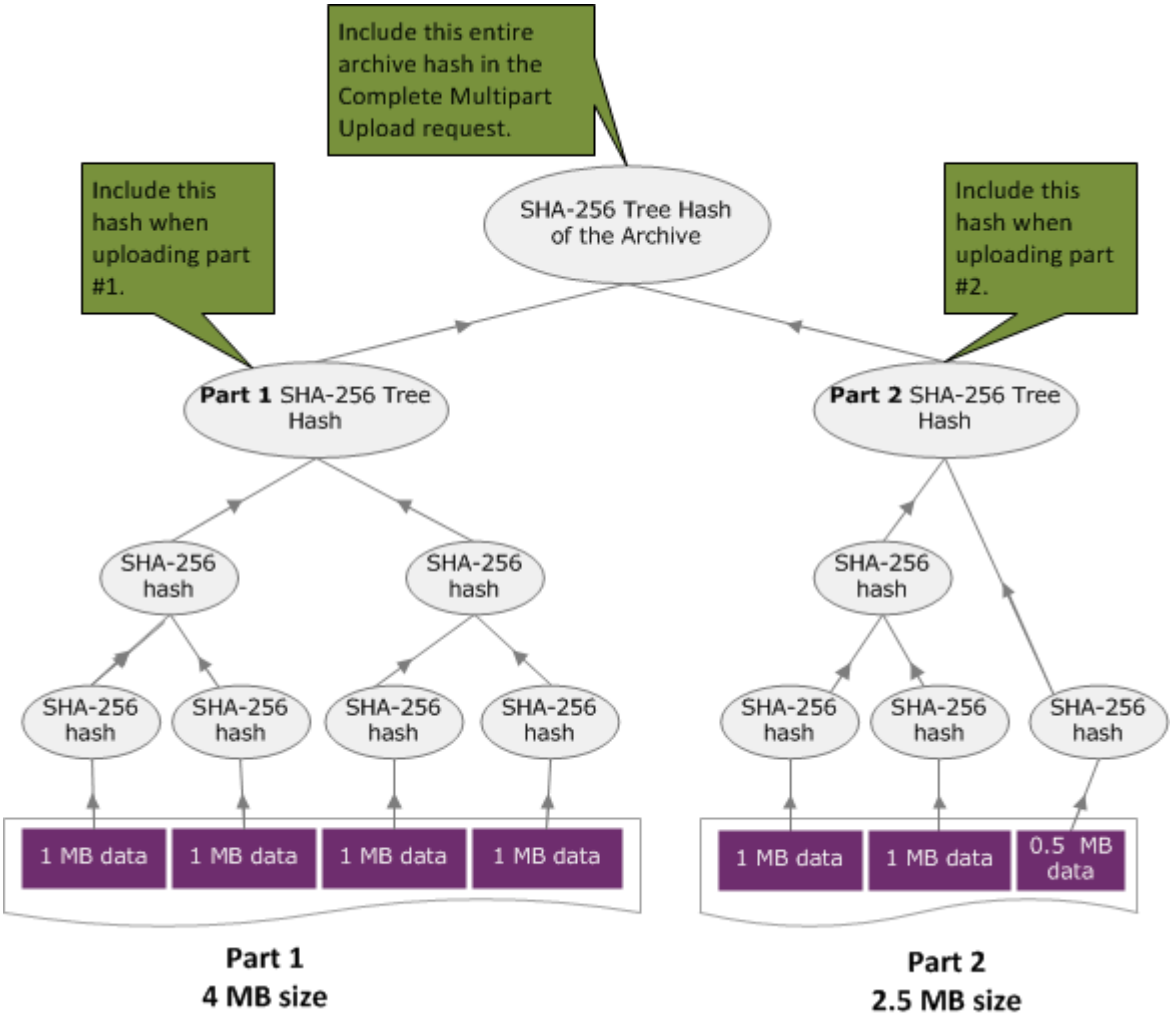
树形哈希示例 1：在单一请求中上传档案

在单一请求中使用上传档案 API（请参阅“上传档案（发布档案）”）上传档案时，请求的有效负载会包括整个档案。因此，您必须在 `x-amz-sha256-tree-hash` 请求标头中包括整个档案的树形哈希。假设您要上传一个 6.5MB 的档案。下图说明了创建档案的 SHA256 哈希的流程。您读取档案并为每个 1MB 区块计算 SHA256 哈希。此外，您还要为剩余的 0.5MB 数据计算哈希，然后按前面的步骤所述构建树。



树形哈希示例 2：使用分段上传来上传档案

在使用分段上传来上传档案时计算树形哈希的流程与在单一请求中上传档案时相同。唯一的区别是，在分段上传中，您在每个请求中只（使用 上传段（设置上传 ID） API）上传档案的一段，因此，您在 `x-amz-sha256-tree-hash` 请求标头中只提供该段的校验和。但是，上传所有段后，您必须发送完成分段上传（请参阅“完成分段上传（发布上传 ID）”）请求，并在 `x-amz-sha256-tree-hash` 请求标头中包含整个档案的树形哈希。



计算文件的树形哈希

此处显示的算法是出于演示目的而选择的。您可以根据实施情况的需要而优化代码。如果您使用 Amazon SDK 对 Amazon S3 Glacier（S3 Glacier）进行编程，则系统会为您完成树形哈希计算，您只需提供文件引用。

Example 1：Java 示例

以下示例说明如何使用 Java 计算文件的 SHA256 树形哈希。您可以通过提供文件位置作为参数来运行此示例，也可以直接从您的代码使用 `TreeHashExample.computeSHA256TreeHash` 方法。

```
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 示例

以下示例说明如何计算文件的 SHA256 树形哈希。您可以通过提供文件位置作为参数来运行此示例。

```
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;
        }
    }
}
```

下载数据时接收校验和

当您使用启动任务 API（请参阅“启动作业（POST 作业）”）检索档案时，您可以选择性地指定要检索的档案范围。类似地，当您使用获取任务输出 API（请参阅“获取任务输出（获取输出）”）下载您的数据时，您可以选择性地指定要下载的数据范围。这些范围有两个特征，您在检索和下载档案的数据时务

必要了解这两个特征。要检索的范围必须与档案以兆字节对齐。为了在下载数据时接收校验和值，要检索的范围与要下载的范围必须以树形哈希对齐。这两种类型的范围对齐的定义如下：

- 兆字节对齐-当范围 [StartByte, EndBytes] 可被 1 MB 整除且EndBytes加 1 可被 1 MB 整除或等于指定存档的末尾（存档字节大小减去 1）时 StartBytes，则以兆字节 (1024*1024) 对齐。启动任务 API 中使用的范围（如果指定了范围）必须以兆字节对齐。
- Tree-hash 对齐-当且仅当在该范围内构建的树形哈希的根等同于整个档案的树形哈希中的节点时，范围 [StartBytes，EndBytes] 才与档案进行树形哈希对齐。为了接收您下载的数据的校验和值，要检索的范围以及要下载的范围必须以树形哈希对齐。有关范围以及它们与档案树形哈希的关系的示例，请参阅“树形哈希示例：检索以树形哈希对齐的档案范围”。

  请注意，以树形哈希对齐的范围也是以兆字节对齐的。但是，以兆字节对齐的范围却不一定是以树形哈希对齐的。

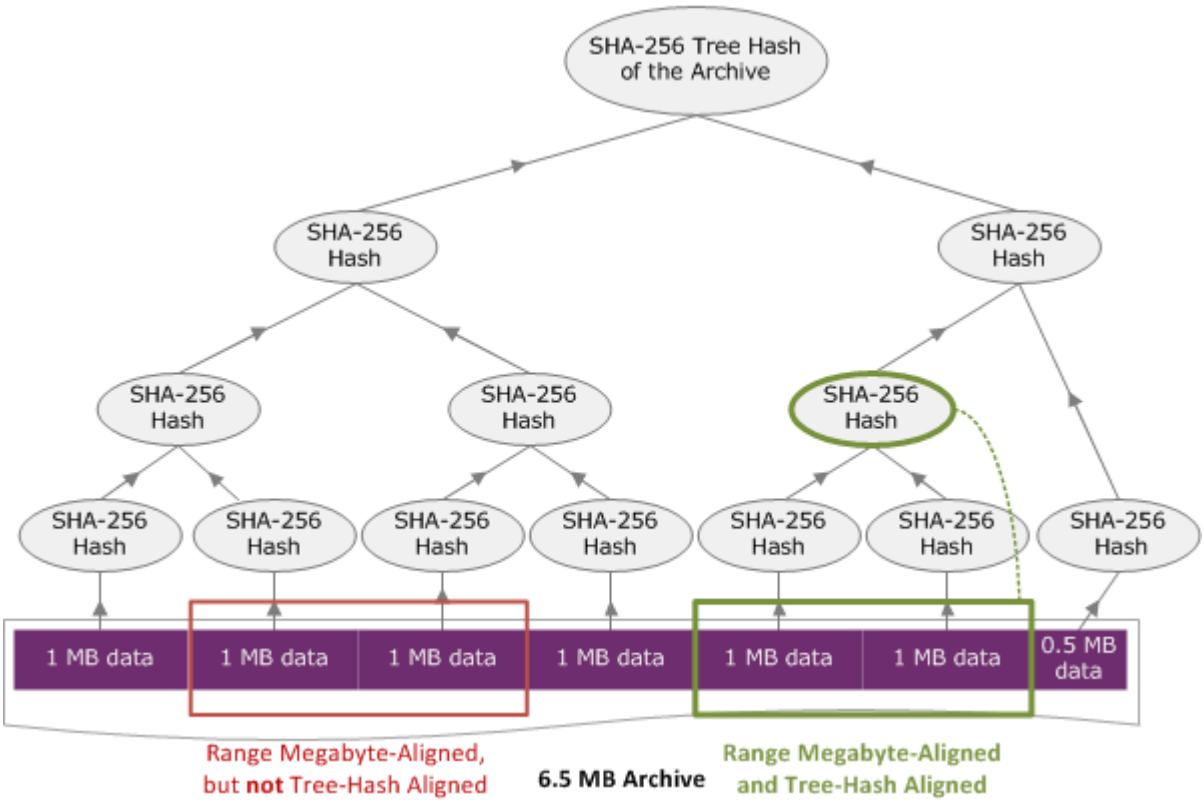
以下案例描述了您在下载档案数据时，会在哪些情况下收到校验和值：

- 如果您在启动任务请求中未指定要检索的范围，并且在获取任务请求中下载整个档案。
- 如果您在启动任务请求中未指定要检索的范围，并且在获取任务请求中指定要下载的以树形哈希对齐的范围。
- 如果您在启动任务请求中指定要检索的以树形哈希对齐的范围，并且在获取任务请求中下载整个范围。
- 如果您在启动任务请求中指定要检索的以树形哈希对齐的范围，并且在获取任务请求中指定要下载的以树形哈希对齐的范围。

如果您在启动任务请求中指定要检索的范围，而该范围未以树形哈希对齐，则您在获取任务请求中下载数据时，仍然可以获取档案数据，但系统却不会返回校验和值。

树形哈希示例：检索以树形哈希对齐的档案范围

假设您的文件库中有一个 6.5MB 的档案，您要检索该档案的 2MB。您在启动任务请求中指定 2MB 范围的方式决定了您在下载数据时是否会收到数据校验和值。下图说明了对于 6.5 MB 档案您可以下载的两个 2 MB 范围。这两个范围均以兆字节对齐，但只有一个范围以树形哈希对齐。



以树形哈希对齐的范围说明

此部分对构成以树形哈希对齐的范围的内容给出了确切的说明。您在下载档案的一部分并且指定要检索的数据范围以及要从检索的数据下载的范围时，以树形哈希对齐的范围非常重要。如果这两个范围均以树形哈希对齐，则您在下载数据时将收到校验和数据。

当且仅当在范围 [A, B] 之上构建新的树形哈希，且该范围的树形哈希的根相当于整个档案的树形哈希中的一个节点时，范围 [A, B] 才相对于档案是以树形哈希对齐的。您可以看到 树形哈希示例：检索以树形哈希对齐的档案范围 中的图表显示了这一点。在此部分中，我们提供了树形哈希对齐的说明。

可将 [P, Q) 视为 N 兆字节 (MB) 的档案的范围查询，而 P 和 Q 是 1 MB 的倍数。请注意，实际包括范围为 [P MB, Q MB – 1 字节]，但为了简单起见，我们将它显示为 [P, Q)。基于这些考虑因素得知：

- 如果 P 为奇数，则只有一个可能的以树形哈希对齐的范围即，[P, P + 1 MB)。
- 如果 P 是偶数，k 是最大数，其中 P 可以写成 2k * X，则最多有 k 个以 P 开头的树形哈希对齐的范围。 X 是大于 0 的整数。以树形哈希对齐的范围分为以下类别：
  - 对于每个 i，其中 (0 <= i <= k) 并且 $P + 2^i < N$，则 $[P, Q + 2^i)$ 是以树形哈希对齐的范围。
  - P = 0 是 A = 2[lgN]*0 时的特殊情况

错误响应

如果发生错误，API 将返回下列异常之一：

代码	描述	HTTP 状态代码	类型
AccessDeniedException	如果有人尝试访问 AWS Identity and Access Management (IAM) 策略不允许的资源，或者请求 URI 中使用了错误的 AWS 账户 ID，则返回。有关更多信息，请参阅适用于 Amazon S3 Glacier 的 Identity and Access Management。	403 Forbidden	客户端
BadRequest	如果无法处理请求，则返回此异常。	400 Bad Request	客户端
ExpiredTokenException	如果请求中使用的安全令牌已过期，则返回此异常。	403 Forbidden	客户端
InsufficientCapacityException	如果没有足够的容量处理此加速请求，则返回此代码。此错误仅适用于加速检索，不适用于标准或批量检索。	503 Service Unavailable	服务器
InvalidParameterValueException	如果错误地指定了请求的参数，则返回此异常。	400 Bad Request	客户端
InvalidSignatureException	如果请求签名无效，则返回此异常。	403 Forbidden	客户端
LimitExceededException	如果请求导致超过文件库限制、标签限制或预配置容量限制中的任何一项，则返回此代码。	400 Bad Request	客户端
MissingAuthenticationTokenException	如果没有为请求找到身份验证数据，则返回此异常。	400 Bad Request	客户端

代码	描述	HTTP 状态代码	类型
MissingParameterValueException	如果请求中缺失必需的标头或参数，则返回此异常。	400 Bad Request	客户端
PolicyEnforcedException	如果检索作业将超出当前数据策略的检索速率限制，则返回此异常。有关数据检索策略的更多信息，请参阅S3 Glacier 数据检索策略。	400 Bad Request	客户端
ResourceNotFoundException	如果指定的资源（例如文件库、上传 ID 或任务 ID）不存在，则返回此异常。	404 Not Found	客户端
RequestTimeoutException	如果正在上传档案并且 Amazon S3 Glacier（S3 Glacier）在接收上传时超时，则返回此异常。	408 Request Timeout	客户端
SerializationException	如果请求正文无效，则返回此异常。如果包括 JSON 有效负载，则检查其格式是否正确。	400 Bad Request	客户端
ServiceUnavailableException	如果服务无法完成请求，则返回此异常。	500 Internal Server Error	服务器
ThrottlingException	如果您需要降低向 S3 Glacier 发送请求的速率，则返回此异常。	400 Bad Request	客户端
UnrecognizedClientException	如果访问密钥 ID 或安全令牌无效，则返回此异常。	400 Bad Request	客户端

各种 S3 Glacier 会 APIs 返回相同的异常，但会返回不同的异常消息，以帮助您解决遇到的特定错误。

S3 Glacier 会在响应正文中返回错误信息。以下示例显示了某些错误响应。

示例 1：具有不存在的任务 ID 的描述任务请求

假设您为不存在的任务发送 描述任务（获取任务 ID） 请求。即，您指定一个不存在的任务 ID。

```
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
```

作为响应，S3 Glacier 返回以下错误响应。

```
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"
  }
```

其中：

代码

常规异常之一。

类型：字符串

消息

特定于返回错误的 API 的错误条件一般描述。

类型：字符串

类型

错误源。该字段可以为以下值之一：`Client`、`Server` 或 `Unknown`。

类型：字符串。

请注意前面响应中的以下情况：

- 对于错误响应，S3 Glacier 返回状态代码值 `4xx` 和 `5xx`。在此示例中，状态代码为 `404 Not Found`。
- `Content-Type` 标头值 `application/json` 表示正文中的 JSON
- 正文中的 JSON 提供了错误信息。

在之前的请求中，假设您指定不存在的文件库，而不是错误的任务 ID。响应会返回不同的消息。

```
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"
}
```

示例 2：请求参数具有无效值的列出任务请求

在此示例中，您发送 列出任务（获取任务） 请求以检索具有特定 `statuscode` 的文件库任务，而您提供了错误的 `statuscode` 值 `finished`，而不是可接受的值 `InProgress`、`Succeeded` 或 `Failed`。

```
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 返回 `InvalidParameterValueException` 以及相应的消息。

```
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"
}
```

文件库操作

以下是可供在 S3 Glacier 中使用的文件库操作。

主题

- 中止文件库锁定（DELETE 锁定策略）
- 向文件库添加标签（POST 标签添加）
- 创建文件库（设置文件库）
- 完成文件库锁定 (POST lockId)
- 删除文件库
- 删除文件库访问策略 (DELETE access-policy)
- 删除文件库通知（删除通知配置）
- 描述文件库（获取文件库）
- 获取文件库访问策略 (GET access-policy)
- 获取文件库锁定（GET 锁定策略）
- 获取文件库通知（获取通知配置）
- 启动文件库锁定（POST 锁定策略）
- 列出文件库的标签（GET 标签）
- 列出文件库（获取文件库）
- 从文件库删除标签（POST 标签删除）
- 设置文件库访问策略 (PUT access-policy)
- 设置文件库通知配置（设置通知配置）

中止文件库锁定（DELETE 锁定策略）

描述

如果文件库锁定未处于 Locked 状态，则此操作会停止文件库锁定过程。如果在请求此操作时文件库锁定处于 Locked 状态，则此操作会返回 AccessDeniedException 错误。停止文件库锁定过程会从指定文件库中删除文件库锁定策略。

通过调用 InProgress 将文件库锁定置于 启动文件库锁定（POST 锁定策略） 状态。通过调用 Locked 将文件库锁定置于 完成文件库锁定 (POST lockId) 状态。您可通过调用 获取文件库锁定（GET 锁定策略） 获取文件库锁定的状态。有关文件库锁定过程的更多信息，请参阅 S3 Glacier 文件库锁定。有关文件库锁定策略的更多信息，请参阅 文件库锁定策略。

此操作是幂等的。如果文件库锁定处于 InProgress 状态或没有与文件库关联的策略，则您可多次成功地调用此操作。

请求

要删除文件库锁定策略，请向文件库的 DELETE 子资源的 URI 发送 HTTP lock-policy 请求。

语法

```
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值为 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

如果成功删除此策略，则 S3 Glacier 会返回 `HTTP 204 No Content` 响应。

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例演示如何停止文件库锁定过程。

示例请求

在此示例中，`DELETE` 请求将发送到名为 **`examplevault`** 的文件库的 `lock-policy` 子资源。

```
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
```

响应示例

如果成功删除此策略，则 S3 Glacier 会返回 `HTTP 204 No Content` 响应，如以下示例中所示。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

相关部分

- 完成文件库锁定 (POST lockId)
- 获取文件库锁定（GET 锁定策略）
- 启动文件库锁定（POST 锁定策略）

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

向文件库添加标签（POST 标签添加）

此操作会向文件库添加指定的标签。每个标签由一个键和一个值组成。每个文件库可最多有 50 个标签。如果您的请求会导致超出文件库的标签限制，则此操作会引发 `LimitExceededException` 错误。

如果在文件库上的某个指定键的下面已存在一个标签，则将覆盖现有的键值。有关标签的更多信息，请参阅 标记 Amazon S3 Glacier 资源。

请求语法

要向文件库添加标签，请将 HTTP POST 请求发送到标签 URI，如以下语法示例中所示。

```
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`值为 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

请求参数

名称	描述	必填
`operation=add`	带有 `operation` 值的单个查询字符串参数 `add`，用于与 从文件库删除标签（POST 标签删除）进行区分。	是

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

请求正文中包含以下 JSON 字段。

标签

要添加到文件库的标签。每个标签由一个键和一个值组成。该值可为空字符串。

类型：字符串到字符串映射

长度约束：最小长度为 1。最大长度为 10。

必需：是

响应

如果操作请求成功，则该服务会返回 HTTP `204 No Content` 响应。

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

以下示例发送一个带有要添加到文件库的标签的 HTTP POST 请求。

```
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"
    }
}
```

响应示例

如果请求成功，S3 Glacier 会返回 `HTTP 204 No Content`，如以下示例中所示。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

相关部分

- 列出文件库的标签（GET 标签）
- 从文件库删除标签（POST 标签删除）

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

创建文件库（设置文件库）

描述

此操作会使用指定的名称创建新的文件库。 文件库的名称在一个 AWS 区域内必须是唯一的 AWS 账户。您最多可以为每个账户创建 1 000 个文件库。有关创建更多文件库的信息，请转到 Amazon S3 Glacier 产品详细信息页。

为文件库命名时，您必须使用以下准则。

- 名称长度在 1 和 255 个字符之间。
- 允许的字符包括 a-z、A-Z、0-9、'-'（下划线）、'_'（连字符）和 '.'（半角句点）。

此操作是幂等的，您可以多次发送相同的请求，但在 Amazon S3 Glacier（S3 Glacier）第一次创建指定的文件库后，它不会再产生影响。

请求

语法

要创建文件库，请将 HTTP PUT 请求发送到要创建的文件库的 URI。

```
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**
>
> 其`AccountId`值是 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作的请求正文必须为空（0 字节）。

响应

语法

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
```

响应标头

除了所有操作通用的响应标头外，成功的响应中还包括以下响应标头。有关通用响应标头的信息，请参阅常用响应标头。

名称	描述
Location	已创建的文件库的相对 URI 路径。 类型：字符串

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

以下示例发送 HTTP PUT 请求，以创建名为 `examplevault` 的文件库。

```
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
```

响应示例

S3 Glacier 创建文件库并在 `Location` 标头中返回文件库的相对 URI 路径。无论请求中指定的是账户 ID 还是连字符（“`Location`”），账户 ID 始终都会显示在 - 标头中。

```
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
```

相关部分

- 列出文件库（获取文件库）
- 删除文件库
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

完成文件库锁定 (POST lockId)

描述

此操作通过将文件库锁定的状态从 `InProgress` 状态转换为 `Locked` 状态来完成文件库锁定过程，这会导致文件库锁定策略不可更改。通过调用 `InProgress` 将文件库锁定置于 启动文件库锁定（POST 锁定策略） 状态。您可通过调用 获取文件库锁定（GET 锁定策略） 获取文件库锁定的状态。有关文件库锁定过程的更多信息，请参阅 S3 Glacier 文件库锁定。

此操作是幂等的。如果文件库锁定处于 `Locked` 状态而且提供的锁定 ID 与最初用于锁定文件库的锁定 ID 匹配，则此请求始终将会成功。

如果当文件库锁定处于 Locked 状态时在请求中传递无效的锁定 ID，则此操作会返回 AccessDeniedException 错误。如果当文件库锁定处于 InProgress 状态时在请求中传递无效的锁定 ID，则此操作会引发 InvalidParameter 错误。

请求

要完成文件库锁定过程，请向文件库的 POST 子资源的 URI 发送带有有效的锁定 ID 的 HTTP lock-policy 请求。

语法

```
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值为 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

lockId 值是从 启动文件库锁定（POST 锁定策略） 请求获取的锁定 ID。

请求参数

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

如果操作请求成功，则该服务会返回 HTTP 204 No Content 响应。

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

以下示例发送带有锁定 ID 的 HTTP POST 请求以完成文件库锁定过程。

```
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
```

响应示例

如果请求成功，Amazon S3 Glacier（S3 Glacier）将返回 `HTTP 204 No Content` 响应，如以下示例中所示。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

相关部分

- 中止文件库锁定（DELETE 锁定策略）
- 获取文件库锁定（GET 锁定策略）
- 启动文件库锁定（POST 锁定策略）

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

删除文件库

描述

此操作会删除文件库。只有在上次清单盘点时文件库中没有任何档案，并且自上次清单盘点以来文件库没有执行过任何写入操作，Amazon S3 Glacier（S3 Glacier）才会删除文件库。如果不满足其中的任一条件，文件库删除操作会失败（即，不会删除文件库），并且 S3 Glacier 返回错误。

您可以使用 描述文件库（获取文件库） 操作，它会提供文件库信息，包括文件库中的档案数；但是，该信息是基于 S3 Glacier 上次生成的文件库清单的。

此操作是幂等的。

> **Note**
>
> 当您删除文件库时，已附加到文件库的文件库访问策略也会被删除。有关文件库访问策略的更多信息，请参阅库访问策略。

请求

要删除文件库，请向文件库资源 URI 发送 DELETE 请求。

语法

```
DELETE /AccountId/vaults/VaultName HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-” (连字符) ，在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

以下示例会删除名为 `examplevault` 的文件库。该示例请求是针对要删除的资源（文件库）的 URI 的 `DELETE` 请求。

```
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
```

响应示例

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

相关部分

- 创建文件库（设置文件库）
- 列出文件库（获取文件库）
- 启动作业（POST 作业）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

删除文件库访问策略 (DELETE access-policy)

描述

此操作将删除与指定文件库关联的访问策略。该操作最终是一致的 - 也就是说，Amazon S3 Glacier（S3 Glacier）可能需要花一些时间来完全删除访问策略，并且在您发送删除请求后，短时间内可能仍将看到该策略的效果。

此操作是幂等的。您可以多次调用删除操作，即使没有与文件库关联的策略。有关文件库访问策略的更多信息，请参阅库访问策略。

请求

要删除当前文件库访问策略，请向文件库的 `access-policy` 子资源的 URI 发送一个 HTTP `DELETE` 请求。

语法

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

作为响应，S3 Glacier 会在成功删除策略时返回 `204 No Content`。

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例演示如何删除文件库访问策略。

示例请求

在此示例中，`DELETE` 请求将发送到名为 **`examplevault`** 的文件库的 `access-policy` 子资源。

```
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
```

响应示例

作为响应，如果已成功删除策略，则 S3 Glacier 将返回 `204 No Content`，如以下示例所示。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

相关部分

- 获取文件库访问策略 (GET access-policy)
- 设置文件库访问策略 (PUT access-policy)

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

删除文件库通知（删除通知配置）

描述

此操作会删除为文件库 设置文件库通知配置（设置通知配置） 设置的通知配置。该操作最终是一致的即，Amazon S3 Glacier（S3 Glacier）可能需要花一些时间来完全禁用通知，并且您在发送删除请求后的短时间内，仍可能会收到一些通知。

请求

要删除文件库的通知配置，请向文件库的 `DELETE` 子资源发送 `notification-configuration` 请求。

语法

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例展示了如何删除文件库的通知配置。

示例请求

在此示例中，DELETE 请求会发送到名为 notification-configuration 的文件库的 examplevault 子资源。

```
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
```

响应示例

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

相关部分

- 获取文件库通知（获取通知配置）
- 设置文件库通知配置（设置通知配置）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

描述文件库（获取文件库）

描述

此操作会返回有关文件库的信息，包括文件库的Amazon 资源名称 (ARN)、文件库的创建日期、文件库中包含的档案数，以及文件库中所有档案的总大小。档案数量及其总大小截至 Amazon S3 Glacier（S3 Glacier）上次生成的文件库清单（参阅在 Amazon S3 Glacier 中处理文件库）。S3

Glacier 大约每天都会生成文件库清单。这意味着，如果您在文件库中添加档案或者从文件库中删除档案，然后立即发送描述文件库请求，则响应可能不会反映这些更改。

请求

要获取有关文件库的信息，请向特定文件库资源的 URI 发送 GET 请求。

语法

```
GET /AccountId/vaults/VaultName HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

语法

```
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
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

CreationDate

创建文件库的 UTC 日期。

类型：以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

LastInventoryDate

S3 Glacier 完成上次文件库清单盘点的 UTC 日期。有关启动文件库清单的信息，请参阅“启动作业（POST 作业）”。

类型：以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

NumberOfArchives

上次文件库库存盘点时，文件库中的档案数。如果文件库中尚未运行清单操作（例如，您刚刚创建了文件库），则此字段将返回 null。

类型：数字

SizeInBytes

截止到上次编制清单日期，文件库中的档案总大小（以字节为单位），包括每个档案所具有的任何开销。如果文件库中尚未运行清单操作（例如，您刚刚创建了文件库），则此字段将返回 `null`。

类型：数字

VaultARN

文件库的Amazon 资源名称 (ARN)。

类型：字符串

VaultName

在创建时间指定的文件库名称。文件库名称也包括在文件库的 ARN 中。

类型：字符串

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

以下示例展示了如何获取有关名为 `examplevault` 的文件库的信息。

```
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
```

响应示例

```
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"
}
```

相关部分

- 创建文件库（设置文件库）
- 列出文件库（获取文件库）
- 删除文件库
- 启动作业（POST 作业）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

获取文件库访问策略 (GET access-policy)

描述

此操作将检索文件库上设置的 `access-policy` 子资源有关设置此子资源的更多信息，请参阅设置文件库访问策略 (PUT access-policy)。如果未在文件库上设置访问策略，该操作将返回 `404 Not found` 错误。有关文件库访问策略的更多信息，请参阅库访问策略。

请求

要返回当前的文件库访问策略，请向文件库的 `GET` 子资源的 URI 发送一个 HTTP `access-policy` 请求。

语法

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

作为响应，Amazon S3 Glacier（S3 Glacier）将在响应正文中以 JSON 格式返回文件库访问策略。

语法

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length

{
  "Policy": "string"
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

Policy

以 JSON 字符串形式表示的文件库访问策略（使用“\”作为转义符）。

类型：字符串

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例演示如何获取文件库访问策略。

示例请求

在此示例中，GET 请求会发送到文件库的 `access-policy` 子资源的 URI。

```
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
```

响应示例

如果请求成功，S3 Glacier 将在响应正文中以 JSON 字符串形式返回文件库访问策略。返回的 JSON 字符串使用“\”作为转义符，如设置文件库访问策略 (PUT access-policy)示例中所示。但为了便于阅读，以下示例显示了不带转义符的返回的 JSON 字符串。

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

相关部分

- 删除文件库访问策略 (DELETE access-policy)
- 设置文件库访问策略 (PUT access-policy)

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

获取文件库锁定（GET 锁定策略）

描述

此操作会从指定文件库上设置的 `lock-policy` 子资源中检索以下属性：

- 在文件库上设置的文件库锁定策略。
- 文件库锁定的状态（InProgess 或 Locked）。
- 当锁定 ID 到期时。锁定 ID 用于完成文件库锁定过程。
- 当文件库锁定启动并进入 InProgress 状态时。

通过调用 InProgress 将文件库锁定置于 启动文件库锁定（POST 锁定策略） 状态。通过调用 Locked 将文件库锁定置于 完成文件库锁定 (POST lockId) 状态。您可通过调用 中止文件库锁定（DELETE 锁定策略） 停止文件库锁定过程。有关文件库锁定过程的更多信息，请参阅 S3 Glacier 文件库锁定。

如果未在文件库上设置文件库锁定策略，则该操作会返回 404 Not found 错误。有关文件库锁定策略的更多信息，请参阅 文件库锁定策略。

请求

要返回当前的文件库锁定策略和其他属性，请向文件库的 GET 子资源的 URI 发送一个 HTTP lock-policy 请求，如以下语法示例中所示。

语法

```
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**
>
> 该AccountId值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

作为响应，Amazon S3 Glacier（S3 Glacier）将在响应正文中以 JSON 格式返回文件库访问策略。

语法

```
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"
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

Policy

以 JSON 字符串形式表示的文件库锁定策略（使用“\”作为转义符）。

类型：字符串

状态

文件库锁定的状态。

类型：字符串

有效值：`InProgress|Locked`

ExpirationDate

锁定 ID 到期的 UTC 日期和时间。如果文件库锁定处于 `null` 状态，则此值可为 `Locked`。

类型：以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

CreationDate

将文件库锁定置于 `InProgress` 状态的 UTC 日期和时间。

类型：以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例演示如何获取文件库锁定策略。

示例请求

在此示例中，`GET` 请求会发送到文件库的 `lock-policy` 子资源的 URI。

```
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
```

响应示例

如果请求成功，S3 Glacier 将在响应正文中以 JSON 字符串形式返回文件库访问策略。返回的 JSON 字符串使用“\”作为转义符，如启动文件库锁定（POST 锁定策略）示例请求中所示。但为了便于阅读，以下示例显示了不带转义符的返回的 JSON 字符串。

```
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"
}
```

相关部分

- 中止文件库锁定（DELETE 锁定策略）
- 完成文件库锁定 (POST lockId)

- 启动文件库锁定（POST 锁定策略）

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

获取文件库通知（获取通知配置）

描述

此操作会检索文件库中设置的 `notification-configuration` 子资源（请参阅“设置文件库通知配置（设置通知配置）”）。如果未设置文件库的通知配置，则该操作会返回 `404 Not Found` 错误。有关文件库通知的更多信息，请参阅“在 Amazon S3 Glacier 中配置文件库通知”。

请求

要检索通知配置信息，请向文件库的 `GET` 子资源的 URI 发送 `notification-configuration` 请求。

语法

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

语法

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length
{
  "Events": [
    String,
    ...
  ],
  "SNSTopic": String
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

事件

Amazon S3 Glacier（S3 Glacier）将向指定 Amazon SNS 主题发送通知的一个或多个事件的列表。有关您可以为其配置文件库以发布通知的文件库事件的信息，请参阅“设置文件库通知配置（设置通知配置）”。

类型：数组

SNSTopic

Amazon Simple Notification Service (Amazon SNS) 主题的 Amazon 资源名称（ARN）。有关更多信息，请参阅《Amazon Simple Notification Service 入门指南》中的 Amazon SNS 入门。

类型：字符串

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例展示了如何检索文件库的通知配置。

示例请求

在此示例中，GET 请求会发送到文件库的 `notification-configuration` 子资源。

```
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
```

响应示例

成功的响应会在响应正文中以 JSON 格式显示审核记录配置文档。在此示例中，该配置显示了两个事件（`ArchiveRetrievalCompleted` 和 `InventoryRetrievalCompleted`）的通知会发送到 Amazon SNS 主题 `arn:aws:sns:us-west-2:012345678901:mytopic`。

```
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"
}
```

相关部分

- 删除文件库通知（删除通知配置）
- 设置文件库通知配置（设置通知配置）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

启动文件库锁定（POST 锁定策略）

描述

此操作通过执行以下操作启动文件库锁定过程：

- 在指定文件库上安装文件库锁定策略。
- 将文件库锁定的锁定状态设置为 `InProgress`。
- 返回一个用于完成文件库锁定过程的锁定 ID。

您可为每个文件库设置一个文件库锁定策略，而且此策略大小最多为 20 KB。有关文件库锁定策略的更多信息，请参阅 文件库锁定策略。

您必须在文件库锁定进入 `InProgress` 状态后的 24 个小时内完成文件库锁定过程。在 24 个小时的窗口结束之后，锁定 ID 将会到期，文件库将自动退出 `InProgress` 状态，并将从文件库中删除文件库锁定策略。您可调用 完成文件库锁定 (POST lockId)，通过将文件库锁定的状态设置为 `Locked` 来完成文件库锁定过程。

> **Note**
>
> 在文件库锁定处于 Locked 状态后，您不能为文件库启动新的文件库锁定。

您可通过调用 中止文件库锁定（DELETE 锁定策略） 停止文件库锁定过程。您可通过调用 获取文件库锁定（GET 锁定策略） 获取文件库锁定的状态。有关文件库锁定过程的更多信息，请参阅 S3 Glacier 文件库锁定。

如果在文件库锁定处于 InProgress 状态时调用此操作，则此操作会返回 AccessDeniedException 错误。当文件库锁定处于 InProgress 状态时，您必须先调用 中止文件库锁定（DELETE 锁定策略），然后才可以启动新的文件库锁定策略。

请求

要启动文件库锁定过程，请向文件库的 POST 子资源的 URI 发送 HTTP lock-policy 请求，如以下语法示例中所示。

语法

```
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值为 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

请求正文中包含以下 JSON 字段。

Policy

以 JSON 字符串形式表示的文件库锁定策略（使用“\”作为转义符）。

类型：字符串

必需：是

响应

如果该策略被接受，Amazon S3 Glacier（S3 Glacier）将返回 `HTTP 201 Created` 响应。

语法

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-lock-id: lockId
```

响应标头

除了所有操作通用的响应标头外，成功的响应中还包括以下响应标头。有关通用响应标头的信息，请参阅常用响应标头。

名称	描述
`x-amz-lock-id`	用于完成文件库锁定过程的锁定 ID。

名称	描述
	类型：字符串

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

以下示例将向文件库的 PUT 子资源的 URI 发送 HTTP `lock-policy` 请求。`Policy` JSON 字符串使用“\”作为转义符。

```
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\"}}}]}"}
```

响应示例

如果请求成功，则 S3 Glacier 会返回 `HTTP 201 Created` 响应，如以下示例中所示。

```
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
```

相关部分

- 中止文件库锁定（DELETE 锁定策略）
- 完成文件库锁定 (POST lockId)
- 获取文件库锁定（GET 锁定策略）

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

列出文件库的标签（GET 标签）

此操作会列出已添加到文件库的所有标签。如果没有标签，则此操作会返回空映射。有关标签的更多信息，请参阅 标记 Amazon S3 Glacier 资源。

请求语法

要列出文件库的标签，请将 HTTP GET 请求发送到标签 URI，如以下语法示例中所示。

```
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`值为 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况

下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

如果此操作成功，则该服务将会发送回 HTTP `200 OK` 响应。

响应语法

```
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"
      }
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

标签

已添加到文件库的标签。每个标签由一个键和一个值组成。

类型：字符串到字符串映射

必需：是

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例：列出文件库的标签

以下示例列出了文件库的标签。

示例请求

在此示例中，将发送一个 GET 请求以从指定文件库中检索标签的列表。

```
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
```

响应示例

如果请求成功，则 Amazon S3 Glacier（S3 Glacier）会返回带有一个标签列表的 `HTTP 200 OK`，如以下示例中所示。

```
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"
      }
}
```

相关部分

- 向文件库添加标签（POST 标签添加）
- 从文件库删除标签（POST 标签删除）

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

列出文件库（获取文件库）

描述

此操作会列出调用用户的账户拥有的所有文件库。响应中返回的列表按文件库名称的 ASCII 顺序排序。

默认情况下，每次请求时此操作最多返回 10 个项目。如果有更多文件库要列出，则响应正文中的 `marker` 字段会包含文件库的Amazon 资源名称 (ARN)，新的列出文件库请求会从该名称处继续列表；否则，`marker` 字段为 `null`。在下一个“列出文件库”请求中，您可以将 `marker` 参数设置为 Amazon S3 Glacier（S3 Glacier）为上一个“列出文件库”请求所回复的值。您也可以通过在请求中指定 `limit` 参数来限制响应中返回的文件库数。

请求

要获取文件库列表，您需要向文件库资源发送 `GET` 请求。

语法

```
GET /AccountId/vaults HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> `AccountId`值为 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

请求参数

此操作接受以下请求参数。

名称	描述	必填
`limit`	指定要返回的文件库最大数目。默认限制为 10。返回的文件库数可能少于指定的限制值，但永远不会超过限制值。 类型：字符串 限制：最小整数值为 1。最大整数值为 10。	否
`marker`	用于分页的字符串。`marker` 指定应从其开始列出文件库的文件库 ARN。（`marker` 指定的文件库不包括在返回的列表中。）从之前的列出文件库响应获取 `marker` 值。只有在您要继续对之前的列出文件库请求中开始的结果进行分页，您才需要包括 `marker`。如果为标记指定空值 ("")，则系统会返回从第一个文件库开始的文件库列表。 类型：字符串	否

名称	描述	必填
	限制：无	

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

语法

```
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
   },
   ...
  ]
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

CreationDate

以协调世界时 (UTC) 来表示的文件库创建日期。

类型：字符串。以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

LastInventoryDate

以协调世界时 (UTC) 来表示的上次文件库库存盘点日期。如果文件库中尚未运行清单操作（例如，您刚刚创建了文件库），则此字段可能为空。有关启动文件库清单的信息，请参阅“启动作业（POST 作业）”。

类型：以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

Marker

表示从何处继续对结果进行分页的 `vaultARN`。您可以在另一个列出文件库请求中使用 `marker` 来获取列表中的更多文件库。如果没有更多文件库，则此值为 `null`。

类型：字符串

NumberOfArchives

截止到上次编制清单日期，文件库中的档案数。

类型：数字

SizeInBytes

截止到上次编制清单日期，文件库中所有档案的总大小（以字节为单位），包括每个档案所具有的任何开销。

类型：数字

VaultARN

文件库的Amazon 资源名称 (ARN)。

类型：字符串

VaultList

数据元数组，其中的每个数据元均提供了文件库描述。

类型：数组

VaultName

文件库名称。

类型：字符串

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例：列出所有文件库

以下示例列出了文件库。由于请求中没有指定 `marker` 和 `limit` 参数，因此，系统最多会返回 10 个文件库。

示例请求

```
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
```

响应示例

`Marker` 为 `null`，表示没有更多文件库要列出。

```
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"
   }
  ]
}
```

示例：文件库的部分列表

以下示例会返回从 marker 指定的文件库开始的两个文件库。

示例请求

```
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
```

响应示例

列表中返回了两个文件库。Marker 包含文件库 ARN，以便在另一个列出文件库请求中继续分页。

```
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"
   }
  ]
}
```

相关部分

- 创建文件库（设置文件库）
- 删除文件库
- 启动作业（POST 作业）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

从文件库删除标签（POST 标签删除）

此操作会从已附加到文件库的标签集中删除一个或多个标签。有关标签的更多信息，请参阅 标记 Amazon S3 Glacier 资源。

此操作是幂等的。即使没有已附加到文件库的标签，此操作也将会成功。

请求语法

要从文件库中删除标签，请将 HTTP POST 请求发送到标签 URI，如以下语法示例中所示。

```
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`值为 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

请求参数

名称	描述	必填
`operation=remove`	带有 `operation` 值的单个查询字符串参数 `remove`，用于与 向文件库添加标签（POST 标签添加）进行区分。	是

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

请求正文中包含以下 JSON 字段。

TagKeys

标签键的列表。从文件库中删除每个对应的标签。

类型：字符串的数组

长度约束：列表中最少 1 个项。列表中最多 10 个项。

必需：是

响应

如果此操作成功，则该服务会发送回带有空 HTTP 正文的 HTTP `204 No Content` 响应。

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

以下示例发送一个 HTTP POST 请求以删除指定的标签。

```
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"
   ]
}
```

响应示例

如果请求成功，Amazon S3 Glacier（S3 Glacier）会返回 `HTTP 204 No Content`，如以下示例中所示。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

相关部分

- 向文件库添加标签（POST 标签添加）
- 列出文件库的标签（GET 标签）

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

设置文件库访问策略 (PUT access-policy)

描述

此操作会为文件库配置一个访问策略并覆盖现有策略。要配置文件库访问策略，请向文件库的 PUT 子资源发送 `access-policy` 请求。您可以为每个文件库设置一个访问策略，该策略的大小最多为 20 KB。有关文件库访问策略的更多信息，请参阅库访问策略。

请求

语法

要设置文件库访问策略，请向文件库的 `access-policy` 子资源的 URI 发送 HTTP PUT 请求，如以下语法示例所示。

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-” (连字符) ，在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

请求正文中包含以下 JSON 字段。

Policy

以 JSON 字符串形式表示的文件库访问策略（使用“\”作为转义符）。

类型：字符串

必需：是

响应

作为响应，S3 Glacier 会返回 `204 No Content`（如果接受了策略）。

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

以下示例将向文件库的 PUT 子资源的 URI 发送 HTTP `access-policy` 请求。`Policy` JSON 字符串使用“\”作为转义符。

```
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\"}]}"}
```

响应示例

如果请求成功，Amazon S3 Glacier（S3 Glacier）会返回 `HTTP 204 No Content`，如以下示例中所示。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

相关部分

- 删除文件库访问策略 (DELETE access-policy)
- 获取文件库访问策略 (GET access-policy)

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

设置文件库通知配置（设置通知配置）

描述

在 Amazon S3 Glacier（S3 Glacier）中，检索档案和文件库清单是异步操作，您必须首先为这些操作启动任务，等到任务完成后，您才能下载任务输出。您可以将文件库配置为在这些任务完成时向 Amazon Simple Notification Service (Amazon SNS) 主题发布消息。您可以使用此操作来设置文件库中的通知配置。有关更多信息，请参阅 在 Amazon S3 Glacier 中配置文件库通知。

要配置文件库通知，请向文件库的 `notification-configuration` 子资源发送设置请求。通知配置是特定于文件库的；因此，它也称为文件库子资源。请求应包括提供 Amazon Simple Notification Service (Amazon SNS) 主题的 JSON 文档，以及要求 S3 Glacier 向该主题发送通知的事件。

您可以配置文件库为以下文件库事件发布通知：

- **`ArchiveRetrievalCompleted`**— 为档案检索启动的任务完成 (启动作业（POST 作业）) 时，会发生此事件。已完成任务的状态可能为 `Succeeded` 或 `Failed`。发送到 SNS 主题的通知是与从 描述任务（获取任务 ID） 返回的输出相同的输出。
- **`InventoryRetrievalCompleted`**— 为清单检索启动的任务完成 (启动作业（POST 作业）) 时，会发生此事件。已完成任务的状态可能为 `Succeeded` 或 `Failed`。发送到 SNS 主题的通知是与从 描述任务（获取任务 ID） 返回的输出相同的输出。

Amazon SNS 主题必须向文件库授予允许向该主题发布通知的权限。

请求

要设置您文件库中的通知配置，请向文件库的 `notification-configuration` 子资源的 URI 发送设置请求。您可以在请求正文中指定配置。配置包括 Amazon SNS 主题名称以及向每个主题触发通知操作的一系列事件。

语法

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-” (连字符) ，在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

请求正文中的 JSON 包含以下字段。

事件

要求 S3 Glacier 发送通知的一个或多个事件。

有效值：`ArchiveRetrievalCompleted` | `InventoryRetrievalCompleted`

必需：是

类型：数组

SNSTopic

Amazon SNS 主题 ARN。有关更多信息，请参阅《Amazon Simple Notification Service 入门指南》中的 Amazon SNS 入门。

必需：是

类型：字符串

响应

作为响应，Amazon S3 Glacier（S3 Glacier）返回 `204 No Content`（如果接受了通知配置）。

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例展示了如何配置文件库通知。

示例请求

以下请求设置了 `examplevault` 通知配置，以便将两个事件（`ArchiveRetrievalCompleted` 和 `InventoryRetrievalCompleted` ）发送到 Amazon SNS 主题 `arn:aws:sns:us-west-2:012345678901:mytopic`。

```
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"
```

```
}
```

响应示例

成功的响应会返回 204 No Content。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

相关部分

- 获取文件库通知（获取通知配置）
- 删除文件库通知（删除通知配置）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

另请参阅

有关在特定语言的 Amazon 中使用此 API 的更多信息 SDKs，请参阅以下内容：

- AWS Command Line Interface

归档操作

以下是可供在 S3 Glacier 中使用的归档操作。

主题

- 删除档案
- 上传档案（发布档案）

删除档案

描述

此操作会从文件库中删除档案。您可以从文件库一次删除一个档案。要删除档案，您必须在删除请求中提供档案 ID。您可以通过下载包含下载档案的文件库的文件库清单来获取档案 ID。有关下载文件库清单的更多信息，请参阅在 Amazon S3 Glacier 中下载文件库清单。

在删除档案后，您仍可能成功请求启动对已删除档案的检索任务，但档案检索任务会失败。

在您删除档案时，对相应档案 ID 正在进行的档案检索可能成功，也可能不成功，具体取决于下面的场景：

- 如果 Amazon S3 Glacier（S3 Glacier）收到删除档案请求时，档案检索任务正在积极地为下载准备数据，则档案检索操作可能会失败。
- 如果 S3 Glacier 收到删除档案请求时，档案检索任务已成功地为下载准备好档案，则您将能够下载输出。

有关档案检索的更多信息，请参阅“在 S3 Glacier 中下载档案”。

此操作是幂等的。尝试删除已删除的档案不会导致错误。

请求

要删除档案，您需要向档案资源 URI 发送 DELETE 请求。

语法

```
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**
>
> 该AccountId值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例展示了如何从名为 `examplevault` 的文件库删除档案。

示例请求

要删除的档案的 ID 被指定为 `archives` 的子资源。

```
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
```

响应示例

如果请求成功，则 S3 Glacier 会做出 `204 No Content` 的响应，以表示已删除档案。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

相关部分

- 启动分段上传（发布分段上传）
- 上传档案（发布档案）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

上传档案（发布档案）

描述

此操作会向文件库添加档案。对于成功的上传，您的数据会持久保留。作为响应，Amazon S3 Glacier（S3 Glacier）在响应的 `x-amz-archive-id` 标头中返回档案 ID。您应保存返回的档案 ID，以便在以后访问档案。

您必须提供正在上传的数据的 SHA256 树形哈希。有关计算 SHA256 树形哈希的信息，请参见计算校验和。

> **Note**
>
> 使用 API 时，只有上传档案（POST 档案）操作才需要 SHA256 树形哈希。使用时不需要这样做 AWS CLI。

上传档案时，您可以选择指定最多包含 1,024 个可打印 ASCII 字符的档案描述。检索档案或获取文件库清单时，S3 Glacier 会返回档案描述。S3 Glacier 不会以任何方式解读描述。档案描述不需要是唯一的。您不能使用描述来检索档案列表或者对档案列表进行排序。

除了可选的档案描述以外，S3 Glacier 不支持档案的任何附加元数据。档案 ID 是字符的不透明序列，您无法从其推断有关档案的任何含意。因此，您可能会在客户端维护有关档案的元数据。有关更多信息，请参阅 在 Amazon S3 Glacier 中处理档案。

档案是不可变的。上传档案后，您无法编辑档案或其描述。

请求

要上传档案，您可以使用 HTTP POST 方法，并将请求纳入到您要在其中保存档案的文件库的 `archives` 子资源。请求必须包括档案有效负载大小、校验和（SHA256 树形哈希），并且可以选择包括档案的描述。

语法

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作的实施不使用请求参数。

请求标头

除了所有操作通用的请求标头外，此操作还使用以下请求标头。有关通用请求标头的信息，请参阅常见请求标头。

名称	描述	必填
`Content-Length`		是

名称	描述	必填
	数据元的大小（以字节为单位）。欲了解更多信息，请访问 http://www.w3。 org/Protocols/rfc2616/rfc2616-sec14.html #sec14 .13。 类型：数字 默认值：无 限制：无	
`x-amz-archive-description`	您要上传的档案的可选描述。它可以为明语描述或者您选择来分配的某个标识符。描述不需要在所有档案中是唯一的。当您检索文件库清单（请参阅“启动作业（POST 作业）”）时，它会包括因做出响应而返回的每个档案的此描述。 类型：字符串 默认值：无 限制：描述的长度必须小于或等于 1024 个字符。允许的字符为不含控制代码的 7 位 ASCII 字符，明确说来就是 ASCII 值为 32-126（十进制）或 0x20-0x7E（十六进制）的字符。	否
`x-amz-content-sha256`	有效载荷的 SHA256 校验和（线性哈希）。这不是您在 `x-amz-sha256-tree-hash` 标头中指定的相同值。 类型：字符串 默认值：无 限制：无	是

名称	描述	必填
x-amz-sha256-tree-hash	用户计算的有效载荷校验和（ SHA256 树形哈希）。有关计算 SHA256 树形哈希的信息，请参见计算校验和。如果 S3 Glacier 对有效负载进行计算得出不同的有效负载校验和，则会拒绝请求。 类型：字符串 默认值：无 限制：无	是

请求正文

请求正文包含要上传的数据。

响应

作为响应，S3 Glacier 持久地存储档案，并返回档案 ID 的 URI 路径。

语法

```
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
```

响应标头

除了所有操作通用的响应标头外，成功的响应中还包括以下响应标头。有关通用响应标头的信息，请参阅常用响应标头。

名称	描述
Location	新添加的档案资源的相对 URI 路径。

名称	描述
	类型：字符串
x-amz-archive-id	档案的 ID。此值也包括在 Location 标头中，作为该标头的一部分。 类型：字符串
x-amz-sha256-tree-hash	S3 Glacier 计算出的档案校验和。 类型：字符串

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

以下示例显示了上传档案的请求。

```
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).>
```

响应示例

以下成功响应具有 `Location` 标头，您可以从中获取 S3 Glacier 分配给档案的 ID。

```
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
```

相关部分

- 在 Amazon S3 Glacier 中处理档案
- 分段上传大型档案（分段上传）
- 删除档案
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

分段上传操作

以下是可供在 S3 Glacier 中使用的分段上传操作。

主题

- 中止分段上传（删除上传 ID）
- 完成分段上传（发布上传 ID）
- 启动分段上传（发布分段上传）
- 列出段（获取上传 ID）
- 列出分段上传（获取分段上传）
- 上传段（设置上传 ID）

中止分段上传（删除上传 ID）

描述

此分段上传操作命令可以停止上传 ID 标识的分段上传。

中止分段上传请求成功后，您无法使用上传 ID 上传其他任何段或执行其他任何操作。停止已完成的分段上传失败。但是，在短时间内，停止已中止的上传将成功。

此操作是幂等的。

有关分段上传的信息，请参阅“分段上传大型档案（分段上传）”。

请求

要停止分段上传，请将 HTTP `DELETE` 请求发送到文件库的 `multipart-uploads` 子资源的 URI，并将特定分段上传 ID 标识为该 URI 的一部分。

语法

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

在以下示例中，DELETE 请求被发送到分段上传 ID 资源的 URI。

```
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
```

响应示例

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
```

```
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

相关部分

- 启动分段上传（发布分段上传）
- 上传段（设置上传 ID）
- 完成分段上传（发布上传 ID）
- 列出分段上传（获取分段上传）
- 列出段（获取上传 ID）
- 分段上传大型档案（分段上传）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

完成分段上传（发布上传 ID）

描述

您可以调用此分段上传操作来通知 Amazon S3 Glacier（S3 Glacier）所有档案段已上传，使 S3 Glacier 能够用上传的段来拼组档案。

有关分段上传的信息，请参阅“分段上传大型档案（分段上传）”。

拼组档案并将其保存到文件库后，S3 Glacier 返回新创建的档案资源的档案 ID。上传档案后，您应保存返回的档案 ID，以便在以后检索该档案。

在请求中，您必须包括已上传的整个档案的计算 SHA256 树形哈希。有关计算 SHA256 树形哈希的信息，请参见计算校验和。在服务器端，S3 Glacier 还会构造已组装档案的 SHA256 树形哈希。如果值匹配，则 S3 Glacier 会将档案保存到文件库；否则，它返回错误，并且操作会失败。列出段（获取上传 ID）操作可以返回特定分段上传已上传段的列表。它包括每个已上传段的校验和信息，可以用于解决错误的校验和问题。

此外，S3 Glacier 还会检查是否有任何缺失的内容范围。上传段时，您可以指定范围值，用于标识每段在最终档案汇编的位置。拼组最终档案时，S3 Glacier 会检查是否有任何缺失的内容范围；如果有任何缺失的内容范围，则 S3 Glacier 返回错误，并且“完成分段上传”操作会失败。

完成分段上传是一种幂等的操作。第一次成功完成分段上传后，如果您在短时间内再次调用该操作，则该操作将成功并返回相同的档案 ID。这在您遇到网络问题或收到 500 服务器错误时很有用，在这种情况下，您可以重复您的完成分段上传请求并获取相同的档案 ID，而无需创建重复的档案。但是，请注

意，在分段上传完成后，您不能调用列出段操作，并且分段上传将不会出现在列出分段上传响应中，即使可能幂等完成也是如此。

请求

要完成分段上传，您需要将 HTTP POST 请求发送到 S3 Glacier 因响应您的启动分段上传请求而创建的上传 ID 的 URI。这是您在上传段时使用的相同 URI。除了常见的必需标头外，还必须包括整个档案的 SHA256 树形哈希结果以及档案的总大小（以字节为单位）。

语法

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

除了所有操作通用的请求标头外，此操作还使用以下请求标头。有关通用请求标头的信息，请参阅常见请求标头。

名称	描述	必填
`x-amz-archive-size`		是

名称	描述	必填
	整个档案的总大小（以字节为单位）。此值应为您上传的各段的所有大小之和。 类型：字符串 默认值：无 限制：无	
`x-amz-sha256-tree-hash`	整个档案的 SHA256 树形哈希。它是各个部分的树形哈希的 SHA256 树形哈希。如果您在请求中指定的值与 S3 Glacier 计算的最终汇编档案的 SHA256 树形哈希值不匹配，S3 Glacier 将返回错误且请求失败。 类型：字符串 默认值：无 限制：无	是

请求元素

此操作不使用请求要素。

响应

Amazon S3 Glacier（S3 Glacier）会创建整个档案的 SHA256 树形哈希。如果该值与您在请求中指定的整个档案的 SHA256树形哈希值相匹配，则 S3 Glacier 会将该档案添加到文件库中。作为响应，它会返回 HTTP `Location` 标头，其中包括新添加的档案资源的 URL 路径。如果档案大小或您在请求中发送 SHA256 的档案大小不匹配，S3 Glacier 将返回错误，并且上传仍处于未完成状态。稍后可以使用正确值重试完成分段上传操作，此时，您可以成功创建档案。如果分段上传未完成，则 S3 Glacier 最终会收回上传 ID。

语法

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

```
Location: Location
x-amz-archive-id: ArchiveId
```

响应标头

除了所有操作通用的响应标头外，成功的响应中还包括以下响应标头。有关通用响应标头的信息，请参阅常用响应标头。

名称	描述
Location	新创建档案的相对 URI 路径。此 URL 包括 S3 Glacier 生成的档案 ID。 类型：字符串
x-amz-archive-id	档案的 ID。此值也包括在 Location 标头中，作为该标头的一部分。 类型：字符串

响应字段

此操作不返回响应正文。

示例

示例请求

在此示例中，HTTP POST 请求会发送到启动分段上传请求返回的 URI。该请求指定了整个档案的 SHA256 树形哈希值和档案的总大小。

```
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
```

响应示例

以下响应示例显示 S3 Glacier 从您上传的段成功创建了档案。该响应包括具有完整路径的档案 ID。

```
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
```

您现在可以向新添加的资源/档案的 URI 发送 HTTP 请求。例如，您可以发送 GET 请求，以检索档案。

相关部分

- 启动分段上传（发布分段上传）
- 上传段（设置上传 ID）
- 中止分段上传（删除上传 ID）
- 列出分段上传（获取分段上传）
- 列出段（获取上传 ID）
- 分段上传大型档案（分段上传）
- 删除档案
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

启动分段上传（发布分段上传）

描述

此操作启动分段上传（请参阅分段上传大型档案（分段上传））。Amazon S3 Glacier（S3 Glacier）创建了分段上传资源并在响应中返回其 ID。在后续的分段上传操作中，您可以使用此上传 ID。

启动分段上传时，您可以指定段大小（以字节数为单位）。分段大小必须为兆字节 (MiB)（1024 千字节 [KiB]）乘以 2 的幂，例如 1048576(1 MiB)、2097152 (2 MiB)、4194304 (4 MiB)、8388608 (8 MiB)，以此类推。允许的最小段大小为 1 MiB，最大为 4 GiB。

除了最后一段以外，您使用此上传 ID 上传的每一段都必须具有相同的大小。最后一段可以为相同的大小或较小的大小。例如，假设您要上传一个 16.2 MiB 的文件。如果您以 4 MiB 的段大小启动分段上传，则您将首先上传四段（每段 4 MiB），最后再上传一段 (0.2 MiB)。

> **Note**
>
> 启动分段上传时，您不需要知道档案的大小，因为 S3 Glacier 不要求您指定整个档案大小。

完成分段上传后，S3 Glacier 会删除 ID 引用的分段上传资源。如果取消分段上传，S3 Glacier 还将删除分段上传资源；或者，如果在 24 小时内没有执行任何活动，分段上传资源也会被删除。24 小时后，该 ID 可能仍然可用，但是应用程序不应期望此行为。

请求

要启动分段上传，您可以将 HTTP POST 请求发送到要在其中保存档案的文件库的 `multipart-uploads` 子资源 URI。请求必须包括段大小，并且可以选择性地包括档案的描述。

语法

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

除了所有操作通用的请求标头外，此操作还使用以下请求标头。有关通用请求标头的信息，请参阅常见请求标头。

名称	描述	必填
`x-amz-part-size`	除了最后一段以外的每一段的大小（以字节为单位）。最后一段可以小于此段大小。 类型：字符串 默认值：无 约束：分段大小必须为兆字节 （1024 KiB）乘以 2 的幂，例如 1048576(1 MiB)、2097 152 (2 MiB)、4194304 (4 MiB)、8388608 (8 MiB)，以此类推。允许的最小段大小为 1MB，最大为 4 GiB (4096 MiB)。	是
`x-amz-archive-description`	您正在分段上传的档案描述。它可以为明语描述或者您选择来分配的某个唯一的标识符。当您检索文件库清单（请参阅“启动作业（POST 作业）”）时，清单会包括因做出响应而返回的每个档案的此描述。档案描述中的前导空白会被删除。 类型：字符串 默认值：无 约束：描述必须小于或等于 1024 字节。允许的字符为不含控制代码的 7 位 ASCII 字符，明确说来就是 ASCII 值为 32-126（十进制）或 0x20-0x7E（十六进制）的字符。	否

请求正文

此操作没有请求正文。

响应

在响应中，S3 Glacier 会创建由 ID 标识的分段上传资源，并返回分段上传 ID 的相对 URI 路径。

语法

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
x-amz-multipart-upload-id: multiPartUploadId
```

响应标头

除了所有操作通用的响应标头外，成功的响应中还包括以下响应标头。有关通用响应标头的信息，请参阅常用响应标头。

名称	描述
Location	S3 Glacier 创建的分段上传 ID 的相对 URI 路径。您可以使用此 URI 路径来纳入您的请求以上传段，以及完成分段上传。 类型：字符串
x-amz-multipart-upload-id	分段上传的 ID。此值也包括在 Location 标头中，作为该标头的一部分。 类型：字符串

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

以下示例通过向名为 POST 的文件库的 multipart-uploads 子资源 URI 发送 HTTP examplevault 请求来启动分段上传。该请求包括标头，以指定段大小 4 MiB（4194304 字节）和可选的档案描述。

```
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
```

响应示例

S3 Glacier 会创建分段上传资源，并将它添加到文件库的 multipart-uploads 子资源。Location 响应标头包括分段上传 ID 的相对 URI 路径。

```
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
```

有关上传各段的信息，请参阅“上传段（设置上传 ID）”。

相关部分

- 上传段（设置上传 ID）
- 完成分段上传（发布上传 ID）

- 中止分段上传（删除上传 ID）
- 列出分段上传（获取分段上传）
- 列出段（获取上传 ID）
- 删除档案
- 分段上传大型档案（分段上传）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

列出段（获取上传 ID）

描述

此分段上传操作会列出上传 ID 标识的特定分段上传中已上传的档案的段。有关分段上传的信息，请参阅“分段上传大型档案（分段上传）”。

在完成分段上传前，您可以在正在进行的分段上传过程中随时发送此请求。S3 Glacier 会返回段列表，该列表按您在每一段上传中指定的范围排序。如果您在完成分段上传后发送“列出段”请求，则 Amazon S3 Glacier（S3 Glacier）返回错误。

列出段操作支持分页。您应始终检查响应正文中的 `Marker` 字段，以查看是否有继续该列表的标记；如果没有更多项目，则 `marker` 字段为 `null`。如果 `marker` 不为 null，则为了获取下一组段，您可以发送另一个“列出段”请求，并将 `marker` 请求参数设置为 S3 Glacier 为响应您之前的“列出段”请求而返回的标记值。

您还可以通过在请求中指定 `limit` 参数来限制响应中返回的段数。

请求

语法

要列出正在进行的分段上传的段，您可以向分段上传 ID 资源的 URI 发送 `GET` 请求。当您启动分段上传 (启动分段上传（发布分段上传）) 时，系统会返回分段上传 ID。您可以选择性地指定 `marker` 和 `limit` 参数。

```
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**
>
> 该AccountId值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-” (连字符) ，在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

名称	描述	必填
limit	要返回的部分最大数目。默认限制为 50。返回的部分数可能少于指定的限制值，但永远不会超过限制值。 类型：字符串 限制：最小值为 1。最大整数值为 50。	否
marker	用于分页的不透明字符串。marker 指定应从其开始列出段的段。从之前的列出段响应获取 marker 值。只有在您要继续对之前的列出段请求中开始的结果进行分页，您才需要包括 marker。 类型：字符串 限制：无	否

请求标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

请求正文

此操作没有请求正文。

响应

语法

```
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
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

ArchiveDescription

在启动分段上传请求中指定的档案描述。如果在启动分段上传操作中没有指定档案描述，则此字段为 `null`。

类型：字符串

CreationDate

分段上传启动的 UTC 时间。

类型：字符串。以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

Marker

表示从何处继续对结果进行分页的不透明字符串。您可以在新的列出段请求中使用 `marker` 来获取列表中的更多段。如果没有更多段，则此值为 `null`。

类型：字符串

MultipartUploadId

段与其相关联的上传的 ID。

类型：字符串

PartSizeInBytes

段大小（以字节为单位）。这是您在启动分段上传请求中指定的相同值。

类型：数字

部分

分段上传的段大小的列表。数组中的每个数据元均包含 `RangeBytes` 和 `sha256-tree-hash` 名称/值对。

类型：数组

RangeInBytes

段的字节范围，包括范围的上限值。

类型：字符串

SHA256TreeHash

S3 Glacier 为该部分计算的 SHA256 树形哈希值。此字段绝不为 `null`。

类型：字符串

VaultARN

向其启动分段上传的文件库的Amazon 资源名称 (ARN)。

类型：字符串

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例 – 列出分段上传的段

以下示例列出了上传的所有段。该示例向正在进行的分段上传的特定分段上传 ID URI 发送 HTTP GET 请求，并且最多返回 1000 段。

示例请求

```
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
```

响应示例

在响应中，S3 Glacier 返回与指定分段上传 ID 相关联的上传段的列表。在此示例中，只有两段。返回的 `Marker` 字段为 `null`，表示没有更多分段上传的段。

```
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"
}
```

示例 – 列出分段上传的段（指定标记和限制请求参数）

以下示例展示了如何使用分页来获取有限数量的结果。该示例向正在进行的分段上传的特定分段上传 ID URI 发送 HTTP GET 请求，以返回一段。开始 marker 参数指定从哪段开始段列表。您可以从之前的段列表请求的响应获取 marker 值。此外，在此示例中，limit 参数设置为 1，并且返回一段。请注意，Marker 字段不为 null，表示至少还有另一段要获取。

示例请求

```
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
```

响应示例

在响应中，S3 Glacier 返回与正在进行的指定分段上传 ID 相关联的上传段的列表。

```
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"
}
```

相关部分

- 启动分段上传（发布分段上传）
- 上传段（设置上传 ID）
- 完成分段上传（发布上传 ID）
- 中止分段上传（删除上传 ID）
- 列出分段上传（获取分段上传）
- 分段上传大型档案（分段上传）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

列出分段上传（获取分段上传）

描述

此分段上传操作会列出指定文件库中正在进行的分段上传。正在进行的分段上传是 启动分段上传（发布分段上传） 请求启动但尚未完成或停止的分段上传。在列出分段上传响应中返回的列表没有固定的顺序。

列出分段上传操作支持分页。默认情况下，此操作最多会在响应中返回 50 个分段上传。您应始终检查响应正文中的 `marker` 字段，以查看是否有继续该列表的标记；如果没有更多项目，则 `marker` 字段为 `null`。

如果 `marker` 不为 null，则为了获取下一组分段上传，您可以再发送一个“列出分段上传”请求，并将 `marker` 请求参数设置为 Amazon S3 Glacier（S3 Glacier）对上一次“列出分段上传”请求所回复的标记值。

请注意此操作与 列出段（获取上传 ID） 操作之间的区别。列出分段上传操作会列出文件库的所有分段上传。列出段操作会返回上传 ID 标识的特定分段上传的段。

有关分段上传的信息，请参阅“分段上传大型档案（分段上传）”。

请求

语法

要列出分段上传，请向文件库的 GET 子资源的 URI 发送 multipart-uploads 请求。您可以选择性地指定 marker 和 limit 参数。

```
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**
>
> 该AccountId值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

名称	描述	必填
limit	指定响应正文中返回的最大上传数。如果未指定，则列出上传操作最多会返回 50 个上传。 类型：字符串 限制：最小值为 1。最大整数值为 50。	否
marker	用于分页的不透明字符串。marker 指定应从其开始列出上传的上传。从之前的列出上传响应获取 marker 值。只有在您要继续对之前的列出上传请求中开始的结果进行分页，您才需要包括 marker。	否

名称	描述	必填
	类型：字符串 限制：无	

请求标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

请求正文

此操作没有请求正文。

响应

语法

```
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
    },
   ...
  ]
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

ArchiveDescription

在启动分段上传请求中指定的档案描述。如果在启动分段上传操作中没有指定档案描述，则此字段为 `null`。

类型：字符串

CreationDate

分段上传启动的 UTC 时间。

类型：字符串。以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

Marker

表示从何处继续对结果进行分页的不透明字符串。您可以在新的列出分段上传请求中使用 `marker` 来获取列表中的更多上传。如果没有更多上传，则此值为 `null`。

类型：字符串

PartSizeInBytes

在 启动分段上传（发布分段上传） 请求中指定的段大小。这是上传中除了最后一段以外的所有段的大小，最后一段可能小于此大小。

类型：数字

MultipartUploadId

分段上传的 ID。

类型：字符串

UploadsList

有关分段上传数据元的元数据的列表。列表中的每个项目均包含相应上传的一组名称-值对，包括 `ArchiveDescription`、`CreationDate`、`MultipartUploadId`、`PartSizeInBytes` 和 `VaultARN`。

类型：数组

VaultARN

包含档案的文件库的Amazon 资源名称 (ARN)。

类型：字符串

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例：列出所有分段上传

以下示例列出了文件库的正在进行的所有分段上传。该示例显示了发送到指定文件库的 `GET` 子资源 URI 的 HTTP `multipart-uploads` 请求。由于请求中没有指定 `marker` 和 `limit` 参数，因此，系统最多会返回 1000 个正在进行的分段上传。

示例请求

```
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
```

响应示例

在响应中，S3 Glacier 返回指定文件库所有正在进行的分段上传的列表。`marker` 字段为 `null`，表示没有更多上传要列出。

```
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"
    }
  ]
}
```

示例：分段上传的部分列表

以下示例展示了如何使用分页来获取有限数量的结果。该示例显示了发送到指定文件库的 GET 子资源 URI 的 HTTP `multipart-uploads` 请求。在此示例中，`limit` 参数设置为 1，这意味着只在列表中返回一个上传，而 `marker` 参数则表示返回的列表从其开始的分段上传 ID。

示例请求

```
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
```

响应示例

在响应中，Amazon S3 Glacier（S3 Glacier）只返回指定文件库的最多两个正在进行的分段上传，从指定的标记开始返回两个结果。

```
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"
    }
  ]
}
```

相关部分

- 启动分段上传（发布分段上传）
- 上传段（设置上传 ID）
- 完成分段上传（发布上传 ID）
- 中止分段上传（删除上传 ID）
- 列出段（获取上传 ID）
- 分段上传大型档案（分段上传）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

上传段（设置上传 ID）

描述

此分段上传操作会上传档案的一段。您可以按任何顺序上传档案段，因为在您的上传段请求中，您会指定将在此段中上传的汇编档案的字节范围。此外，您还可以并行上传这些段。您最多可以为一个分段上传上传 10 000 段。

有关分段上传的信息，请参阅“分段上传大型档案（分段上传）”。

如果以下任何条件成立，则 Amazon S3 Glacier（S3 Glacier）会拒绝您的上传段请求：

- SHA256 树形哈希不匹配 — 为确保部件数据在传输过程中不会损坏，您需要计算该部分的 SHA256 树形哈希并将其包含在请求中。收到零件数据后，S3 Glacier 还会计算 SHA256 树形哈希。如果这两个哈希值不匹配，则操作会失败。有关计算 SHA256 树形哈希的信息，请参见计算校验和。
- SHA256 线性哈希不匹配 — 授权需要计算整个上传有效负载的 SHA256 线性哈希值并将其包含在请求中。有关计算 SHA256 线性哈希的信息，请参见计算校验和。
- 段大小不匹配 — 除了最后一段以外，每一段的大小必须与相应 启动分段上传（发布分段上传） 请求中指定的大小相匹配。最后一段的大小必须是与指定大小相同的大小或者小于指定大小。

> **Note**
>
> 如果您上传一段，其大小小于您在启动分段上传请求中指定的段大小，并且该段不是最后一段，则上传段请求会成功。但是，后续的完成分段上传请求会失败。

- 范围未对齐 — 请求中的字节范围值未与相应启动请求中指定的段大小对齐。例如，如果您指定 4194304 字节 (4 MB) 的段大小，则 0 到 4194303 字节 (4 MB —1) 以及 4194304 (4 MB) 到 8388607 (8 MB —1) 为有效的段范围。但是，如果您将范围值设置为 2MB 到 6MB，则范围与段大小未对齐，上传将失败。

此操作是幂等的。如果您多次上传相同的段，则包括在最近请求中的数据会覆盖之前上传的数据。

请求

您将该 HTTP PUT 请求发送到启动分段上传请求返回的上传 ID 的 URI。S3 Glacier 使用上传 ID 将段上传与特定的分段上传关联起来。请求必须包括分段数据（`x-amz-SHA256-tree-hash`标头）的 SHA256 树形哈希、整个有效负载（`x-amz-content-sha256`标头）的 SHA256 线性哈希、字节范围（`Content-Range`标头）以及以字节为单位的部分长度（`Content-Length`标头）。

语法

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-” （连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

除了所有操作通用的请求标头外，此操作还使用以下请求标头。有关通用请求标头的信息，请参阅常见请求标头。

名称	描述	必填
`Content-Length`	标识段的长度（以字节为单位）。 类型：字符串 默认值：无 限制：无	否
`Content-Range`		是

名称	描述	必填
	标识将在此段中上传的拼组档案中的字节范围。S3 Glacier 使用此信息按正确的顺序拼组档案。此标头的格式遵循 RFC 2616。示例标头为 `Content-Range:bytes 0-4194303/*` 。 类型：字符串 默认值：无 约束：范围不能大于您在启动分段上传时指定的段大小。	
`x-amz-content-sha256`	上传的有效载荷的 SHA256 校验和（线性哈希）。这不是您在 `x-amz-sha256-tree-hash` 标头中指定的相同值。 类型：字符串 默认值：无 限制：无	是
`x-amz-sha256-tree-hash`	指定正在上传的数据的 SHA256 树形哈希。有关计算 SHA256 树形哈希的信息，请参见计算校验和。 类型：字符串 默认值：无 限制：无	是

请求正文

请求正文包含要上传的数据。

响应

成功上传段后，S3 Glacier 返回 `204 No Content` 响应。

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-sha256-tree-hash: ChecksumComputedByAmazonGlacier
```

响应标头

除了所有操作通用的响应标头外，成功的响应中还包括以下响应标头。有关通用响应标头的信息，请参阅常用响应标头。

名称	描述
x-amz-sha256-tree-hash	S3 Glacier 为上传的部分计算的 SHA256 树形哈希。 类型：字符串

响应正文

此操作不返回响应正文。

示例

以下请求会上传一个 4MB 的段。请求设置了字节范围，使它作为档案中的第一段。

示例请求

示例发送 HTTP PUT 请求，以上传一个 4MB 的段。该请求会发送到启动分段上传请求返回的上传 ID 的 URI。Content-Range 标头将该段标识为档案的第一个 4MB 数据段。

```
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
```

要上传下一部分，步骤相同；但是，您必须计算要上传的段的新SHA256 树形哈希，并指定新的字节范围以指示该部分在最终装配中的位置。以下请求会使用相同的上传 ID 来上传另一段。该请求会指定之前请求后档案的下一个 4MB 以及 4MB 的段大小。

```
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
```

可以按任何顺序上传段；S3 Glacier 会使用每一段的范围说明来确定组装这些段的顺序。

响应示例

```
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
```

相关部分

- 启动分段上传（发布分段上传）
- 上传段（设置上传 ID）
- 完成分段上传（发布上传 ID）
- 中止分段上传（删除上传 ID）
- 列出分段上传（获取分段上传）

- 列出段（获取上传 ID）
- 分段上传大型档案（分段上传）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

任务操作

以下是可供在 S3 Glacier 中使用的任务操作。

主题

- 描述任务（获取任务 ID）
- 获取任务输出（获取输出）
- 启动作业（POST 作业）
- 列出任务（获取任务）

描述任务（获取任务 ID）

描述

此操作返回有关您之前启动的任务的信息，包括任务启动日期、启动任务的用户、任务状态代码/消息，以及要在 Amazon S3 Glacier（S3 Glacier）完成任务后通知的 Amazon Simple Notification Service (Amazon SNS) 主题。有关启动任务的更多信息，请参阅“启动作业（POST 作业）”。

> **Note**
>
> 此操作可让您检查您的任务的状态。但是，我们强烈建议您设置 Amazon SNS 主题并在您的启动任务请求中指定它，这样，S3 Glacier 可以在完成任务后通知该主题。

S3 Glacier 完成任务后，任务 ID 至少在 24 小时内都不会过期。

请求

语法

要获取有关任务的信息，您可以使用 HTTP GET 方法，并将请求纳入到特定任务中。请注意，相对 URI 路径是您在启动任务时 S3 Glacier 返回给您的同一路径。

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

> **Note**
>
> 在该请求中，如果您省略 `JobID`，则响应会返回指定文件库中所有有效任务的列表。有关列出任务的更多信息，请参阅“列出任务（获取任务）”。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

语法

```
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"
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

操作

任务类型。它为 `ArchiveRetrieval`、`InventoryRetrieval` 或 `Select`。

类型：字符串

ArchiveId

为 select 或档案检索任务请求的档案 ID。否则，此字段为 `null`。

类型：字符串

存档SHA256TreeHash

档案检索任务的整个档案的 SHA256 树形哈希。对于清单检索任务，此字段为 `null`。

类型：字符串

ArchiveSizeInBytes

对于 `ArchiveRetrieval` 任务，这是正在请求下载的档案的大小（以字节为单位）。对于 `InventoryRetrieval` 任务，该值为 `null`。

类型：数字

已完成

任务状态。当档案或清单检索任务完成后，您将使用 获取任务输出（获取输出） 来获取任务的输出。

类型：布尔值

CompletionDate

任务请求完成时的通用协调时间 (UTC) 时间。当任务正在进行时，该值将为空。

类型：字符串

CreationDate

创建任务的 UTC 时间。

类型：以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

InventoryRetrievalParameters

用于范围清单检索的输入参数。

类型：InventoryRetrievalJobInput 对象

InventorySizeInBytes

对于 `InventoryRetrieval` 任务，这是请求下载的清单的大小（以字节为单位）。对于 `ArchiveRetrieval` 或 `Select` 任务，该值为 `null`。

类型：数字

JobDescription

您在启动任务时提供的任务描述。

类型：字符串

JobId

标识 S3 Glacier 中的任务的 ID。

类型：字符串

JobOutputPath

包含任务输出位置。

类型：字符串

OutputLocation

一个对象，其中包含有关 select 任务结果和错误的存储位置的信息。

类型：OutputLocation 对象

RetrievalByteRange

档案检索任务的检索字节范围，格式为 “*`StartByteValue-EndByteValue`*。” 如果您未在档案检索中指定范围，则会检索整个档案；也StartByteValue等于 0，EndByteValue等于档案大小减去 1。对于清单检索或 select 任务，此字段为 `null`。

类型：字符串

SelectParameters

一个对象，其中包含有关用于 select 的参数的信息。

类型：SelectParameters 对象

SHA256TreeHash

档案请求范围的 SHA256 树形哈希值。如果档案的 启动作业（POST 作业） 请求指定了以树形哈希对齐的范围，则此字段会返回值。有关档案范围检索的树形哈希对齐的更多信息，请参阅“下载数据时接收校验和”。

对于检索整个档案时的特定情况，此值与 `ArchiveSHA256TreeHash` 值相同。

在以下情况中，此字段为 `null`：

- 指定未以树形哈希对齐的范围的档案检索任务。
- 指定等于整个档案的范围并且任务状态为 `InProgress` 的档案任务。
- 清单任务。
- 选择任务。

类型：字符串

SNSTopic

接收通知的 Amazon SNS 主题。

类型：字符串

StatusCode

指示任务状态的代码。

有效值: `InProgress` | `Succeeded` | `Failed`

类型：字符串

StatusMessage

描述任务状态的友好消息。

类型：字符串

套餐

用于 select 或档案检索的数据访问。

有效值: `Bulk` | `Expedited` | `Standard`

类型：字符串

VaultARN

任务为其子资源的文件库的Amazon 资源名称 (ARN)。

类型：字符串

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例显示了检索档案的任务的请求。

示例请求：获取任务描述

```
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
```

响应示例

响应正文包括描述指定任务的 JSON。请注意，对于清单检索和档案检索任务，JSON 字段相同。但是，当字段不适用于任务类型时，其值为 `null`。以下是档案检索任务的示例响应。请注意以下几点：

- `Action` 字段值为 `ArchiveRetrieval`。
- `ArchiveSizeInBytes` 字段显示了档案检索任务中请求的档案的大小。
- 该`ArchiveSHA256TreeHash`字段显示整个档案的 SHA256 树形哈希。
- `RetrievalByteRange` 字段显示了启动任务请求中请求的范围。在此示例中，请求了整个档案。
- 该`SHA256TreeHash`字段显示了 Initiate Job 请求中请求的范围的 SHA256 树形哈希。在此示例中，它与 `ArchiveSHA256TreeHash` 字段相同，这意味着请求了整个档案。
- `InventorySizeInBytes` 字段值为 `null`，因为它不适用于档案检索任务。

```
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"
}
```

以下是清单检索任务的示例响应。请注意以下几点：

- Action 字段值为 InventoryRetrieval。
- ArchiveSizeInBytes、ArchiveSHA256TreeHash 和 RetrievalByteRange 字段值为空，因为这些字段不适用于清单检索任务。
- InventorySizeInBytes 字段值为 null，因为任务仍在进行，尚未为下载完全准备好清单。如果任务在您的描述任务请求前已完成，则此字段将为您提供输出的大小。

```
{
   "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"
}
```

以下例举了一个针对已完成清单检索任务的响应，该任务包含用于继续文件库清单检索分页的标记。

```
{
    "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
    },
}
```

相关部分

- 获取任务输出（获取输出）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

获取任务输出（获取输出）

描述

此操作会下载您使用 启动作业（POST 作业） 启动的任务的输出。根据您在启动任务时指定的任务类型，输出将为档案的内容或文件库清单。

您可以下载所有任务输出，也可以通过指定字节范围下载输出的一部分。对于档案和清单检索任务，您应针对在 Get Job Output 响应的标头中返回的大小对下载大小进行验证。

对于档案检索任务，还应验证大小是否与预期相符。如果下载部分输出，则预期大小基于您指定的字节范围。例如，如果指定 `bytes=0-1048575` 范围，则需要验证下载大小为 1048576 字节。如果下载整个档案，则预期大小是您上传到 Amazon S3 Glacier（S3 Glacier）的档案的大小。Get Job Output 响应的标头中也会返回预期大小。

对于档案检索任务的情况，根据您指定的字节范围，S3 Glacier 返回该部分数据的校验和。要确保您下载的部分是正确的数据，请在客户端计算校验和，验证值是否匹配，并且验证大小是否和预期一致。

S3 Glacier 完成任务后，任务 ID 至少在 24 小时内都不会过期。也就是说，您可以在 S3 Glacier 完成任务后的 24 小时期限内下载任务输出。

请求

语法

要检索任务输出，您可以向特定任务的 `GET` 的 URI 发送 HTTP `output` 请求。

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

除了所有操作通用的请求标头外，此操作还使用以下请求标头。有关通用请求标头的信息，请参阅常见请求标头。

名称	描述	必填
`Range`	要从输出检索的字节范围。例如，如果您要下载前 1048576 字节，请指定范围 `bytes=0-1048575` 。有关更多信息，请转到“范围标头字段定义”。该范围与启动任务请求中指定的任何范围是相对而言的。默认情况下，此操作会下载整个输出。 如果任务输出很大，则您可以使用 `Range` 请求标头来检索输出的一部分。这样，您能够以较小的字节区块下载整个输出。例如，假设您有 1GB 任务输出需要下载，您决定一次下载 128MB 数据区块，则总共有八个获取任务输出请求。您将使用以下流程下载任务输出： 1. 通过使用 `Range` 标头指定相应的字节范围来下载 128MB 输出区块。验证是否收到所有 128MB 数据。 2. 随数据一起，响应将包括有效负载的校验和。您可以计算客户端有效负载的校验和，并将它与您在响应中接收到的校验和相比较，以确保您收到了所有预期的数据。 3. 对输出数据的所有八个 128MB 区块重复执行步骤 1 和 2，每次都指定相应的字节范围。 4. 下载任务输出的所有段后，您会拥有八个校验和值的列表。计算这些值的树形哈希，以得出整个输出的校验和。使用 描	否

名称	描述	必填
	述任务（获取任务 ID）操作，获取为您提供了输出的任务的任务信息。响应包括 S3 Glacier 中存储的整个档案的校验和。您可以将此值与您计算的校验和相比较，以确保您正确地下载了整个档案内容。 类型：字符串 默认值：无 限制：无	

请求正文

此操作没有请求正文。

响应

语法

对于返回所有任务数据的检索请求，任务输出响应会返回 `200 OK` 响应代码。当请求部分内容（例如，如果您在请求中指定了 `Range` 标头）时，则会返回响应代码 `206 Partial Content`。

```
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.]
```

响应标头

标题	描述
`Content-Range`	

标题	描述
	S3 Glacier 返回的字节范围。如果只下载部分输出，则响应会提供 S3 Glacier 返回的字节范围。 例如，`bytes 0-1048575/8388608` 会从 8MB 返回前 1MB。 有关 `Content-Range` 标头的更多信息，请转到“内容范围标头字段定义”。 类型：字符串
`Content-Type`	内容类型取决于任务输出是档案还是文件库清单。 • 对于档案数据，内容类型为 `application/octet-stream` 。 • 对于文件库清单，如果您在启动任务时请求了 CSV 格式，则内容类型为 `text/csv`。否则，默认情况下，以 JSON 返回文件库清单，并且内容类型为 `application/json` 。 类型：字符串

标题	描述
`x-amz-sha256-tree-hash`	响应中的数据的校验和。只有在检索档案检索任务的输出时，才会返回此标头。此外，如果启动任务请求中请求的检索数据范围以树形哈希对齐，并且获取任务输出中要下载的范围也以树形哈希对齐，则此标头会显示。有关以树形哈希对齐的范围的更多信息，请参阅“下载数据时接收校验和”。 例如，如果您在您的启动任务请求中指定了要检索的以树形哈希对齐的范围（包括整个档案），则您将收到您在以下条件下下载的数据的校验和： • 您获取了检索数据的整个范围。 • 您请求了检索数据的字节范围，该数据的大小为 1 兆字节 (1024KB) 乘以 2 的幂，并且其起始位置和结束位置为请求范围大小的倍数。例如，如果您有 3.1MB 的检索数据，并且您指定了要返回的范围（该范围起始于 1MB，结束于 2MB），则 `x-amz-sha256-tree-hash` 会作为响应标头返回。 • 您请求了检索数据中要返回的范围（该范围的结束位置为数据的结束位置），该范围的起始位置为要检索的范围向上舍入到下一个 2 的幂但不小于 1 兆字节 (1024KB) 的大小的倍数。例如，如果您有 3.1MB 的检索数据，并且您指定了范围（该范围起始于 2MB，结束于数据的结束位置 3.1MB），则 `x-amz-sha256-tree-hash` 会作为响应标头返回。 类型：字符串

响应正文

S3 Glacier 会在响应正文中返回任务输出。根据任务类型，输出可以为档案内容或文件库库存。如果为文件库清单，则默认情况下，清单列表会作为以下 JSON 正文返回。

```
{
```

```
 "VaultARN": String,
 "InventoryDate": String,
 "ArchiveList": [
      {"ArchiveId": String,
       "ArchiveDescription": String,
       "CreationDate": String,
       "Size": Number,
       "SHA256TreeHash": String
      },
      ...
    ]
}
```

如果您在启动文件库清单任务时请求了 CSV 格式，则文件库清单会以 CSV 格式返回在正文中。CSV 格式有五列 ArchiveId “”、“”、ArchiveDescription “”、CreationDate “大小” 和 “”，其定义与相应的 JSON 字段相同。SHA256 TreeHash

> **Note**
>
> 在返回的 CSV 格式中，返回的字段可能整个字段用双引号括起来。包含逗号或双引号的字段在返回时始终用双引号括起来。例如，`my archive description,1` 返回为 `"my archive description,1"`。用双引号括起来的返回字段中的双引号字符通过在前面附加反斜杠字符来转义。例如，`my archive description,1"2` 返回为 `"my archive description,1\"2"`，`my archive description,1\"2` 返回为 `"my archive description,1\\"2"`。反斜杠字符不进行转义。

JSON 响应正文包含以下 JSON 字段。

ArchiveDescription

档案的描述。

类型：字符串

ArchiveId

档案的 ID。

类型：字符串

ArchiveList

档案元数据数组。数组中的每个数据元均表示文件库中包含的一个档案的元数据。

类型：数组

CreationDate

创建档案的 UTC 日期和时间。

类型：以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

InventoryDate

对文件库进行更改后完成文件库上次库存盘点的 UTC 日期和时间。即使 S3 Glacier 每天准备一次文件库清单，清单日期也不会随时更新；只有在上次清单盘点后对文件库执行过添加或删除档案的操作时，清单日期才会更新。

类型：以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

SHA256TreeHash

档案的树形哈希。

类型：字符串

大小

档案的大小（以字节为单位）。

类型：数字

VaultARN

从中请求档案检索的Amazon 资源名称 (ARN) 资源。

类型：字符串

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例显示了检索档案的任务的请求。

示例 1：下载输出

此示例会检索 S3 Glacier 因响应您的启动档案检索任务请求而准备的数据。

示例请求

```
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
```

响应示例

以下是档案检索任务的示例响应。请注意，`Content-Type` 标头为 `application/octet-stream`，并且 `x-amz-sha256-tree-hash` 标头包括在响应中，这意味着返回所有任务数据。

```
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.]
```

以下是清单检索任务的示例响应。请注意，`Content-Type` 标头为 `application/json`。另请注意，响应不包括 `x-amz-sha256-tree-hash` 标头。

```
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"
   }
  ]
}
```

示例 2：只下载部分输出

此示例只检索 S3 Glacier 因响应您的启动档案检索任务请求而准备的档案的一部分。该请求使用了可选的 Range 标头，以便只检索前 1024 字节。

示例请求

```
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
```

响应示例

以下成功的响应显示了 `206 Partial Content` 响应。在此案例中，响应还包括 `Content-Range` 标头，该标头指定了 S3 Glacier 返回的字节范围。

```
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.]
```

相关部分

- 描述任务（获取任务 ID）
- 启动作业（POST 作业）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

启动作业（POST 作业）

此操作将启动以下类型的 Amazon S3 Glacier（S3 Glacier）任务：

- `archive-retrieval` — 检索档案
- `inventory-retrieval` — 清点文件库

主题

- 初始化档案或文件库清单检索任务
- 请求
- 响应
- 示例
- 相关部分

初始化档案或文件库清单检索任务

检索档案或文件库清单是异步操作，这些操作要求您启动任务。一旦启动，就无法取消作业。此检索包括两个步骤：

1. 使用 启动作业（POST 作业） 操作启动检索任务。

> **Important**
>
> 数据检索策略可能导致您启动检索任务的请求以发生 `PolicyEnforcedException` 而失败。有关数据检索策略的更多信息，请参阅S3 Glacier 数据检索策略。有关 `PolicyEnforcedException` 异常的更多信息，请参阅错误响应。

2. 在任务完成后，使用 获取任务输出（获取输出） 操作下载字节。

检索请求会异步运行。当您启动检索任务时，S3 Glacier 会创建任务并在响应中返回任务 ID。S3 Glacier 完成任务时，您可以获取任务输出（档案或清单数据）。有关获取任务输出的信息，请参阅“获取任务输出（获取输出）”操作。

任务必须先完成，然后，您才能获取其输出。要确定任务何时完成，您有以下选择：

- 使用 Amazon SNS 通知 — 您可以指定一个 Amazon SNS 主题，S3 Glacier 会在任务完成后向该主题发布通知。您可以为每个任务请求指定 SNS 主题。只有在 S3 Glacier 完成任务后，系统才会发送通知。除了为每个任务请求指定 SNS 主题以外，您还可以配置文件库的文件库通知，这样，系统就会为所有检索操作发送任务通知。有关更多信息，请参阅 设置文件库通知配置（设置通知配置）。
- 获取任务详细信息 — 当任务正在进行时，您可以发送 描述任务（获取任务 ID） 请求以获取任务状态信息。但是，更有效的方法是使用 Amazon SNS 通知来确定任务何时完成。

> **Note**
>
> 您通过通知获取的信息与通过调用 描述任务（获取任务 ID） 获取的信息相同。

对于特定事件，如果您在文件库中添加了两种通知配置，并且也在您的启动任务请求中指定了 SNS 主题，则 S3 Glacier 会发送这两种通知。有关更多信息，请参阅 设置文件库通知配置（设置通知配置）。

文件库清单

从您第一次将档案上传到文件库的日期开始，S3 Glacier 大约每天都会更新一次文件库清单。如果在上次清单盘点后没有对文件库执行过添加或删除档案的操作，则不会更新库存日期。当您启动文件库清单任务时，S3 Glacier 会返回其生成的最后一个清单，即 point-in-time快照而不是实时数据。

S3 Glacier 为文件库创建第一份清单后，通常需要经过半天（最多一天）时间，该清单才可用于检索操作。

您可能没有发现为每个档案上传操作检索文件库清单有什么好处。但是，假设您在客户端维护数据库，且该客户端关联了您上传到 S3 Glacier 的档案的元数据。此时，您可能会发现，文件库库存对于根据需要将您数据库中的信息与实际文件库库存进行协调很有用。有关清单任务输出中返回的数据字段的更多信息，请参阅响应正文。

确定清单检索范围

您可以通过筛选档案创建日期或设置限制，来限制检索的清单项目数。

按档案创建日期筛选

通过在启动任务请求中为这些参数指定值，您可以检索在 `StartDate` 和 `EndDate` 之间创建的档案的清单项目。将会返回在 `StartDate` 之后且 `EndDate` 之前创建的档案。如果您仅提供 `StartDate`，而不提供 `EndDate`，则会检索在 `StartDate` 或之后创建的所有的档案的清单。如果您仅提供 `EndDate`，而不提供 `StartDate`，则会检索在 `EndDate` 之前创建的所有档案的清单。

限制每次检索的清单项目

通过在启动任务请求中设置 `Limit` 参数，可以限制返回的清单项目数量。清单任务输出包含的清单项目数最多为指定的 `Limit`。如果有更多清单项目可用，则结果会分页。任务完成之后，您可以使用 描述任务（获取任务 ID） 操作获取在后续启动任务请求中使用的标记。该标记将指示检索下一组清单项目的起点。通过使用之前的描述任务输出中的标记反复提出启动任务请求，可以浏览整个清单。如此操作，直至从描述任务获取一个返回 null 的标记 (这指示无更多清单项目可用)。

您可以将 `Limit` 参数与日期范围参数一起使用。

关于具有范围的档案检索

您可以为整个档案或某个范围的档案启动档案检索操作。对于具有范围的档案检索操作情况，您可以指定要返回的字节范围或整个档案。指定的范围必须以兆字节 (MB) 对齐。换言之，范围起始值必须可被 1MB 整除，并且范围结束值加 1 必须可被 1MB 整除或者等于档案的结束值。如果具有范围的档案检索操作没有以兆字节对齐，则此操作会返回 `400` 响应。此外，为了确保您获取您使用获取任务输出 (获

取任务输出（获取输出）) 下载的数据的校验和值，范围必须以树形哈希对齐。有关以树形哈希对齐的范围的更多信息，请参阅“下载数据时接收校验和”。

加速、标准和批量层

在启动档案检索任务时，您可以在请求正文的 `Tier` 字段中指定以下选项之一：

- **`Expedited`** – 加速层允许您在偶尔需要紧急请求还原档案时快速访问数据。对于除了最大型档案 (250 MB+) 之外的所有其他档案，使用加速层访问的数据通常在 1 到 5 分钟内可用。
- **`Standard`** – 标准层允许您在数小时内访问您的任意档案。使用标准层访问的数据通常在 3–5 小时内可用。此选项是未指定层选项的任务请求的默认选项。
- **`Bulk`** – 批量层是 S3 Glacier 的最低成本层，使您可以在一天内以较低的成本检索大量（甚至是 PB 级）的数据。使用批量层访问的数据通常在 5–12 小时内可用。

有关加速和批量检索的更多信息，请参阅检索 S3 冰川档案。

请求

要启动任务，您可以使用 HTTP `POST` 方法，并将请求纳入到文件库的 `jobs` 子资源中。您可以在您请求的 JSON 文档中指定任务请求的详细信息。任务类型是通过 `Type` 字段指定的。（可选）您可以指定 `SNSTopic` 字段来表示 S3 Glacier 在完成任务后可以向其发布通知的 Amazon SNS 主题。

> **Note**
>
> 要向 Amazon SNS 发布通知，您必须自己创建主题（如果主题不存在）。S3 Glacier 不会为您创建主题。该主题必须具有从 S3 Glacier 文件库接收出版物的权限。S3 Glacier 不会验证文件库是否有权向该主题发布内容。如果没有适当配置权限，则即使任务完成后，您可能也不会收到通知。

语法

以下是用于启动任务的请求语法。

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求正文

请求接受请求正文中采用 JSON 格式的以下数据。

jobParameters

提供用于指定任务信息的选项。

类型：jobParameters 对象

必需：是

响应

S3 Glacier 创建了任务。在响应中，它会返回任务的 URI。

语法

```
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
```

响应标头

标题	描述
Location	任务的相对 URI 路径。您可以使用此 URI 路径来查找任务状态。有关更多信息，请参阅 描述任务（获取任务 ID）。 类型：字符串 默认值：无
x-amz-job-id	任务的 ID。此值也包括在 Location 标头中，作为该标头的一部分。 类型：字符串 默认值：无
x-amz-job-output-path	存储 Select 结果的位置的路径。

标题	描述
	类型：字符串 默认值：无

响应正文

此操作不返回响应正文。

错误

除了所有 Amazon S3 Glacier 操作中常见的可能错误外，此操作还包括以下一个或多个错误。有关 Amazon S3 Glacier 错误的信息以及错误代码列表，请参阅错误响应。

代码	描述	HTTP 状态代码	类型
`InsufficientCapacityException`	如果没有足够的容量处理此加速请求，则返回此代码。此错误仅适用于加速检索，不适用于标准或批量检索。	`503 Service Unavailable`	服务器

示例

示例请求：启动档案检索任务

```
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"
}
```

以下是请求正文的示例，它使用 RetrievalByteRange 字段指定了要检索的档案范围。

```
{
  "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"
}
```

响应示例

```
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
```

示例请求：启动清单检索任务

以下请求会启动清单检索任务，以从 examplevault 文件库获取档案列表。在请求正文中，设置为 Format 的 CSV 表示清单会以 CSV 格式返回。

```
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"
}
```

响应示例

```
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
```

示例请求：使用日期筛选及设置限制来启动清单检索任务，以及检索下一页清单项目的后续请求。

以下请求通过使用日期筛选及设置限制来启动文件库清单检索任务。

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

以下例举了使用从描述任务（获取任务 ID）获取的标记检索下一页清单项目的后续请求。

```
{
    "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
    },
}
```

响应示例

```
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/
```

相关部分

- 描述任务（获取任务 ID）
- 获取任务输出（获取输出）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

列出任务（获取任务）

描述

此操作会列出文件库的任务，包括正在进行的任务以及最近完成的任务。

> **Note**
>
> Amazon S3 Glacier（S3 Glacier）在删除最近完成的任务前会将这些任务保留一段时间；但是，它最终会删除已完成的任务。您可以检索已完成任务的输出。在任务完成后，通过将完成的任务保留一段时间，您可以在错过任务完成通知或者首次尝试下载失败的情况下获取任务输出。例如，假设您启动档案检索任务以下载档案。任务完成后，您开始下载档案，但是遇到了网络错误。在这种情况下，您可以在任务存在时重试并下载档案。

`List Jobs` 操作支持分页。您应该始终查看响应 `Marker` 字段。如果没有更多列出的任务，则 `Marker` 字段将设置为 `null`。如果还有更多列出的任务，则 `Marker` 字段将设置为非 null 值，您可使用该值继续对列表分页。要返回从特定任务开始的任务列表，请将 `marker` 请求参数设置为您从之前的 `Marker` 请求获取的该任务的 `List Jobs` 值。

您也可以通过在请求中指定 `limit` 参数来设置响应中返回的最大作业数限制值。默认限制为 50。返回的作业数可能少于限制值，但永远不会超过限制值。

此外，您还可以通过指定可选的 `statuscode` 参数和/或 `completed` 参数来筛选返回的任务列表。使用 `statuscode` 参数，您可以指定只返回与 `InProgress`、`Succeeded` 或 `Failed` 状态匹配的任务。使用 `completed` 参数，您可以指定只返回已完成 (`true`) 的任务或未完成 (`false`) 的任务。

请求

语法

要返回所有类型的任务列表，请向文件库的 `GET` 子资源的 URI 发送 `jobs` 请求。

```
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**
>
> 该`AccountId`值是拥有文件库的账户的 AWS 账户 ID。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您使用账户 ID，请勿在 ID 中包含任何连字符 ('-')。

请求参数

名称	描述	必填
completed	要返回的任务的状态。您可以指定 true 或 false。 类型：布尔值 限制：无	否
limit	要返回的作业最大数目。默认限制为 50。返回的作业数可能少于指定的限制值，但永远不会超过限制值。 类型：字符串 限制：最小整数值为 1。最大整数值为 50。	否
marker	用于分页的不透明字符串可指定应从其开始列出任务的任务。您可从之前的 marker 响应获取 List Jobs 值。只有在您要继续对之前的 List Jobs 请求中开始的结果进行分页时，才需要包括 marker。 类型：字符串 限制：无	否
statuscode	要返回的任务状态的类型。 类型：字符串 限制：InProgress 、Succeeded 或 Failed 这三个值之一。	否

请求标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

请求正文

此操作没有请求正文。

响应

语法

```
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"
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

JobList

任务对象的列表。每个任务对象包含描述任务的元数据。

类型：GlacierJobDescription 对象数组

Marker

表示从何处继续对结果进行分页的不透明字符串。您可以在新的 `marker` 请求中使用 `List Jobs` 值来获取列表中的更多任务。如果没有列出更多任务，则此值为 `null`。

类型：字符串

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例展示了如何返回有关文件库任务的信息。第一个示例返回两个任务的列表，第二个示例返回各项任务的子集。

示例：返回所有任务

示例请求

以下 GET 请求可为文件库返回任务。

```
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
```

响应示例

以下响应包括一个档案检索任务和一个清单检索任务，后者包含用于继续对文件库清单检索分页的标记。响应同时还可显示 Marker 字段已设定为 null，即表示没有更多列出任务。

```
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
}
```

示例：返回任务的部分列表

示例请求

以下 GET 请求返回 marker 参数指定的任务。如果将 limit 参数设置为 ，则指定最多返回两个任务。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
```

响应示例

以下响应显示了两项返回任务，Marker 字段已设置为非 null 值，可用于对任务列表继续分页。

```
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"
}
```

相关部分

- 描述任务（获取任务 ID）
- 适用于 Amazon S3 Glacier 的 Identity and Access Management

在任务操作中使用的数据类型

以下是在 S3 Glacier 中和任务操作一起使用的数据类型。

主题

- CSVInput
- CSVOutput
- 加密
- GlacierJobDescription
- 授权
- 被授权者
- InputSerialization
- InventoryRetrievalJobInput
- jobParameters
- OutputLocation
- OutputSerialization
- S3Location
- SelectParameters

CSVInput

有关包含逗号分隔值 (CSV) 文件的信息。

内容

注释

单个字符，用于指示在行的开头出现该字符时应忽略该行。

类型：字符串

必需：否

FieldDelimiter

单个字符，用于分隔记录中的各个字段。该字符必须是 \n、\r 或范围 32–126 中的 ASCII 字符。默认值为逗号 (,)。

类型：字符串

默认值：,

必需：否

FileHeaderInfo

一个值，描述如何处理输入中的第一行。

类型：字符串

有效值: `Use` | `Ignore` | `None`

必需：否

QuoteCharacter

用作转义字符的单个字符，其中字段分隔符是值的一部分。

类型：字符串

必需：否

QuoteEscapeCharacter

单个字符，用于对已转义值内的引号字符进行转义。

类型：字符串

必需：否

RecordDelimiter

单个字符，用于分隔各个记录。

类型：字符串

必需：否

更多信息

- 启动作业（POST 作业）

CSVOutput

包含有关存储作业结果的逗号分隔值 (CSV) 格式的信息。

内容

FieldDelimiter

单个字符，用于分隔记录中的各个字段。

类型：字符串

必需：否

QuoteCharacter

用作转义字符的单个字符，其中字段分隔符是值的一部分。

类型：字符串

必需：否

QuoteEscapeCharacter

单个字符，用于对已转义值内的引号字符进行转义。

类型：字符串

必需：否

QuoteFields

一个值，指示是否所有输出字段都应包含在引号中。

有效值：`ALWAYS | ASNEEDED`

类型：字符串

必需：否

RecordDelimiter

单个字符，用于分隔各个记录。

类型：字符串

必需：否

更多信息

- 启动作业（POST 作业）

加密

包含有关用于在 Amazon S3 中存储任务结果的加密的信息。

内容

加密

在 Amazon S3 中存储任务结果时使用的服务器端加密算法。默认值为无加密。

类型：字符串

有效值：`aws:kms` | `AES256`

必需：否

KMSContext

可选。如果加密类型是 `aws:kms`，，您可以使用此值为任务结果指定加密上下文。

类型：字符串

必需：否

KMSKeyId

用于对象加密的 AWS Key Management Service (AWS KMS) 密钥 ID。

类型：字符串

必需：否

更多信息

- 启动作业（POST 作业）

GlacierJobDescription

包含 Amazon S3 Glacier（S3 Glacier）任务的描述。

内容

操作

任务类型。它为 `ArchiveRetrieval`、`InventoryRetrieval` 或 `Select`。

类型：字符串

ArchiveId

为 select 或档案检索任务请求的档案 ID。否则，此字段为 `null`。

类型：字符串

存档SHA256TreeHash

用于存档检索的整个档案的 SHA256 树形哈希。对于清单检索任务，此字段为 `null`。

类型：字符串

ArchiveSizeInBytes

对于 `ArchiveRetrieval` 任务，这是正在请求下载的档案的大小（以字节为单位）。对于 `InventoryRetrieval` 任务，该值为 `null`。

类型：数字

已完成

如果任务已完成，则为 `true`；否则为 `false`。

类型：布尔值

CompletionDate

任务完成的日期。

任务请求完成时的通用协调时间 (UTC) 时间。当任务正在进行时，该值将为空。

类型：以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

CreationDate

任务启动时的通用协调时间 (UTC) 日期。

类型：以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

InventoryRetrievalParameters

用于范围清单检索的输入参数。

类型：InventoryRetrievalJobInput 对象

InventorySizeInBytes

对于 `InventoryRetrieval` 任务，这是请求下载的清单的大小（以字节为单位）。对于 `ArchiveRetrieval` 或 `Select` 任务，该值为 `null`。

类型：数字

JobDescription

您在启动任务时提供的任务描述。

类型：字符串

JobId

标识 S3 Glacier 中的任务的 ID。

类型：字符串

JobOutputPath

包含任务输出位置。

类型：字符串

OutputLocation

一个对象，其中包含有关 select 任务结果和错误的存储位置的信息。

类型：OutputLocation 对象

RetrievalByteRange

档案检索任务的检索字节范围，格式为 “*`StartByteValue-EndByteValue`*。” 如果在档案检索中未指定范围，则会检索整个存档，其StartByteValue值等于 0，EndByteValue等于档案大小减去 1。对于清单检索任务，此字段为 `null`。

类型：字符串

SelectParameters

一个对象，其中包含有关用于 select 的参数的信息。

类型：SelectParameters 对象

SHA256TreeHash

档案请求范围的 SHA256 树形哈希值。如果档案的 启动作业（POST 作业） 请求指定了以树形哈希对齐的范围，则此字段会返回值。有关档案范围检索的树形哈希对齐的更多信息，请参阅“下载数据时接收校验和”。

对于检索整个档案时的特定情况，此值与 `ArchiveSHA256TreeHash` 值相同。

在以下情况中，此字段为 `null`：

- 指定未以树形哈希对齐的范围的档案检索任务。
- 指定等于整个档案的范围并且任务状态为 `InProgress` 的档案任务。
- 清单任务。
- 选择任务。

类型：字符串

SNSTopic

表示 Amazon SNS 主题的 Amazon 资源名称（ARN）；如果在任务启动 (启动作业（POST 作业）) 中配置了通知，则系统会向该主题发送任务完成或失败的通知。

类型：字符串

StatusCode

指示任务状态的代码。

有效值: `InProgress` | `Succeeded` | `Failed`

类型：字符串

StatusMessage

任务状态消息。

类型：字符串

套餐

用于 select 或档案检索的数据访问。

有效值: `Expedited` | `Standard` | `Bulk`

类型：字符串

VaultARN

任务为其子资源的文件库的 ARN。

类型：字符串

更多信息

- 启动作业（POST 作业）

授权

包含有关授权的信息。

内容

被授权者

被授权者。

类型：被授权者 对象

必需：否

许可

授予被授权者的权限。

类型：字符串

有效值: `FULL_CONTROL` | `WRITE` | `WRITE_ACP` | `READ` | `READ_ACP`

必需：否

更多信息

- 启动作业（POST 作业）

被授权者

包含有关被授权者的信息。

内容

DisplayName

被授权者的屏幕名称。

类型：字符串

必需：否

EmailAddress

被授权者的电子邮件地址。

类型：字符串

必需：否

ID

被授权者的规范用户 ID。

类型：字符串

必需：否

类型

被授权者的类型。

类型：字符串

有效值: `AmazonCustomerByEmail` | `CanonicalUser` | `Group`

必需：否

URI

被授权者组的 URI。

类型：字符串

必需：否

更多信息

- 启动作业（POST 作业）

InputSerialization

描述如何序列化档案。

内容

CSV

一个对象，用于描述 CSV 编码对象的序列化。

类型：CSVInput 对象

必需：否

更多信息

- 启动作业（POST 作业）

InventoryRetrievalJobInput

提供用于指定范围清单检索任务的选项。

内容

EndDate

文件库清单检索的结束日期 (采用 UTC 时间格式)，在此日期之前创建的存档都包括在检索范围内。

有效值：以 ISO 8601 日期格式 (`YYYY-MM-DDThh:mm:ssTZD`) 表示的字符串，以秒为单位，例如 `2013-03-20T17:03:43Z`。

类型：字符串。以 ISO 8601 日期格式 (YYYY-MM-DDThh:mm:ssTZD) 表示的字符串，以秒为单位，例如 2013-03-20T17:03:43Z。

必需：否

格式

文件库清单列表的输出格式，在启动任务以检索文件库清单时通过 启动作业（POST 作业） 请求进行设置。

有效值：CSV | JSON

必需：否

类型：字符串

限制

每个文件库清单检索请求可以返回的最大清单项目数。

有效值：大于或等于 1 的整数值。

类型：字符串

必需：否

Marker

表示从何处继续对结果进行分页的不透明字符串。您可以在新 Initiate Job 请求中使用此标记获取其他清单项目。如果没有更多清单项目，则此值为 null。

类型：字符串

必需：否

StartDate

文件库清单检索的开始日期 (采用 UTC 时间格式)，包含当日或之后创建的档案。

有效值：以 ISO 8601 日期格式 (YYYY-MM-DDThh:mm:ssTZD) 表示的字符串，以秒为单位，例如 2013-03-20T17:03:43Z。

类型：字符串。以 ISO 8601 日期格式 (YYYY-MM-DDThh:mm:ssTZD) 表示的字符串，以秒为单位，例如 2013-03-20T17:03:43Z。

必需：否

更多信息

- 启动作业（POST 作业）

jobParameters

提供用于定义任务的选项。

内容

ArchiveId

您需要的档案的 ID。如果 Type 字段设置为 `select` 或 `archive-retrieval`，则此字段是必需的。如果您为清单检索任务请求指定此字段，则会出现错误。

有效值：必须为您从之前发送到 Amazon S3 Glacier（S3 Glacier）的请求获取的有效档案 ID。

类型：字符串

必需：如果 Type 设置为 `select` 或 `archive-retrieval`，则是必需的。

描述

任务的可选描述。

有效值：描述的长度必须小于或等于 1024 字节。允许的字符为不含控制代码的 7 位 ASCII 字符，明确说来就是 ASCII 值为 32-126（十进制）或 0x20-0x7E（十六进制）的字符。

类型：字符串

必需：否

格式

(可选) 在启动任务以检索文件库清单时的输出格式。如果您要启动清单任务，并且不指定 `Format` 字段，则 JSON 为默认格式。

有效值：`CSV | JSON`

类型：字符串

必需：否

InventoryRetrievalParameters

用于范围清单检索的输入参数。

类型：InventoryRetrievalJobInput 对象

必需：否

OutputLocation

一个对象，其中包含有关 select 任务结果的存储位置的信息。

类型：OutputLocation 对象

必需：对于 `select` 任务是必需的。

RetrievalByteRange

要检索的字节范围`archive-retrieval`，格式为 “*`StartByteValue-EndByteValue`*”。如果不指定此字段，则会检索整个档案。如果指定此字段，则字节范围必须以兆字节 (1024*1024) 对齐。兆字节对齐意味着StartByteValue必须可以被 1 MB 整除，而EndByteValue加 1 必须可以被 1 MB 整除，或者是存档的结尾，指定为存档字节大小值减去 1。如果RetrievalByteRange不是以兆字节对齐的，则此操作将返回响应。`400`

如果您为 `inventory-retrieval` 或 `select` 任务请求指定此字段，则会出现错误。

类型：字符串

必需：否

SelectParameters

一个对象，其中包含有关用于 select 的参数的信息。

类型：SelectParameters 对象

必需：否

SNSTopic

任务完成并且输出已准备好供您下载时 S3 Glacier 向其发送通知的 Amazon SNS 主题的 Amazon 资源名称（ARN）。指定的主题会向其订阅者发布通知。

SNS 主题必须存在。如果不存在，则 S3 Glacier 不会为您创建该主题。此外，SNS 主题必须拥有允许创建任务的账户向主题发布消息的策略。有关 SNS 主题名称的信息，请参阅CreateTopic《亚马逊简单通知服务 API 参考》。

类型：字符串

必需：否

套餐

用于 select 或档案检索任务的层。`Standard` 是默认使用的值。

有效值: `Expedited` | `Standard` | `Bulk`

类型：字符串

必需：否

类型

任务类型。您可以启动 作业以对档案执行 select 查询、检索档案或获取文件库的清单。

有效值: `select` | `archive-retrieval` | `inventory-retrieval`

类型：字符串

必需：是

更多信息

- 启动作业（POST 作业）

OutputLocation

包含有关作业结果和错误的存储位置的信息。

内容

S3

一个对象，描述用于接收还原请求结果的 Amazon S3 位置。

类型：S3Location

必需：是

更多信息

- 启动作业（POST 作业）

OutputSerialization

描述如何序列化输出。

内容

CSV

一个对象，描述逗号分隔值 (CSV) 编码的查询结果的序列化。

类型：CSVOutput 对象

必需：否

更多信息

- 启动作业（POST 作业）

S3Location

包含有关任务结果在 Amazon S3 中的存储位置的信息。

内容

AccessControlList

控制对存储结果的访问权限的授权列表。

类型：授权 对象数组

必需：否

BucketName

存储了任务结果的 Amazon S3 存储桶的名称。存储桶必须与包含输入档案对象的文件库位于同一 AWS 区域。

类型：字符串

必需：是

CannedACL

要应用于任务结果的标准访问控制列表 (ACL)。

类型：字符串

有效值: `private` | `public-read` | `public-read-write` | `aws-exec-read` | `authenticated-read` | `bucket-owner-read` | `bucket-owner-full-control`

必需：否

加密

一个对象，其中包含有关用于在 Amazon S3 中存储任务结果的加密的信息。

类型：加密 对象

必需：否

前缀

在此请求的结果前面添加的前缀。前缀的最大长度是 512 字节。

类型：字符串

必需：是

StorageClass

用于存储任务结果的存储类别。

类型：字符串

有效值: `STANDARD` | `REDUCED_REDUNDANCY` | `STANDARD_IA`

必需：否

标记

应用于任务结果的标签集。

类型：字符串到字符串映射

必需：否

UserMetadata

在 Amazon S3 中与任务结果一起存储的元数据的映射。

类型：字符串到字符串映射

必需：否

更多信息

- 启动作业（POST 作业）

SelectParameters

包含有关用于 select 的参数的信息。

内容

Expression

用于选择对象的表达式。该表达式不得超过 128,000 个字符的配额。

类型：字符串

必需：是

ExpressionType

所提供表达式的类型，例如 SQL。

有效值：SQL

类型：字符串

必需：是

InputSerialization

描述 select 中对象的序列化格式。

类型：InputSerialization 对象

必需：否

OutputSerialization

描述如何序列化 select 任务的结果。

必需：否

类型：OutputSerialization 对象

更多信息

- 启动作业（POST 作业）

数据检索操作

以下是 S3 Glacier 中可用的数据检索相关操作 API。

主题

- 获取数据检索策略（GET 策略）
- 列出预配置容量 (GET provisioned-capacity)
- 购买预配置容量 (POST provisioned-capacity)
- 设置数据检索策略（PUT 策略）

获取数据检索策略（GET 策略）

描述

此操作返回GET请求中指定的 AWS 账户 和 AWS 区域的当前数据检索策略。有关数据检索策略的更多信息，请参阅S3 Glacier 数据检索策略。

请求

要返回当前数据检索策略，请如以下语法示例所示，向数据检索策略 URI 发送 HTTP GET 请求。

语法

```
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`值为 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-” (连字符) ，在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

语法

```
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
         }
       ]
    }
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

BytesPerHour

一个小时内可以检索的最大字节数。

仅当 Strategy 字段的值为 `BytesPerHour` 时才会显示此字段。

类型：数字

规则

策略规则。尽管这是一种列表类型，但目前只有一条规则，它包含策略字段和可选 BytesPerHour 字段。

类型：数组

战略

数据检索策略的类型。

类型：字符串

有效值：`BytesPerHour`|`FreeTier`|None. `BytesPerHour` 等同于在控制台中选择 Max Retrieval Rate (最大检索速率)。`FreeTier` 等同于在控制台中选择 Free Tier Only (仅免费套餐)。None 等同

于在控制台中选择 No Retrieval Policy (无检索策略)。有关在控制台中选择数据检索策略的更多信息，请参阅S3 Glacier 数据检索策略。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例演示了如何获取数据检索策略。

示例请求

此示例将一个 GET 请求发送到策略位置的 URI。

```
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
```

响应示例

成功的响应会以 JSON 格式在响应正文中显示数据检索策略。

```
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"
          }
       ]
```

```
    }
}
```

相关部分

- 设置数据检索策略（PUT 策略）
- 启动作业（POST 作业）

列出预配置容量 (GET provisioned-capacity)

此操作列出指定 AWS 账户的预置容量。更多有关预配置容量的详细信息，请参阅 档案检索选项。

预配置容量单位将持续一个月，以购买日期和时间作为开始日期。此单位过期的日期为开始日期的整整一个月之后 (精确到秒)。

如果开始日期为一个月的第 31 天，过期日期为下个月的最后一天。例如，如果开始日期为 8 月 31 日，则过期日期为 9 月 30 日。如果开始日期为 1 月 31 日，则过期日期为 2 月 28 日。您可以在 响应示例 中看到此功能。

请求语法

要列出账户的预配置检索容量，请发送 HTTP GET 请求到预配置容量 URI，如以下语法示例中所示。

```
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`值为 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

如果此操作成功，则该服务将会发送回 HTTP 200 OK 响应。

响应语法

```
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"
      }
}
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

响应正文包含以下 JSON 字段。

CapacityId

用于确定预配置容量单位的 ID。

类型：字符串。

StartDate

预配置容量单位的购买日期，采用通用协调时间 (UTC)。

类型：字符串。以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

ExpirationDate

预配置容量单位的到期日期，采用通用协调时间 (UTC)。

类型：字符串。以 ISO 8601 日期格式表示的字符串，例如 `2013-03-20T17:03:43.221Z`。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

以下示例列出了账户的预配置容量单位。

示例请求

在此示例中，发送 GET 请求来检索指定账户的预配置容量单位列表。

```
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
```

响应示例

如果请求成功，Amazon S3 Glacier（S3 Glacier）会返回带有一个账户预配置容量单位列表的 HTTP `200 OK`，如以下示例中所示。

列出的第一个预配置容量单位示例开始日期为 2017 年 1 月 31 日，过期日期为 2017 年 2 月 28 日。如之前所述，如果开始日期为某个月的第 31 天，过期日期则是下个月的最后一天。

```
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" ",
      },
      ...
}
```

相关部分

- 购买预配置容量 (POST provisioned-capacity)

购买预配置容量 (POST provisioned-capacity)

此操作为 AWS 账户购买预置容量单位。

预配置容量单位将持续一个月，以购买日期和时间作为开始日期。此单位过期的日期为开始日期的整整一个月之后 (精确到秒)。

如果开始日期为一个月的第 31 天，过期日期为下个月的最后一天。例如，如果开始日期为 8 月 31 日，则过期日期为 9 月 30 日。如果开始日期为 1 月 31 日，则过期日期为 2 月 28 日。

预配置容量帮助确保在您需要时，可以使用针对加速检索的检索容量。每个容量单位确保每五分钟至少可以执行三个加速检索，并提供高达 150 MB/秒的检索吞吐量。更多有关预配置容量的详细信息，请参阅 档案检索选项。

> **Note**
>
> 每个配置的容量单位限制为两个单位。 AWS 账户

请求

要购买的预配置容量单位，请向预配置容量 URI AWS 账户 发送 HTTP POST 请求。

语法

```
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`值为 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-”（连字符），在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

请求参数

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

此操作没有请求正文。

响应

如果操作请求成功，则该服务会返回 HTTP `201 Created` 响应。

语法

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

```
x-amz-capacity-id: CapacityId
```

响应标头

除了所有操作通用的响应标头外，成功的响应中还包括以下响应标头。有关通用响应标头的信息，请参阅常用响应标头。

名称	描述
x-amz-capacity-id	用于确定预配置容量单位的 ID。 类型：字符串

响应正文

此操作不返回响应正文。

错误

除了所有 Amazon S3 Glacier 操作中常见的可能错误外，此操作还包括以下一个或多个错误。有关 Amazon S3 Glacier 错误的信息以及错误代码列表，请参阅错误响应。

代码	描述	HTTP 状态代码	类型
LimitExceededException	如果指定请求将超出账户的预配置容量单位限制，则返回此代码。	400 Bad Request	客户端

示例

以下示例为账户购买预配置容量。

示例请求

以下示例发送 HTTP POST 请求以购买预配置容量单位。

```
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
```

响应示例

如果请求成功，Amazon S3 Glacier（S3 Glacier）将返回 `HTTP 201 Created` 响应，如以下示例中所示。

```
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
```

相关部分

- 列出预配置容量 (GET provisioned-capacity)

设置数据检索策略（PUT 策略）

描述

此操作在PUT请求中指定的 AWS 区域设置并随后制定数据检索策略。您可以为每个 AWS 区域设置一个策略 AWS 账户。该策略在 PUT 操作成功后数分钟内应用。

在策略应用之前，设置策略的操作并不影响正在进行的检索作业。有关数据检索策略的更多信息，请参阅S3 Glacier 数据检索策略。

请求

语法

要设置数据检索策略，请如以下语法示例所示，向数据检索策略 URI 发送 HTTP PUT 请求。

```
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`值为 AWS 账户 ID。此值必须与用于签署请求的凭据关联的 AWS 账户 ID 相匹配。您可以指定一个 AWS 账户 ID，也可以选择指定一个 “-” (连字符) ，在这种情况下，Amazon S3 Glacier 使用与用于签署请求的凭证关联的 AWS 账户 ID。如果您指定账户 ID，请勿在 ID 中包含连字符 ('-')。

请求参数

此操作不使用请求参数。

请求标头

此操作仅使用所有操作通用的请求标头。有关通用请求标头的信息，请参阅 常见请求标头。

请求正文

请求正文中包含以下 JSON 字段。

BytesPerHour

一个小时内可以检索的最大字节数。

仅当 Strategy 字段的值为 `BytesPerHour` 时，此字段才是必需字段。若您设置此字段，但 Strategy 字段并未设置为 `BytesPerHour`，则 PUT 操作会被拒绝。

类型：数字

必需：如果 Strategy 字段设置为 `BytesPerHour`，则为必需。否则不是必需。

有效值：最小整数值 1。最大整数值 2^63 - 1（含）。

规则

策略规则。尽管这是一种列表类型，但目前只能有一个规则，其中包含策略字段和可选 BytesPerHour 字段。

类型：数组

必需：是

战略

要设置的数据检索策略的类型。

类型：字符串

必需：是

有效值：`BytesPerHour`|`FreeTier`|None. `BytesPerHour` 等同于在控制台中选择 Max Retrieval Rate (最大检索速率)。`FreeTier` 等同于在控制台中选择 Free Tier Only (仅免费套餐)。None 等同于在控制台中选择 No Retrieval Policy (无检索策略)。有关在控制台中选择数据检索策略的更多信息，请参阅S3 Glacier 数据检索策略。

响应

语法

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

响应标头

此操作仅使用大多数响应通用的响应标头。有关通用响应标头的信息，请参阅 常用响应标头。

响应正文

此操作不返回响应正文。

错误

有关 Amazon S3 Glacier 异常和错误消息的信息，请参阅错误响应。

示例

示例请求

以下示例发送一个 HTTP PUT 请求，其中 Strategy 字段设置为 `BytesPerHour`。

```
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
          }
       ]
    }
}
```

以下示例发送一个 HTTP PUT 请求，其中 Strategy 字段设置为 `FreeTier`。

```
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"
          }
       ]
    }
}
```

以下示例发送一个 HTTP PUT 请求，其中 Strategy 字段设置为 None。

```
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"
          }
       ]
    }
}
```

响应示例

如以下示例所示，如果请求成功，Amazon S3 Glacier（S3 Glacier）会设置策略并返回 `HTTP 204 No Content`。

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

相关部分

- 获取数据检索策略（GET 策略）

- 启动作业（POST 作业）

文档历史记录

- 当前产品版本：2012-06-01

下表介绍 2018 年 7 月 5 日之后《Amazon S3 Glacier 开发人员指南》每个版本的重要修改。要获得本文档的更新通知，您可以订阅 RSS 源。

变更	说明	日期
缩短了通过 S3 批量操作发出的标准还原请求的启动时间	现在，通过 S3 批量操作发出的还原请求的标准检索可以在几分钟内启动。有关更多信息，请参阅归档检索选项。	2023 年 8 月 9 日
Amazon S3 对于 S3 Glacier Flexible Retrieval 和 S3 Glacir Deep Archive 支持更高的还原请求速率	对于 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类，Amazon S3 支持以每个 AWS 账户 每秒最多 1000 个事务的速率发出还原请求。	2022 年 11 月 15 日
Amazon Glacier 名称更改	Amazon Glacier 现在名为 Amazon S3 Glacier，可以更好地反映 Glacier 与 Amazon S3 的集成。	2018 年 11 月 20 日
现在可通过 RSS 更新	您现在可以订阅 RSS 源来接收有关《Amazon S3 Glacier 开发人员指南》更新的通知。	2018 年 7 月 5 日

早期更新

下表描述了 2018 年 7 月 5 日之前每个《Amazon S3 Glacier 开发人员指南》发行版中的重要更改。

更改	描述	发行日期
加速和批量数据检索	现在，除了标准检索之外，S3 Glacier 还支持加速和批量数据检索。有关更多信息，请参阅 档案检索选项。	2016 年 11 月 21 日
保管库锁定	S3 Glacier 现在支持文件库锁定，让您轻松地利用文件库锁定策略对单独的 S3 Glacier 文件库进行部署和实施合规性控制。有关更多信息，请参阅S3 Glacier 文件库锁定 和文件库锁定策略。	2015 年 7 月 8 日
文件库标记	S3 Glacier 现在让您能够标记 S3 Glacier 文件库，从而更加轻松地管理资源和成本。标签是您可以定义并与文件库关联的标签，使用标签可以为 AWS 成本报告等操作添加筛选功能。有关更多信息，请参阅标记 Amazon S3 Glacier 资源 和标记 S3 Glacier 文件库。	2015 年 6 月 22 日
文件库访问策略	S3 Glacier 现在支持通过使用文件库访问策略来管理对单独的 S3 Glacier 文件库的访问。您现在可以直接在文件库上定义访问策略，以便更轻松地向组织内部的用户和业务组以及外部业务合作伙伴授予文件库访问权。有关更多信息，请参阅 库访问策略。	2015 年 4 月 27 日
数据检索策略和审核记录	S3 Glacier 现在支持数据检索策略和审核记录。数据检索策略允许您轻松设置数据检索限制并简化数据检索费用管理。只需在 AWS Management Console 或使用 S3 Glacier API 中单击几下即可定义自己的数据检索限制。有关更多信息，请参阅 S3 Glacier 数据检索策略。 此外，S3 Glacier 现在支持使用进行审核日志 AWS CloudTrail，它可以记录您的账户的 S3 Glacier API 调用，并将日志文件传输到您指定的 Amazon S3 存储桶。有关更多信息，请参阅 使用记录亚马逊 S3 Glacier API 调用 AWS CloudTrail。	2014 年 12 月 11 日
Java 示例的更新	更新了本指南中使用 适用于 Java 的 AWS SDK的 Java 代码示例。	2014 年 6 月 27 日

更改	描述	发行日期
限制文件库清单检索	您现在可以通过筛选档案创建日期或设置限制，来限制检索的文件库清单项目数。有关限制清单检索的更多信息，请参阅 确定清单检索范围 主题中的 启动作业（POST 作业）。	2013 年 12 月 31 日
已移除过时 URLs	从代码示例 URLs 中删除了指向旧安全证书页面的内容。	2013 年 7 月 26 日
支持范围检索	S3 Glacier 现在支持检索特定范围的档案。您可以启动任务，请求 S3 Glacier 为后续下载准备整个档案或档案的一部分。如果档案非常大，则您可能会发现，启动多个连续的任务来准备档案具有成本效率。 有关更多信息，请参阅 在 S3 Glacier 中下载档案。	2012 年 11 月 13 日
新指南	本指南是《Amazon S3 Glacier 开发人员指南》的第一个版本。	2012 年 8 月 20 日

AWS 词汇表

有关最新 AWS 术语，请参阅《AWS 词汇表 参考资料》中的AWS 词汇表。