aws

Developer Guide

Amazon Glacier

API Version 2012-06-01

Amazon Glacier: Developer Guide

Table of Contents

This page is only for existing customers of the Amazon Glacier service using Vaults and the original REST API from 2012.

If you're looking for archival storage solutions, we recommend using the Amazon Glacier storage classes in Amazon S3, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive. To learn more about these storage options, see Amazon Glacier storage classes.

Amazon Glacier (original standalone vault-based service) is no longer accepting new customers. Amazon Glacier is a standalone service with its own APIs that stores data in vaults and is distinct from Amazon S3 and the Amazon S3 Glacier storage classes. Your existing data will remain secure and accessible in Amazon Glacier indefinitely. No migration is required. For low-cost, long-term archival storage, AWS recommends the Amazon S3 Glacier storage classes, which deliver a superior customer experience with S3 bucket-based APIs, full AWS Region availability, lower costs, and AWS service integration. If you want enhanced capabilities, consider migrating to Amazon S3 Glacier storage classes by using our AWS Solutions Guidance for transferring data from Amazon Glacier vaults to Amazon S3 Glacier storage classes.

What Is Amazon Glacier?

If you're currently using the Amazon Glacier service and want to learn more, you'll find the information that you need in this guide. Amazon Glacier is a secure and durable service for low-cost data archiving and long-term backup using vaults. For more information about Amazon Glacier service pricing, see Amazon Glacier pricing.

Topics

Do You Currently Use Amazon Glacier?

> **Note**
>
> This section is about the Amazon Glacier service. If you currently use the Amazon S3 Glacier storage classes (**S3 Glacier Instant Retrieval**, **S3 Glacier Flexible Retrieval**, and **S3 Glacier Deep Archive**), see Storage classes for archiving objects in the *Amazon S3 User Guide*.

If you currently use the Amazon Glacier service and want to learn more, we recommend that you begin by reading the following sections:

- **What is Amazon Glacier** – The rest of this section describes the underlying data model, the operations it supports, and the AWS SDKs that you can use to interact with the service.
- **Getting Started** – The Getting Started with Amazon Glacier section walks you through the process of creating a vault, uploading archives, creating jobs to download archives, retrieving the job output, and deleting archives.

> **Important**
>
> Amazon Glacier does provide a console. However, any archive operation, such as upload, download, or deletion, requires you to use the AWS Command Line Interface (AWS CLI)

or write code. There is no console support for archive operations. For example, to upload data, such as photos, videos, and other documents, you must either use the AWS CLI or write code to make requests, by using either the REST API directly or by using the AWS SDKs.

To install the AWS CLI, see AWS Command Line Interface. For more information about using Amazon Glacier with the AWS CLI, see the AWS CLI Reference for Amazon Glacier. For examples of using the AWS CLI to upload archives to Amazon Glacier, see Using Amazon Glacier with the AWS Command Line Interface.

Beyond the getting started section, you'll probably want to learn more about Amazon Glacier operations. The following sections provide detailed information about working with Amazon Glacier by using the REST API and the AWS SDKs for Java and Microsoft .NET:

- Using the AWS SDKs with Amazon Glacier

  This section provides an overview of the AWS SDKs used in various code examples in this guide. A review of this section will help when reading the following sections. It includes an overview of the high-level and the low-level APIs that these SDKs offer, when to use them, and common steps for running the code examples provided in this guide.

- Working with Vaults in Amazon Glacier

  This section provides details of various vault operations, such as creating a vault, retrieving vault metadata, using jobs to retrieve vault inventory, and configuring vault notifications. In addition to using the Amazon Glacier console, you can use the AWS SDKs for various vault operations. This section describes the API and provides working samples by using the AWS SDK for Java and the AWS SDK for .NET.

- Working with Archives in Amazon Glacier

  This section provides details of archive operations, such as uploading an archive in a single request or using a multipart upload operation to upload large archives in parts. The section also explains how to create jobs to download archives asynchronously. The section provides examples by using the AWS SDK for Java and the AWS SDK for .NET.

- API Reference for Amazon Glacier

Amazon Glacier is a RESTful service. This section describes the REST operations, including the syntax, and example requests and responses for all the operations. The AWS SDK libraries wrap this API, simplifying your programming tasks.

Amazon Glacier Data Model

The Amazon Glacier data model core components include vaults and archives. Amazon Glacier is a REST-based web service. In terms of REST, vaults and archives are the resources. In addition, the Amazon Glacier data model includes job and notification-configuration resources. These resources complement the core resources.

Topics

Vault

In Amazon Glacier, a *vault* is a container for storing archives. A vault is similar to an Amazon S3 bucket. When you create a vault, you specify a name and choose an AWS Region where you want to create the vault.

Each vault resource has a unique address. The general form is:

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

For example, suppose that you create a vault (`examplevault`) in the US West (Oregon) Region in your account with the ID 111122223333. You can address this vault by using the following URI:

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

Here is what the various components of the URI mean:

- `glacier.us-west-2.amazonaws.com` identifies the US West (Oregon) Region.
- `111122223333` is the AWS account ID that owns the vault.

- `vaults` refers to the collection of vaults that are owned by the AWS account.
- `examplevault` identifies a specific vault in the vaults collection.

An AWS account can create vaults in any supported AWS Region. For list of supported AWS Regions, see Accessing Amazon Glacier. Within a Region, an account must use unique vault names. An AWS account can create same-named vaults in different Regions.

You can store an unlimited number of archives in a vault. Depending on your business or application needs, you can store these archives in one vault or multiple vaults.

Amazon Glacier supports various vault operations. Vault operations are Region-specific. For example, when you create a vault, you create it in a specific Region. When you request a vault list, you request it from a specific AWS Region, and the resulting list includes only vaults created in that specific Region.

Archive

An *archive* can be any data, such as a photo, video, or document. An archive is similar to an Amazon S3 object, and is the base unit of storage in Amazon Glacier. Each archive has a unique ID and an optional description. You can specify this optional description only during the upload of an archive. Amazon Glacier assigns the archive an ID, which is unique in the AWS Region in which the archive is stored.

Each archive has a unique address. The general form is as follows:

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

The following is an example URI of an archive stored in the `examplevault` vault in the US West (Oregon) Region in account 111122223333:

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

You can store an unlimited number of archives in a vault.

Job

An Amazon Glacier job can retrieve an archive, or get an inventory of a vault.

Retrieving archives and vault inventories (lists of archives) are asynchronous operations in Amazon Glacier, in which you first initiate a job, and then download the job output after Amazon Glacier completes the job.

> **Note**
>
> Amazon Glacier offers a cold-storage data-archival solution. If your application needs a storage solution that requires real-time data retrieval, you might consider using Amazon S3. For more information, see Amazon Simple Storage Service (Amazon S3).

To initiate a vault inventory job, you provide a vault name. Archive retrieval jobs require both the vault name and the archive ID. You can also provide an optional job description to help identify the jobs.

Archive retrieval and vault inventory jobs are associated with a vault. A vault can have multiple jobs in progress at any point in time. When you send a job request (initiate a job), Amazon Glacier returns to you a job ID to track the job. Each job is uniquely identified by a URI of the form:

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

The following is an example of a job associated with an `examplevault` vault in the US West (Oregon) Region in account 111122223333.

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

For each job, Amazon Glacier maintains information, such as the job type, description, creation date, completion date, and job status. You can obtain information about a specific job or obtain a list of all your jobs associated with a vault. The list of jobs that Amazon Glacier returns includes all the in-progress and recently finished jobs.

Notification Configuration

Because jobs take time to run, Amazon Glacier supports a notification mechanism to notify you when a job is completed. You can configure a vault to send a notification to an Amazon Simple Notification Service (Amazon SNS) topic when a job is completed. You can specify one Amazon SNS topic per vault in the notification configuration.

Amazon Glacier stores the notification configuration as a JSON document. The following is an example vault notification configuration:

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

Notification configurations are associated with vaults; you can have one for each vault. Each notification configuration resource is uniquely identified by a URI of the form:

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

Amazon Glacier supports operations to set, get, and delete a notification configuration. When you delete a notification configuration, no notifications are sent when any data retrieval operation on the vault is completed.

Supported Operations in Amazon Glacier

To work with vaults and archives (see Amazon Glacier Data Model), Amazon Glacier supports a set of operations. Among all the supported operations, only the following operations are asynchronous:

- Retrieving an archive
- Retrieving a vault inventory (list of archives)

These operations require you to first initiate a job and then download the job output. The following sections summarize the Amazon Glacier operations.

Vault Operations

Amazon Glacier provides operations to create and delete vaults. You can obtain a vault description for a specific vault or for all vaults in an AWS Region. The vault description provides information, such as the creation date, the number of archives in the vault, the total size in bytes used by all the archives in the vault, and the date that Amazon Glacier generated the vault inventory. Amazon Glacier also provides operations to set, retrieve, and delete a notification configuration on the vault. For more information, see Working with Vaults in Amazon Glacier.

Archive Operations

Amazon Glacier provides operations for you to upload and delete archives. You cannot update an existing archive; you must delete the existing archive and upload a new archive. Each time that you upload an archive, Amazon Glacier generates a new archive ID. For more information, see Working with Archives in Amazon Glacier.

Jobs

You can initiate an Amazon Glacier job to perform a retrieval on an archive or get an inventory of a vault.

The following are the types of Amazon Glacier jobs:

- `archive-retrieval` – Retrieve an archive.

  For more information, see Downloading an Archive in Amazon Glacier.
- `inventory-retrieval` – Inventory a vault.

  For more information, see Downloading a Vault Inventory in Amazon Glacier.

Accessing Amazon Glacier

Amazon Glacier is a RESTful web service that uses HTTP and HTTPS as a transport protocol and JavaScript Object Notation (JSON) as a message-serialization format. Your application code can make requests directly to the Amazon Glacier web service API. When using the REST API directly, you must write the necessary code to sign and authenticate your requests. For more information about the API, see API Reference for Amazon Glacier.

Alternatively, you can simplify application development by using the AWS SDKs that wrap the Amazon Glacier REST API calls. You provide your credentials, and these libraries take care of authentication and request signing. For more information about using the AWS SDKs, see Using the AWS SDKs with Amazon Glacier.

Amazon Glacier also provides a console. However, all archive and job operations require you to write code and make requests by using either the REST API directly or the AWS SDK wrapper libraries. To access the Amazon Glacier console, go to Amazon Glacier Console.

Regions and Endpoints

You create a vault in a specific AWS Region. You always send your Amazon Glacier requests to an endpoint specific to an AWS Region. For a list of the AWS Regions supported by Amazon Glacier, see Amazon Glacier endpoints and quotas in the *AWS General Reference*.

Getting Started with Amazon Glacier

You can get started with Amazon Glacier (Amazon Glacier) by working with vaults and archives. A *vault* is a container for storing archives, and an *archive* is any object, such as a photo, video, or document, that you store in a vault. An archive is the base unit of storage in Amazon Glacier. This getting started exercise provides instructions for you to explore basic Amazon Glacier operations on vaults and archives. For more information about these resources, see the Amazon Glacier Data Model section.

In the getting started exercise, you will create a vault, upload and download an archive, and then delete the archive and the vault. You can do all these operations programmatically. However, the getting started exercise uses the Amazon Glacier management console to create and delete a vault. For uploading and downloading an archive, this getting started section uses the high-level API for the AWS SDK for Java and the AWS SDK for .NET. The high-level API provides a simplified programming experience when working with Amazon Glacier. For more information about using the high-level API with the AWS SDKs, see Using the AWS SDKs with Amazon Glacier.

> **Important**
>
> Amazon Glacier does provide a console. However, any archive operation, such as upload, download, or deletion, requires you to use the AWS Command Line Interface (CLI) or write code. There is no console support for archive operations. For example, to upload data, such as photos, videos, and other documents, you must either use the AWS CLI or write code to make requests, by using either the REST API directly or by using the AWS SDKs.
> To install the AWS CLI, see AWS Command Line Interface. For more information about using Amazon Glacier with the AWS CLI, see the AWS CLI Reference for Amazon Glacier. For examples of using the AWS CLI to upload archives to Amazon Glacier, see Using Amazon Glacier with the AWS Command Line Interface.

This getting started exercise provides code examples in Java and C# for you to upload and download an archive. The last section of the getting started exercise provides steps where you can learn more about the developer experience with Amazon Glacier.

Topics

Step 1: Before You Begin with Amazon Glacier

Before you can start with this exercise, you must sign up for an AWS account (if you don't already have one), and then download one of the AWS SDKs. See the following sections for instructions.

Topics

Set Up an AWS account and an Administrator User

If you have not already done so, you must sign up for an AWS account and create an administrator user in the account.

To complete the setup, follow the instructions in the following topics.

Set Up an AWS account and Create an Administrator User

Sign up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including Amazon Glacier. You are charged only for the services that you use. For more information about Amazon Glacier usage rates, see the Amazon Glacier pricing page.

If you already have an AWS account, skip to Download the Appropriate AWS SDK. If you don't have an AWS account, use the following procedure to create one.

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

To create an administrator user, choose one of the following options.

Choose one way to manage your administrator	To	By	You can also
In IAM Identity Center (Recommended)	Use short-term credentials to access AWS. This aligns with the security best practices. For information about best practices, see Security best practices in IAM in the *IAM User Guide*.	Following the instructions in Getting started in the *AWS IAM Identity Center User Guide*.	Configure programmatic access by Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*.
In IAM	Use long-term credentials to access AWS.	Following the instructions in Create an IAM user for	Configure programmatic access by Manage access keys

Choose one way to manage your administrator	To	By	You can also
(Not recommended)		emergency access in the *IAM User Guide*.	for IAM users in the *IAM User Guide*.

Download the Appropriate AWS SDK

To try the getting started exercise, you must decide which programming language you want to use, and then download the appropriate AWS SDK for your development platform.

The getting started exercise provides examples in Java and C#.

Downloading the AWS SDK for Java

To test the Java examples in this developer guide, you need the AWS SDK for Java. You have the following download options:

- If you are using Eclipse, you can download and install the AWS Toolkit for Eclipse by using the update site http://aws.amazon.com/eclipse/. For more information, see AWS Toolkit for Eclipse.
- If you are using any other IDE to create your application, download the AWS SDK for Java.

Downloading the AWS SDK for .NET

To test the C# examples in this developer guide, you need the AWS SDK for .NET. You have the following download options:

- If you are using Visual Studio, you can install both the AWS SDK for .NET and the AWS Toolkit for Visual Studio. The toolkit provides AWS Explorer for Visual Studio and project templates that you can use for development. To download the AWS SDK for .NET, go to http://aws.amazon.com/sdkfornet. By default, the installation script installs both the AWS SDK and the AWS Toolkit for Visual Studio. To learn more about the toolkit, see the AWS Toolkit for Visual Studio User Guide.

- If you are using any other IDE to create your application, you can use the same link provided in the preceding step and install only the AWS SDK for .NET.

Step 2: Create a Vault in Amazon Glacier

A vault is a container for storing archives. Your first step is to create a vault in one of the supported AWS Regions. For a list of the AWS Regions that are supported by Amazon Glacier, see Amazon Glacier endpoints and quotas in the *AWS General Reference*.

You can create vaults programmatically or by using the Amazon Glacier console. This section uses the console to create a vault.

To create a vault

1. Sign in to the AWS Management Console and open the Amazon Glacier console at https://console.aws.amazon.com/glacier/home.
2. In the left navigation pane, choose **Vaults**.
3. Choose **Create vault**.

   The **Create vault** page opens.
4. Under **Select a Region**, select an AWS Region from the Region selector. Your vault will be located in the Region that you select.
5. For **Vault name**, enter a name for your vault.

   The following are the vault-naming requirements:

   - A vault name must be unique within an AWS account and the AWS Region in which the vault is created.
   - A vault name must be between 1 and 255 characters long.
   - A vault name can contain only the following characters: **a–z**, **A–Z**, **0–9**, **_** (underscore), **-** (hyphen), and **.** (period).
6. Under **Event notifications**, to turn on or off notifications on a vault for when a job is completed, choose one of the following settings:

   - **Turn off notifications** – Notifications are turned off, and notifications are not sent to an Amazon Simple Notification Service (Amazon SNS) topic when a specified job is completed.

- **Turn on notifications** – Notifications are turned on, and notifications are sent to the provided Amazon SNS topic when a specified job is completed.

  If you chose **Turn on notifications**, see Configuring Vault Notifications by Using the Amazon Glacier Console.

7. If the AWS Region and vault name are correct, then choose **Create vault**.

Your new vault is now listed on the **Vaults** page in the Amazon Glacier console.

Step 3: Upload an Archive to a Vault in Amazon Glacier

In this step, you'll upload a sample archive to the vault that you created in the preceding step (see Step 2: Create a Vault in Amazon Glacier). Depending on the development platform that you're using, choose one of the links at the end of this section.

> **Important**
>
> Any archive operation, such as upload, download, or deletion, requires you to use the AWS Command Line Interface (CLI) or write code. There is no console support for archive operations. For example, to upload data, such as photos, videos, and other documents, you must either use the AWS CLI or write code to make requests, by using either the REST API directly or by using the AWS SDKs.
> To install the AWS CLI, see AWS Command Line Interface. For more information about using Amazon Glacier with the AWS CLI, see AWS CLI Reference for Amazon Glacier. For examples of using the AWS CLI to upload archives to Amazon Glacier, see Using Amazon Glacier with the AWS Command Line Interface.

An archive is any object, such as a photo, video, or document, that you store in a vault. An archive is the base unit of storage in Amazon Glacier. You can upload an archive in a single request. For large archives, Amazon Glacier provides a multipart upload API operation that enables you to upload an archive in parts.

In this getting started section, you upload a sample archive in a single request. For this exercise, you specify a file that is smaller in size. For larger files, multipart upload is suitable. For more information, see Uploading Large Archives in Parts (Multipart Upload).

Topics

Upload an Archive to a Vault in Amazon Glacier by Using the AWS SDK for Java

The following Java code example uses the high-level API of the AWS SDK for Java to upload a sample archive to the vault. In the code example, note the following:

- The example creates an instance of the `AmazonGlacierClient` class.
- The example uses the `upload` API operation of the `ArchiveTransferManager` class from the high-level API of the AWS SDK for Java.
- The example uses the US West (Oregon) Region (`us-west-2`).

For step-by-step instructions on how to run this example, see Running Java Examples for Amazon Glacier Using Eclipse. You must update the code as shown with the name of the archive file that you want to upload.

> **Note**
>
> Amazon Glacier keeps an inventory of all the archives in your vaults. When you upload the archive in the following example, it will not appear in a vault in the management console until the vault inventory has been updated. This update usually happens once a day.

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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();
    }
}
```

- For API details, see UploadArchive in *AWS SDK for Java 2.x API Reference*.

Upload an Archive to a Vault in Amazon Glacier by Using the AWS SDK for .NET

The following C# code example uses the high-level API of the AWS SDK for .NET to upload a sample archive to the vault. In the code example, note the following:

- The example creates an instance of the `ArchiveTransferManager` class for the specified Amazon Glacier Region endpoint.
- The code example uses the US West (Oregon) Region (`us-west-2`).

- The example uses the `Upload` API operation of the `ArchiveTransferManager` class to upload your archive. For small archives, this operation uploads the archive directly to Amazon Glacier. For larger archives, this operation uses the multipart upload API operation in Amazon Glacier to split the upload into multiple parts for better error recovery, if any errors are encountered while streaming the data to Amazon Glacier.

For step-by-step instructions on how to run the following example, see Running Code Examples. You must update the code as shown with the name of your vault and the name of the archive file to upload.

> **Note**
>
> Amazon Glacier keeps an inventory of all the archives in your vaults. When you upload the archive in the following example, the archive will not appear in a vault in the management console until the vault inventory has been updated. This update usually happens once a day.

Example— Uploading an Archive by Using the High-Level API of the AWS SDK for .NET

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

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

        public static void Main(string[] args)
        {
            try
            {
                var manager = new
 ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
                // Upload an archive.
```

```
                string archiveId = manager.Upload(vaultName, "getting started archive
 test", archiveToUpload).ArchiveId;
                Console.WriteLine("Copy and save the following Archive ID for the next
 step.");
                Console.WriteLine("Archive ID: {0}", archiveId);
                Console.WriteLine("To continue, press Enter");
                Console.ReadKey();
            }
            catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
            catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
            catch (Exception e) { Console.WriteLine(e.Message); }
            Console.WriteLine("To continue, press Enter");
            Console.ReadKey();
        }
    }
}
```

Step 4: Download an Archive from a Vault in Amazon Glacier

In this step, you'll download the sample archive that you uploaded previously in Step 3: Upload an Archive to a Vault in Amazon Glacier.

> **Important**
>
> Amazon Glacier does provide a console. However, any archive operation, such as upload, download, or deletion, requires you to use the AWS Command Line Interface (CLI) or write code. There is no console support for archive operations. For example, to upload data, such as photos, videos, and other documents, you must either use the AWS CLI or write code to make requests, by using either the REST API directly or by using the AWS SDKs.
> To install the AWS CLI, see AWS Command Line Interface. For more information about using Amazon Glacier with the AWS CLI, see AWS CLI Reference for Amazon Glacier. For examples of using the AWS CLI to upload archives to Amazon Glacier, see Using Amazon Glacier with the AWS Command Line Interface.

In general, retrieving your data from Amazon Glacier is a two-step process:

1. Initiate a retrieval job.
2. After the job is completed, download the bytes of data.

To retrieve an archive from Amazon Glacier, you first initiate a job. After the job is completed, you download the data. For more information about archive retrievals, see Retrieving Amazon Glacier Archives.

The access time of your request depends on the retrieval option that you choose: Expedited, Standard, or Bulk retrievals. For all but the largest archives (250 MB+), archives accessed by using Expedited retrievals are typically made available within 1–5 minutes. Archives retrieved by using Standard retrievals typically are available between 3–5 hours. Bulk retrievals typically are available within 5–12 hours. For more information about the various retrieval options, see the Amazon Glacier FAQ. For information about data retrieval charges, see the Amazon Glacier pricing page.

The code examples shown in the following topics initiate the job, wait for it to be completed, and then download the archive's data.

Topics

- Download an Archive from a Vault in Amazon Glacier by Using the AWS SDK for Java
- Download an Archive from a Vault in Amazon Glacier by Using the AWS SDK for .NET

Download an Archive from a Vault in Amazon Glacier by Using the AWS SDK for Java

The following Java code example uses the high-level API of the AWS SDK for Java to download the archive that you uploaded in the previous step. In the code example, note the following:

- The example creates an instance of the `AmazonGlacierClient` class.
- The code uses the US West (Oregon) Region (`us-west-2`) to match the location where you created the vault in Step 2: Create a Vault in Amazon Glacier.
- The example uses the `download` API operation of the `ArchiveTransferManager` class from the high-level API of the AWS SDK for Java. The example creates an Amazon Simple Notification Service (Amazon SNS) topic, and an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to that topic. If you created an AWS Identity and Access Management (IAM) admin user as instructed in Step 1: Before You Begin with Amazon Glacier, your user has the necessary IAM permissions for the creation and use of the Amazon SNS topic and Amazon SQS queue.

For step-by-step instructions on how to run this example, see Running Java Examples for Amazon Glacier Using Eclipse. You must update the code as shown with the archive ID of the file that you uploaded in Step 3: Upload an Archive to a Vault in Amazon Glacier.

Example— Downloading an Archive by Using the AWS SDK for Java

```
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);
        }
    }
}
```

Download an Archive from a Vault in Amazon Glacier by Using the AWS SDK for .NET

The following C# code example uses the high-level API of the AWS SDK for .NET to download the archive that you uploaded previously in Upload an Archive to a Vault in Amazon Glacier by Using the AWS SDK for .NET. In the code example, note the following:

- The example creates an instance of the `ArchiveTransferManager` class for the specified Amazon Glacier Region endpoint.
- The code example uses the US West (Oregon) Region (`us-west-2`) to match the location where you created the vault previously in Step 2: Create a Vault in Amazon Glacier.
- The example uses the `Download` API operation of the `ArchiveTransferManager` class to download your archive. The example creates an Amazon Simple Notification Service (Amazon SNS) topic, and an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to that topic. If you created an AWS Identity and Access Management (IAM) admin user as instructed in Step 1: Before You Begin with Amazon Glacier, your user has the necessary IAM permissions for the creation and use of the Amazon SNS topic and Amazon SQS queue.
- The example then initiates the archive retrieval job and polls the queue for the archive to be available. When the archive is available, the download begins. For information about retrieval times, see Archive Retrieval Options.

For step-by-step instructions on how to run this example, see Running Code Examples. You must update the code as shown with the archive ID of the file that you uploaded in Step 3: Upload an Archive to a Vault in Amazon Glacier.

Example— Download an Archive by Using the High-Level API of the AWS SDK for .NET

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

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

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

        public static void Main(string[] args)
        {
            try
            {
                var manager = new
 ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);

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

        static int currentPercentage = -1;
        static void progress(object sender, StreamTransferProgressArgs args)
        {
            if (args.PercentDone != currentPercentage)
            {
                currentPercentage = args.PercentDone;
```

```
                Console.WriteLine("Downloaded {0}%", args.PercentDone);
            }
        }
    }
}
```

Step 5: Delete an Archive from a Vault in Amazon Glacier

In this step, you'll delete the sample archive that you uploaded in Step 3: Upload an Archive to a Vault in Amazon Glacier.

> **Important**
>
> You cannot delete an archive by using the Amazon Glacier console. Any archive operation, such as upload, download, or deletion, requires you to use the AWS Command Line Interface (CLI) or write code. To upload data, such as photos, videos, and other documents, you must either use the AWS CLI or write code to make requests, by using either the REST API directly or by using the AWS SDKs.
>
> To install the AWS CLI, see AWS Command Line Interface. For more information about using Amazon Glacier with the AWS CLI, see AWS CLI Reference for Amazon Glacier. For examples of using the AWS CLI to upload archives to Amazon Glacier, see Using Amazon Glacier with the AWS Command Line Interface.

Delete the sample archive by following one of these SDKs or the AWS CLI:

- Delete an Archive from a Vault in Amazon Glacier by Using the AWS SDK for Java
- Delete an Archive from a Vault in Amazon Glacier by Using the AWS SDK for .NET
- Delete an Archive in Amazon Glacier by Using the AWS CLI

Related Sections

- Step 3: Upload an Archive to a Vault in Amazon Glacier
- Deleting an Archive in Amazon Glacier

Delete an Archive from a Vault in Amazon Glacier by Using the AWS SDK for Java

The following code example uses the AWS SDK for Java to delete the archive. In the code, note the following:

- The `DeleteArchiveRequest` object describes the delete request, including the vault name where the archive is located and the archive ID.
- The `deleteArchive` API operation sends the request to Amazon Glacier to delete the archive.
- The example uses the US West (Oregon) Region (`us-west-2`).

For step-by-step instructions on how to run this example, see Running Java Examples for Amazon Glacier Using Eclipse. You must update the code as shown with the archive ID of the file that you uploaded in Step 3: Upload an Archive to a Vault in Amazon Glacier.

Example— Deleting an Archive by Using the AWS SDK for Java

```
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);
        }
    }
}
```

Delete an Archive from a Vault in Amazon Glacier by Using the AWS SDK for .NET

The following C# code example uses the high-level API of the AWS SDK for .NET to delete the archive that you uploaded in the previous step. In the code example, note the following:

- The example creates an instance of the `ArchiveTransferManager` class for the specified Amazon Glacier Region endpoint.
- The code example uses the US West (Oregon) Region (`us-west-2`).
- The example uses the `Delete` API operation of the `ArchiveTransferManager` class that's provided as part of the high-level API of the AWS SDK for .NET.

For step-by-step instructions on how to run this example, see Running Code Examples. You must update the code as shown with the archive ID of the file that you uploaded in Step 3: Upload an Archive to a Vault in Amazon Glacier.

Example— Deleting an Archive by Using the High-Level API of the AWS SDK for .NET

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

namespace glacier.amazon.com.docsamples
{
  class ArchiveDeleteHighLevel_GettingStarted
  {
```

```
    static string vaultName = "examplevault";
    static string archiveId = "*** Provide archive ID ***";

    public static void Main(string[] args)
    {
      try
      {
        var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USWest2);
        manager.DeleteArchive(vaultName, archiveId);
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }
  }
}
```

Delete an Archive in Amazon Glacier by Using the AWS CLI

You can delete archives in Amazon Glacier by using the AWS Command Line Interface (AWS CLI).

Topics

- (Prerequisite) Setting Up the AWS CLI
- Example: Deleting an Archive by Using the AWS CLI

(Prerequisite) Setting Up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Installing the AWS Command Line Interface

   Configuring the AWS Command Line Interface

2. Verify your AWS CLI setup by entering the following commands at the command prompt. These commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try using the help command.

```
aws help
```

- To get a list of Amazon Glacier vaults on the configured account, use the `list-vaults` command. Replace *123456789012* with your AWS account ID.

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

- To see the current configuration data for the AWS CLI, use the `aws configure list` command.

```
aws configure list
```

Example: Deleting an Archive by Using the AWS CLI

1. Use the `initiate-job` command to start an inventory retrieval job. For more information on the `initiate-job` command, see Initiate Job.

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

Expected output:

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

2. Use the `describe-job` command to check the status of the previous retrieval job. For more information on the `describe-job` command, see Describe Job.

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

Expected output:

```
{
    "InventoryRetrievalParameters": {
        "Format": "JSON"
```

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

3. Wait for the job to be completed.

   You must wait until the job output is ready for you to download. If you set a notification configuration on the vault or specified an Amazon Simple Notification Service (Amazon SNS) topic when you initiated the job, Amazon Glacier sends a message to the topic after it completes the job.

   You can set notification configuration for specific events on the vault. For more information, see Configuring Vault Notifications in Amazon Glacier. Amazon Glacier sends a message to the specified Amazon SNS topic anytime the specific event occurs.

4. When the job is complete, use the `get-job-output` command to download the retrieval job to the file `output.json`. For more information on the `get-job-output` command, see Get Job Output.

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

   This command produces a file with the following fields.

   ```
   {
   "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. Use the `delete-archive` command to delete each archive from a vault until none remain.

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

For more information on the `delete-archive` command, see Delete Archive.

Step 6: Delete a Vault in Amazon Glacier

A vault is a container for storing archives. To delete an Amazon Glacier vault, you must first delete all existing archives in the vault as of the last inventory that Amazon Glacier computed.

You can delete a vault programmatically or by using the Amazon Glacier console. For information about deleting a vault programmatically, see Deleting a Vault in Amazon Glacier.

> **Important**
>
> If you upload an archive to a vault or delete an archive from a vault within the recent 24 hours, you must wait until the last vault inventory is updated to reflect the latest information. Amazon Glacier prepares an inventory for each vault periodically, every 24 hours.

To delete an empty vault

1. Sign in to the AWS Management Console and open the Amazon Glacier console at https://console.aws.amazon.com/glacier/home.
2. From the **Select a Region** menu, choose the AWS Region for the vault that you want to delete.

   In this getting started exercise, your example vault is in the US West (Oregon) Region.
3. Select the option button next to the empty vault that you want to delete. If the vault is not empty, you must delete all archives before deleting the vault. For more information, see Deleting an Archive in Amazon Glacier.

> **Important**
>
> Deleting a vault can't be undone.

4. Choose **Delete**.
5. The **Delete vault** dialog box appears. Choose **Delete**.

To delete a nonempty vault

1. If you're deleting a nonempty vault, you must first delete all existing archives before deleting the vault. You can do this by writing code to make a delete archive request by using either the REST API, the AWS SDK for Java, the AWS SDK for .NET or the AWS CLI. For information about deleting archives, see Step 5: Delete an Archive from a Vault in Amazon Glacier.
2. After the vault is empty, follow the steps to delete an empty vault in the preceding procedure.

Where Do I Go From Here?

Now that you have tried the getting started exercise, you can explore the following sections to learn more about Amazon Glacier.

- Working with Vaults in Amazon Glacier
- Working with Archives in Amazon Glacier

Working with Vaults in Amazon Glacier

A vault is a container for storing archives. When you create a vault, you specify a vault name and the AWS Region in which you want to create the vault. For a list of the AWS Regions supported by Amazon Glacier, see Amazon Glacier endpoints and quotas in the *AWS General Reference*.

You can store an unlimited number of archives in a vault.

Important

Amazon Glacier does provide a console. However, any archive operation, such as upload, download, or deletion, requires you to use the AWS Command Line Interface (AWS CLI) or write code. There is no console support for archive operations. For example, to upload data, such as photos, videos, and other documents, you must either use the AWS CLI or write code to make requests, by using either the REST API directly or by using the AWS SDKs. To install the AWS CLI, see AWS Command Line Interface. For more information about using Amazon Glacier with the AWS CLI, see the AWS CLI Reference for Amazon Glacier. For examples of using the AWS CLI to upload archives to Amazon Glacier, see Using Amazon Glacier with the AWS Command Line Interface.

Topics

Vault Operations in Amazon Glacier

Amazon Glacier supports various vault operations. Vault operations are specific to particular AWS Regions. In other words, when you create a vault, you create it in a specific AWS Region. When you list vaults, Amazon Glacier returns the vault list from the AWS Region that you specified in the request.

Creating and Deleting Vaults

An AWS account can create up to 1,000 vaults per AWS Region. For a list of the AWS Regions supported by Amazon Glacier, see Amazon Glacier endpoints and quotas in the *AWS General Reference*.

You can delete a vault only if there are no archives in the vault as of the last inventory that Amazon Glacier computed and if there have been no writes to the vault since the last inventory.

> **Note**
>
> Amazon Glacier prepares an inventory for each vault periodically, every 24 hours. Because the inventory might not reflect the latest information, Amazon Glacier ensures that the vault is indeed empty by checking if there were any write operations since the last vault inventory.

For more information, see Creating a Vault in Amazon Glacier and Deleting a Vault in Amazon Glacier.

Retrieving Vault Metadata

You can retrieve vault information such as the vault creation date, number of archives in the vault, and the total size of all the archives in the vault. Amazon Glacier provides API calls for you to retrieve this information for a specific vault or all the vaults in a specific AWS Region in your account. For more information, see Retrieving Vault Metadata in Amazon Glacier.

Downloading a Vault Inventory

A *vault inventory* refers to the list of archives in a vault. For each archive in the list, the inventory provides archive information, such as the archive ID, creation date, and size. Amazon Glacier

updates the vault inventory once a day, starting on the day that the first archive is uploaded to the vault. A vault inventory must exist for you to be able to download it.

Downloading a vault inventory is an asynchronous operation. You must first initiate a job to download the inventory. After receiving the job request, Amazon Glacier prepares your inventory for download. After the job is completed, you can download the inventory data.

Given the asynchronous nature of the job, you can use Amazon Simple Notification Service (Amazon SNS) notifications to notify you when the job is completed. You can specify an Amazon SNS topic for each individual job request or configure your vault to send a notification when specific vault events occur.

Amazon Glacier prepares an inventory for each vault periodically, every 24 hours. If there have been no archive additions or deletions to the vault since the last inventory, the inventory date is not updated.

When you initiate a job for a vault inventory, Amazon Glacier returns the last inventory that it generated, which is a point-in-time snapshot and not real-time data. You might not find it useful to retrieve vault inventory for each archive upload. However, suppose that you maintain a database on the client-side that contains metadata associated with the archives that you upload to Amazon Glacier. In that case, you might find the vault inventory useful to reconcile information in your database with the actual vault inventory.

For more information about retrieving a vault inventory, see Downloading a Vault Inventory in Amazon Glacier.

Configuring Vault Notifications

Retrieving anything from Amazon Glacier, such as an archive from a vault or a vault inventory, is a two-step process. First, you initiate a job. After the job is completed, you download the output. To learn when your job is complete, you can use Amazon Glacier notifications. Amazon Glacier sends notification messages to an Amazon Simple Notification Service (Amazon SNS) topic that you provide.

You can configure notifications on a vault and identify vault events and the Amazon SNS topic to be notified when the event occurs. Anytime the vault event occurs, Amazon Glacier sends a notification to the specified Amazon SNS topic. For more information, see Configuring Vault Notifications in Amazon Glacier.

Creating a Vault in Amazon Glacier

Creating a vault adds a vault to the set of vaults in your account. An AWS account can create up to 1,000 vaults per AWS Region. For a list of the AWS Regions supported by Amazon Glacier (Amazon Glacier), see Regions and Endpoints in the *AWS General Reference*.

When you create a vault, you must provide a vault name. The following are the vault naming requirements:

- Names can be between 1 and 255 characters long.
- Allowed characters are a–z, A–Z, 0–9, '_' (underscore), '-' (hyphen), and '.' (period).

Vault names must be unique within an account and the AWS Region in which the vault is being created. That is, an account can create vaults with the same name in different AWS Regions but not in the same AWS Region.

Topics

Creating a Vault in Amazon Glacier Using the AWS SDK for Java

The low-level API provides methods for all the vault operations, including creating and deleting vaults, getting a vault description, and getting a list of vaults created in a specific AWS Region. The following are the steps to create a vault using the AWS SDK for Java.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region in which you want to create a vault. All operations you perform using this client apply to that AWS Region.
2. Provide request information by creating an instance of the `CreateVaultRequest` class.

Amazon Glacier (Amazon Glacier) requires you to provide a vault name and your account ID. If you don't provide an account ID, then the account ID associated with the credentials you provide to sign the request is used. For more information, see Using the AWS SDK for Java with Amazon Glacier.

3. Run the `createVault` method by providing the request object as a parameter.

   The response Amazon Glacier returns is available in the `CreateVaultResult` object.

The following Java code snippet illustrates the preceding steps. The snippet creates a vault in the `us-west-2` Region. The `Location` it prints is the relative URI of the vault that includes your account ID, the AWS Region, and the vault name.

```
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

For information about the underlying REST API, see Create Vault (PUT vault).

Example: Creating a Vault Using the AWS SDK for Java

The following Java code example creates a vault in the `us-west-2` Region (for more information on AWS Regions, see Accessing Amazon Glacier). In addition, the code example retrieves the vault information, lists all vaults in the same AWS Region, and then deletes the vault created.

For step-by-step instructions on how to run the following example, see Running Java Examples for Amazon Glacier Using Eclipse.

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);
    }

}
```

Creating a Vault in Amazon Glacier Using the AWS SDK for .NET

Both the high-level and low-level APIs provided by the Amazon SDK for .NET provide a method to create a vault.

Topics

Creating a Vault Using the High-Level API of the AWS SDK for .NET

The `ArchiveTransferManager` class of the high-level API provides the `CreateVault` method you can use to create a vault in an AWS Region.

Example: Vault Operations Using the High-Level API of the AWS SDK for .NET

The following C# code example creates and delete a vault in the US West (Oregon) Region. For a list of AWS Regions in which you can create vaults, see Accessing Amazon Glacier.

For step-by-step instructions on how to run the following example, see Running Code Examples. You need to update the code as shown with a vault name.

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();
      }
    }
}
```

Creating a Vault Using the Low-Level API of the AWS SDK for .NET

The low-level API provides methods for all the vault operations, including create and delete vaults, get a vault description, and get a list of vaults created in a specific AWS Region. The following are the steps to create a vault using the AWS SDK for .NET.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region in which you want to create a vault. All operations you perform using this client apply to that AWS Region.
2. Provide request information by creating an instance of the `CreateVaultRequest` class.

   Amazon Glacier (Amazon Glacier) requires you to provide a vault name and your account ID. If you don't provide an account ID, then account ID associated with the credentials you provide to sign the request is assumed. For more information, see Using the AWS SDK for .NET with Amazon Glacier.
3. Run the `CreateVault` method by providing the request object as a parameter.

   The response Amazon Glacier returns is available in the `CreateVaultResponse` object.

Example: Vault Operations Using the Low-Level API of the AWS SDK for .NET

The following C# example illustrates the preceding steps. The example creates a vault in the US West (Oregon) Region. In addition, the code example retrieves the vault information, lists all vaults

in the same AWS Region, and then deletes the vault created. The `Location` printed is the relative URI of the vault that includes your account ID, the AWS Region, and the vault name.

> **Note**
>
> For information about the underlying REST API, see Create Vault (PUT vault).

For step-by-step instructions on how to run the following example, see Running Code Examples. You need to update the code as shown with a vault name.

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);
    }
  }
}
```

Creating a Vault in Amazon Glacier Using the REST API

To create a vault using the REST API, see Create Vault (PUT vault).

Creating a Vault Using the Amazon Glacier Console

To create a vault using the Amazon Glacier (Amazon Glacier) console, see Step 2: Create a Vault in Amazon Glacier in the *Getting Started* tutorial.

Creating a Vault in Amazon Glacier Using the AWS Command Line Interface

Follow these steps to create a vault in Amazon Glacier (Amazon Glacier) using the AWS Command Line Interface (AWS CLI).

Topics

- (Prerequisite) Setting Up the AWS CLI

- Example: Creating a Vault Using the AWS CLI

(Prerequisite) Setting Up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Installing the AWS Command Line Interface

   Configuring the AWS Command Line Interface

2. Verify your AWS CLI setup by entering the following commands at the command prompt. These commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try using the help command.

     ```
     aws help
     ```

   - To get a list of Amazon Glacier vaults on the configured account, use the `list-vaults` command. Replace *123456789012* with your AWS account ID.

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

   - To see the current configuration data for the AWS CLI, use the `aws configure list` command.

     ```
     aws configure list
     ```

Example: Creating a Vault Using the AWS CLI

1. Use the `create-vault` command to create a vault named *awsexamplevault* under account *111122223333*.

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

   Expected output:

   ```
   
   ```

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

2. Verify creation using the `describe-vault` command.

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

Retrieving Vault Metadata in Amazon Glacier

You can retrieve vault information such as the vault creation date, number of archives in the vault, and the total size of all the archives in the vault. Amazon Glacier (Amazon Glacier) provides API calls for you to retrieve this information for a specific vault or all the vaults in a specific AWS Region in your account.

If you retrieve a vault list, Amazon Glacier returns the list sorted by the ASCII values of the vault names. The list contains up to 1,000 vaults. You should always check the response for a marker at which to continue the list; if there are no more items the marker field is `null`. You can optionally limit the number of vaults returned in the response. If there are more vaults than are returned in the response, the result is paginated. You need to send additional requests to fetch the next set of vaults.

Topics

Retrieving Vault Metadata in Amazon Glacier Using the AWS SDK for Java

Topics

- Example: Retrieving Vault Metadata Using the Amazon SDK for Java

Retrieve Vault Metadata for a Vault

You can retrieve metadata for a specific vault or all the vaults in a specific AWS Region. The following are the steps to retrieve vault metadata for a specific vault using the low-level API of the Amazon SDK for Java.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where the vault resides. All operations you perform using this client apply to that AWS Region.
2. Provide request information by creating an instance of the `DescribeVaultRequest` class.

   Amazon Glacier (Amazon Glacier) requires you to provide a vault name and your account ID. If you don't provide an account ID, then the account ID associated with the credentials you provide to sign the request is assumed. For more information, see Using the AWS SDK for Java with Amazon Glacier.
3. Run the `describeVault` method by providing the request object as a parameter.

   The vault metadata information that Amazon Glacier returns is available in the `DescribeVaultResult` object.

The following Java code snippet illustrates the preceding steps.

```
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**
>
> For information about the underlying REST API, see Describe Vault (GET vault).

Retrieve Vault Metadata for All Vaults in a Region

You can also use the `listVaults` method to retrieve metadata for all the vaults in a specific AWS Region.

The following Java code snippet retrieves list of vaults in the `us-west-2` Region. The request limits the number of vaults returned in the response to 5. The code snippet then makes a series of `listVaults` calls to retrieve the entire vault list from the AWS Region.

```
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);
```

In the preceding code segment, if you don't specify the `Limit` value in the request, Amazon Glacier returns up to 10 vaults, as set by the Amazon Glacier API. If there are more vaults to list, the response `marker` field contains the vault Amazon Resource Name (ARN) at which to continue the list with a new request; otherwise, the `marker` field is null.

Note that the information returned for each vault in the list is the same as the information you get by calling the `describeVault` method for a specific vault.

> **Note**
>
> The `listVaults` method calls the underlying REST API (see List Vaults (GET vaults)).

Example: Retrieving Vault Metadata Using the Amazon SDK for Java

For a working code example, see Example: Creating a Vault Using the AWS SDK for Java. The Java code example creates a vault and retrieves the vault metadata.

Retrieving Vault Metadata in Amazon Glacier Using the AWS SDK for .NET

Topics

- Retrieve Vault Metadata for a Vault
- Retrieve Vault Metadata for All Vaults in a Region
- Example: Retrieving Vault Metadata Using the Low-Level API of the AWS SDK for .NET

Retrieve Vault Metadata for a Vault

You can retrieve metadata for a specific vault or all the vaults in a specific AWS Region. The following are the steps to retrieve vault metadata for a specific vault using the low-level API of the AWS SDK for .NET.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where the vault resides. All operations you perform using this client apply to that AWS Region.
2. Provide request information by creating an instance of the `DescribeVaultRequest` class.

   Amazon Glacier (Amazon Glacier) requires you to provide a vault name and your account ID. If you don't provide an account ID, then the account ID associated with the credentials you provide to sign the request is assumed. For more information, see Using the AWS SDK for .NET with Amazon Glacier.

3. Run the `DescribeVault` method by providing the request object as a parameter.

   The vault metadata information that Amazon Glacier returns is available in the `DescribeVaultResult` object.

The following C# code snippet illustrates the preceding steps. The snippet retrieves metadata information of an existing vault in the US West (Oregon) Region.

```
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

For information about the underlying REST API, see Describe Vault (GET vault).

Retrieve Vault Metadata for All Vaults in a Region

You can also use the `ListVaults` method to retrieve metadata for all the vaults in a specific AWS Region.

The following C# code snippet retrieves list of vaults in the US West (Oregon) Region. The request limits the number of vaults returned in the response to 5. The code snippet then makes a series of `ListVaults` calls to retrieve the entire vault list from the AWS Region.

```
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);
```

In the preceding code segment, if you don't specify the `Limit` value in the request, Amazon Glacier returns up to 10 vaults, as set by the Amazon Glacier API.

Note that the information returned for each vault in the list is the same as the information you get by calling the `DescribeVault` method for a specific vault.

> **Note**
>
> The `ListVaults` method calls the underlying REST API (see List Vaults (GET vaults)).

Example: Retrieving Vault Metadata Using the Low-Level API of the AWS SDK for .NET

For a working code example, see Example: Vault Operations Using the Low-Level API of the AWS SDK for .NET. The C# code example creates a vault and retrieves the vault metadata.

Retrieving Vault Metadata Using the REST API

To list vaults using the REST API, see List Vaults (GET vaults). To describe one vault, see Describe Vault (GET vault).

Retrieving Vault Metadata in Amazon Glacier Using the AWS Command Line Interface

This example shows how to retrieve vault information and metadata in Amazon Glacier (Amazon Glacier) using the AWS Command Line Interface (AWS CLI).

Topics

(Prerequisite) Setting Up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Installing the AWS Command Line Interface

   Configuring the AWS Command Line Interface

2. Verify your AWS CLI setup by entering the following commands at the command prompt. These commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try using the help command.

     ```
     aws help
     ```

   - To get a list of Amazon Glacier vaults on the configured account, use the `list-vaults` command. Replace *123456789012* with your AWS account ID.

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

   - To see the current configuration data for the AWS CLI, use the `aws configure list` command.

```
aws configure list
```

Example: Retrieving Vault Metadata Using the AWS CLI

- Use the `describe-vault` command to describe a vault named *awsexamplevault* under account *111122223333*.

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

Downloading a Vault Inventory in Amazon Glacier

After you upload your first archive to your vault, Amazon Glacier (Amazon Glacier) automatically creates a vault inventory and then updates it approximately once a day. After Amazon Glacier creates the first inventory, it typically takes half a day and up to a day before that inventory is available for retrieval. You can retrieve a vault inventory from Amazon Glacier with the following two-step process:

1. Initiate an inventory retrieval job by using the Initiate Job (POST jobs) operation.

> **Important**
>
> A data retrieval policy can cause your initiate retrieval job request to fail with a `PolicyEnforcedException` exception. For more information about data retrieval policies, see Amazon Glacier Data Retrieval Policies. For more information about the `PolicyEnforcedException` exception, see Error Responses.

2. After the job completes, download the bytes using the Get Job Output (GET output) operation.

For example, retrieving an archive or a vault inventory requires you to first initiate a retrieval job. The job request is run asynchronously. When you initiate a retrieval job, Amazon Glacier creates a job and returns a job ID in the response. When Amazon Glacier completes the job, you can get the job output, the archive bytes, or the vault inventory data.

The job must complete before you can get its output. To determine the status of the job, you have the following options:

- **Wait for job completion notification**—You can specify an Amazon Simple Notification Service (Amazon SNS) topic to which Amazon Glacier can post a notification after the job is completed. You can specify Amazon SNS topic using the following methods:
  - Specify an Amazon SNS topic per job basis.

    When you initiate a job, you can optionally specify an Amazon SNS topic.
  - Set notification configuration on the vault.

    You can set notification configuration for specific events on the vault (see Configuring Vault Notifications in Amazon Glacier). Amazon Glacier sends a message to the specified SNS topic any time the specific event occur.

  If you have notification configuration set on the vault and you also specify an Amazon SNS topic when you initiate a job, Amazon Glacier sends job completion message to both the topics.

  You can configure the SNS topic to notify you via email or store the message in an Amazon Simple Queue Service (Amazon SQS) that your application can poll. When a message appears in the queue, you can check if the job is completed successfully and then download the job output.
- **Request job information explicitly**—Amazon Glacier also provides a describe job operation (Describe Job (GET JobID)) that enables you to poll for job information. You can periodically send this request to obtain job information. However, using Amazon SNS notifications is the recommended option.

> **Note**
>
> The information you get via SNS notification is the same as what you get by calling Describe Job.

Topics

About the Inventory

Amazon Glacier updates a vault inventory at least once per day, starting on the day you first upload an archive to the vault. If there have been no archive additions or deletions to the vault since the last inventory, the inventory date is not updated. When you initiate a job for a vault inventory, Amazon Glacier returns the last inventory it generated, which is a point-in-time snapshot and not real-time data. Note that after Amazon Glacier creates the first inventory for the vault, it typically takes half a day and up to a day before that inventory is available for retrieval.

You might not find it useful to retrieve a vault inventory for each archive upload. However, suppose you maintain a database on the client-side associating metadata about the archives you upload to Amazon Glacier. Then, you might find the vault inventory useful to reconcile information, as needed, in your database with the actual vault inventory. You can limit the number of inventory items retrieved by filtering on the archive creation date or by setting a quota. For more information about limiting inventory retrieval, see Range Inventory Retrieval.

The inventory can be returned in two formats, comma-separated values (CSV) or JSON. You can optionally specify the format when you initiate the inventory job. The default format is JSON. For more information about the data fields returned in an inventory job output, see Response Body of the *Get Job Output API*.

Downloading a Vault Inventory in Amazon Glacier Using the AWS SDK for Java

The following are the steps to retrieve a vault inventory using the low-level API of the AWS SDK for Java. The high-level API does not support retrieving a vault inventory.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where the vault resides. All operations you perform using this client apply to that AWS Region.

2. Initiate an inventory retrieval job by executing the `initiateJob` method.

   Run `initiateJob` by providing job information in an `InitiateJobRequest` object.

> **Note**
>
> Note that if an inventory has not been completed for the vault an error is returned. Amazon Glacier (Amazon Glacier) prepares an inventory for each vault periodically, every 24 hours.

Amazon Glacier returns a job ID in response. The response is available in an instance of the `InitiateJobResult` class.

```
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. Wait for the job to complete.

   You must wait until the job output is ready for you to download. If you have either set a notification configuration on the vault, or specified an Amazon Simple Notification Service (Amazon SNS) topic when you initiated the job, Amazon Glacier sends a message to the topic after it completes the job.

   You can also poll Amazon Glacier by calling the `describeJob` method to determine job completion status. However, using an Amazon SNS topic for notification is the recommended approach. The code example given in the following section uses Amazon SNS for Amazon Glacier to publish a message.

4. Download the job output (vault inventory data) by executing the `getJobOutput` method.

   You provide your account ID, job ID, and vault name by creating an instance of the `GetJobOutputRequest` class. If you don't provide an account ID, then the account ID

associated with the credentials you provide to sign the request is used. For more information, see Using the AWS SDK for Java with Amazon Glacier.

The output that Amazon Glacier returns is available in the `GetJobOutputResult` object.

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

Note

For information about the job related underlying REST API, see Job Operations.

Example: Retrieving a Vault Inventory Using the Amazon SDK for Java

The following Java code example retrieves the vault inventory for the specified vault.

The example performs the following tasks:

- Creates an Amazon Simple Notification Service (Amazon SNS) topic.

  Amazon Glacier sends notification to this topic after it completes the job.
- Creates an Amazon Simple Queue Service (Amazon SQS) queue.

  The example attaches a policy to the queue to enable the Amazon SNS topic to post messages to the queue.
- Initiates a job to download the specified archive.

  In the job request, the Amazon SNS topic that was created is specified so that Amazon Glacier can publish a notification to the topic after it completes the job.
- Checks the Amazon SQS queue for a message that contains the job ID.

  If there is a message, parse the JSON and check if the job completed successfully. If it did, download the archive.

- Cleans up by deleting the Amazon SNS topic and the Amazon SQS queue that it created.

```
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));
    }
}
```

Downloading a Vault Inventory in Amazon Glacier Using the AWS SDK for .NET

The following are the steps to retrieve a vault inventory using the low-level API of the AWS SDK for .NET. The high-level API does not support retrieving a vault inventory.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where the vault resides. All operations you perform using this client apply to that AWS Region.

2. Initiate an inventory retrieval job by executing the `InitiateJob` method.

   You provide job information in an `InitiateJobRequest` object. Amazon Glacier (Amazon Glacier) returns a job ID in response. The response is available in an instance of the `InitiateJobResponse` class.

   ```
   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. Wait for the job to complete.

   You must wait until the job output is ready for you to download. If you have either set a notification configuration on the vault identifying an Amazon Simple Notification Service

(Amazon SNS) topic, or specified an Amazon SNS topic when you initiated a job, Amazon Glacier sends a message to that topic after it completes the job. The code example given in the following section uses Amazon SNS for Amazon Glacier to publish a message.

You can also poll Amazon Glacier by calling the `DescribeJob` method to determine job completion status. Although using Amazon SNS topic for notification is the recommended approach.

4. Download the job output (vault inventory data) by executing the `GetJobOutput` method.

   You provide your account ID, vault name, and the job ID information by creating an instance of the `GetJobOutputRequest` class. If you don't provide an account ID, then the account ID associated with the credentials you provide to sign the request is assumed. For more information, see Using the AWS SDK for .NET with Amazon Glacier.

   The output that Amazon Glacier returns is available in the `GetJobOutputResponse` object.

```
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**
>
> For information about the job related underlying REST API, see Job Operations.

Example: Retrieving a Vault Inventory Using the Low-Level API of the AWS SDK for .NET

The following C# code example retrieves the vault inventory for the specified vault.

The example performs the following tasks:

- Set up an Amazon SNS topic.

  Amazon Glacier sends notification to this topic after it completes the job.
- Set up an Amazon SQS queue.

  The example attaches a policy to the queue to enable the Amazon SNS topic to post messages.
- Initiate a job to download the specified archive.

  In the job request, the example specifies the Amazon SNS topic so that Amazon Glacier can send a message after it completes the job.
- Periodically check the Amazon SQS queue for a message.

  If there is a message, parse the JSON and check if the job completed successfully. If it did, download the archive. The code example uses the JSON.NET library (see JSON.NET) to parse the JSON.
- Clean up by deleting the Amazon SNS topic and the Amazon SQS queue it created.

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\",&TCX5-2025-waiver;" +
        "    \"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);
      }
    }
  }
}
```

Downloading a Vault Inventory Using the REST API

To download a vault inventory using the REST API

Downloading a vault inventory is a two-step process.

1. Initiate a job of the `inventory-retrieval` type. For more information, see Initiate Job (POST jobs).
2. After the job completes, download the inventory data. For more information, see Get Job Output (GET output).

Downloading a Vault Inventory in Amazon Glacier Using the AWS Command Line Interface

Follow these steps to download a vault inventory in Amazon Glacier (Amazon Glacier) using the AWS Command Line Interface (AWS CLI).

Topics

(Prerequisite) Setting Up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Installing the AWS Command Line Interface

   Configuring the AWS Command Line Interface

2. Verify your AWS CLI setup by entering the following commands at the command prompt. These commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try using the help command.

     ```
     aws help
     ```

   - To get a list of Amazon Glacier vaults on the configured account, use the `list-vaults` command. Replace *123456789012* with your AWS account ID.

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

   - To see the current configuration data for the AWS CLI, use the `aws configure list` command.

     ```
     aws configure list
     ```

Example: Downloading a Vault Inventory Using the AWS CLI

1. Use the `initiate-job` command to start an inventory retrieval job.

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

Expected output:

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

2. Use the `describe-job` command to check status of the previous retrieval job.

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

Expected output:

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

3. Wait for the job to complete.

You must wait until the job output is ready for you to download. The job ID does not expire for at least 24 hours after Amazon Glacier completes the job. If you have either set a notification

configuration on the vault, or specified an Amazon Simple Notification Service (Amazon SNS) topic when you initiated the job, Amazon Glacier sends a message to the topic after it completes the job.

You can set the notification configuration for specific events on the vault. For more information, see Configuring Vault Notifications in Amazon Glacier. Amazon Glacier sends a message to the specified SNS topic anytime the specific events occur.

4. When it's complete, use the `get-job-output` command to download the retrieval job to the file `output.json`.

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

This command produces a file with the following fields.

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

Configuring Vault Notifications in Amazon Glacier

Retrieving anything from Amazon Glacier, such as an archive from a vault or a vault inventory, is a two-step process.

1. Initiate a retrieval job.
2. After the job is completed, download the job output.

You can set a notification configuration on a vault so that when a job is completed, a message is sent to an Amazon Simple Notification Service (Amazon SNS) topic.

Topics

Configuring Vault Notifications in Amazon Glacier: General Concepts

A Amazon Glacier retrieval job request is run asynchronously. You must wait until Amazon Glacier completes the job before you can get its output. You can periodically poll Amazon Glacier to determine the job status, but that is not an optimal approach. Amazon Glacier also supports notifications. When a job is completed, the job can post a message to an Amazon Simple Notification Service (Amazon SNS) topic. Using this feature requires you to set a notification configuration on the vault. In the configuration, you identify one or more events and an Amazon SNS topic to which you want Amazon Glacier to send a message when the event occurs.

Amazon Glacier defines events specifically related to job completion (`ArchiveRetrievalCompleted`, `InventoryRetrievalCompleted`) that you can add to the vault's notification configuration. When a specific job is completed, Amazon Glacier publishes a notification message to the SNS topic.

The notification configuration is a JSON document as shown in the following example.

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

You can configure only one Amazon SNS topic for a vault.

> **Note**
>
> Adding a notification configuration to a vault causes Amazon Glacier to send a notification each time the event specified in the notification configuration occurs. You can also optionally specify an Amazon SNS topic in each job initiation request. If you add both the notification configuration on the vault and also specify an Amazon SNS topic in your initiate job request, Amazon Glacier sends both notifications.

The job completion message Amazon Glacier sends include information such as the type of job (`InventoryRetrieval`, `ArchiveRetrieval`), job completion status, SNS topic name, job status code, and the vault ARN. The following is an example notification Amazon Glacier sent to an SNS topic after an `InventoryRetrieval` job is completed.

```
{
 "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"
}
```

If the `Completed` field is true, you must also check the `StatusCode` to check if the job completed successfully or failed.

> **Note**
>
> The Amazon SNS topic must allow the vault to publish a notification. By default, only the Amazon SNS topic owner can publish a message to the topic. However, if the Amazon SNS topic and the vault are owned by different AWS accounts, then you must configure the

Amazon SNS topic to accept publications from the vault. You can configure the Amazon SNS topic policy in the Amazon SNS console.

For more information about Amazon SNS, see Getting Started with Amazon SNS.

Configuring Vault Notifications in Amazon Glacier Using the AWS SDK for Java

The following are the steps to configure notifications on a vault using the low-level API of the AWS SDK for Java.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where the vault resides. All operations you perform using this client apply to that AWS Region.
2. Provide notification configuration information by creating an instance of the `SetVaultNotificationsRequest` class.

   You need to provide the vault name, notification configuration information, and account ID. In specifying a notification configuration, you provide the Amazon Resource Name (ARN) of an existing Amazon SNS topic and one or more events for which you want to be notified. For a list of supported events, see Set Vault Notification Configuration (PUT notification-configuration)).
3. Run the `setVaultNotifications` method by providing the request object as a parameter.

The following Java code snippet illustrates the preceding steps. The snippet sets a notification configuration on a vault. The configuration requests Amazon Glacier (Amazon Glacier) to send a notification to the specified Amazon SNS topic when either the `ArchiveRetrievalCompleted` event or the `InventoryRetrievalCompleted` event occurs.

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

```
        );
client.setVaultNotifications(request);
```

> **Note**
>
> For information about the underlying REST API, see Vault Operations.

Example: Setting the Notification Configuration on a Vault Using the AWS SDK for Java

The following Java code example sets a vault's notifications configuration, deletes the configuration, and then restores the configuration. For step-by-step instructions on how to run the following example, see Using the AWS SDK for Java with Amazon Glacier.

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);
    }
}
```

Configuring Vault Notifications in Amazon Glacier Using the AWS SDK for .NET

The following are the steps to configure notifications on a vault using the low-level API of the AWS SDK for .NET.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where the vault resides. All operations you perform using this client apply to that AWS Region.
2. Provide notification configuration information by creating an instance of the `SetVaultNotificationsRequest` class.

   You need to provide the vault name, notification configuration information, and account ID. If you don't provide an account ID, then the account ID associated with the credentials you provide to sign the request is assumed. For more information, see Using the AWS SDK for .NET with Amazon Glacier.

   In specifying a notification configuration, you provide the Amazon Resource Name (ARN) of an existing Amazon SNS topic and one or more events for which you want to be notified. For a list of supported events, see Set Vault Notification Configuration (PUT notification-configuration)).
3. Run the `SetVaultNotifications` method by providing the request object as a parameter.
4. After setting notification configuration on a vault, you can retrieve configuration information by calling the `GetVaultNotifications` method, and remove it by calling the `DeleteVaultNotifications` method provided by the client.

Example: Setting the Notification Configuration on a Vault Using the AWS SDK for .NET

The following C# code example illustrates the preceding steps. The example sets the notification configuration on the vault ("`examplevault`") in the US West (Oregon) Region, retrieves the configuration, and then deletes it. The configuration requests Amazon Glacier

(Amazon Glacier) to send a notification to the specified Amazon SNS topic when either the `ArchiveRetrievalCompleted` event or the `InventoryRetrievalCompleted` event occurs.

> **Note**
>
> For information about the underlying REST API, see Vault Operations.

For step-by-step instructions to run the following example, see Running Code Examples. You need to update the code as shown and provide an existing vault name and an Amazon SNS topic.

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);
```

```
    }
  }
}
```

Configuring Vault Notifications in Amazon Glacier Using the REST API

To configure vault notifications using the REST API, see Set Vault Notification Configuration (PUT notification-configuration). Additionally, you can also get vault notifications (Get Vault Notifications (GET notification-configuration)) and delete vault notifications (Delete Vault Notifications (DELETE notification-configuration)).

Configuring Vault Notifications by Using the Amazon Glacier Console

This section describes how to configure vault notifications by using the Amazon Glacier console. When you configure notifications, you specify job-completion events that send a notification to an Amazon Simple Notification Service (Amazon SNS) topic. In addition to configuring notifications for the vault, you can also specify a topic to publish notifications to when you initiate a job. If your vault is configured to send a notification for a specific event and you also configure notifications in the job-initiation request, then two notifications are sent.

To configure a vault notification

1. Sign in to the AWS Management Console and open the Amazon Glacier console at https://console.aws.amazon.com/glacier/home.
2. In the left navigation pane, choose **Vaults**.
3. In the **Vaults** list, choose a vault.
4. In the **Notifications** section, choose **Edit**.
5. On the **Event notifications** page, choose **Turn on notifications**.
6. In the **Notifications** section, choose one of the following Amazon Simple Notification Service (Amazon SNS) options, and then follow the corresponding steps:

Amazon SNS options	Action
Create new SNS topic	1. Choose **Create new SNS topic**. 2. For **Topic name**, enter the name of the new topic.

Amazon SNS options	Action
	Topic names can be up to 256 characters. Alphanumeric characters, hyphens (-), and und erscores (_) are allowed. Topic names must be unique within the account and AWS Region. 3. (Optional) If you want to subscribe to the topic by using SMS messages, enter a name for **Display name**. A display name can have up to 100 characters.
Choose an existing SNS topic	1. Choose **Choose an existing SNS topic**. 2. Under **Specify SNS topic**, choose one of the fol lowing options: • **Choose from your SNS topics** An **SNS topic** dropdown list appears. Choose an existing topic from the dropdown list. • **Enter SNS topic ARN** An **Amazon SNS topic ARN** text box appears. Enter the Amazon Resource Name (ARN) for your SNS topic. An SNS topic ARN has the following format: `arn:aws:sns:` *`region`*`:`*`account-id`*`:`*`topic-name`* You can find the SNS topic ARN in the Amazon SNS console.

7. Under **Events**, select one or both events that you want to send notifications:

   - To send a notification only when archive retrieval jobs are complete, select **Archive Retrieval Job Complete**.
   - To send a notification only when vault inventory jobs are complete, select **Vault Inventory Retrieval Job Complete**.

Configuring Vault Notifications Using the AWS Command Line Interface

This section describes how to configure vault notifications using the AWS Command Line Interface. When you configure notifications, you specify job completion events that trigger notification to an Amazon Simple Notification Service (Amazon SNS) topic. In addition to configuring notifications for the vault, you can also specify a topic to publish notification to when you initiate a job. If your vault is configured to notify for a specific event and you specify notification in the job initiation request, then two notifications are sent.

Follow these steps to configure vault notification using the AWS CLI.

Topics

(Prerequisite) Setting Up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Installing the AWS Command Line Interface

   Configuring the AWS Command Line Interface

2. Verify your AWS CLI setup by entering the following commands at the command prompt. These commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try using the help command.

     ```
     aws help
     ```

- To get a list of Amazon Glacier vaults on the configured account, use the `list-vaults` command. Replace *123456789012* with your AWS account ID.

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

- To see the current configuration data for the AWS CLI, use the `aws configure list` command.

```
aws configure list
```

Example: Configure Vault Notifications Using the AWS CLI

1. Use the `set-vault-notifications` command to configure notifications that will be sent when specific events happen to a vault. By default, you don't get any notifications.

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

2. The notification configuration is a JSON document as shown in the following example.

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

   For more information about using Amazon SNS topics for Amazon Glacier see, Configuring Vault Notifications in Amazon Glacier: General Concepts

   For more information about Amazon SNS, see Getting Started with Amazon SNS.

Deleting a Vault in Amazon Glacier

Amazon Glacier (Amazon Glacier) deletes a vault only if there are no archives in the vault as of the last inventory it computed and there have been no writes to the vault since the last inventory. For information about deleting archives, see Deleting an Archive in Amazon Glacier. For information about downloading a vault inventory, Downloading a Vault Inventory in Amazon Glacier.

> **Note**
>
> Amazon Glacier prepares an inventory for each vault periodically, every 24 hours. Because the inventory might not reflect the latest information, Amazon Glacier ensures the vault is indeed empty by checking if there were any write operations since the last vault inventory.

> **Note**
>
> For automated deletion of vault archives, see Automated deletion of vault archives in Amazon S3 Glacier.

Topics

Deleting a Vault in Amazon Glacier Using the AWS SDK for Java

The following are the steps to delete a vault using the low-level API of the AWS SDK for Java.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region from where you want to delete a vault. All operations you perform using this client apply to that AWS Region.

2. Provide request information by creating an instance of the `DeleteVaultRequest` class.

   You need to provide the vault name and account ID. If you don't provide an account ID, then account ID associated with the credentials you provide to sign the request is assumed. For more information, see Using the AWS SDK for Java with Amazon Glacier.

3. Run the `deleteVault` method by providing the request object as a parameter.

Amazon Glacier (Amazon Glacier) deletes the vault only if it is empty. For more information, see Delete Vault (DELETE vault).

The following Java code snippet illustrates the preceding steps.

```
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

For information about the underlying REST API, see Delete Vault (DELETE vault).

Example: Deleting a Vault Using the AWS SDK for Java

For a working code example, see Example: Creating a Vault Using the AWS SDK for Java. The Java code example shows basic vault operations including create and delete vault.

Deleting a Vault in Amazon Glacier Using the AWS SDK for .NET

Both the high-level and low-level APIs provided by the Amazon SDK for .NET provide a method to delete a vault.

Topics

- Deleting a Vault Using the High-Level API of the AWS SDK for .NET
- Deleting a Vault Using the Low-Level API of the AWS SDK for .NET

Deleting a Vault Using the High-Level API of the AWS SDK for .NET

The `ArchiveTransferManager` class of the high-level API provides the `DeleteVault` method you can use to delete a vault.

Example: Deleting a Vault Using the High-Level API of the AWS SDK for .NET

For a working code example, see Example: Vault Operations Using the High-Level API of the AWS SDK for .NET. The C# code example shows basic vault operations including create and delete vault.

Deleting a Vault Using the Low-Level API of the AWS SDK for .NET

The following are the steps to delete a vault using the AWS SDK for .NET.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region from where you want to delete a vault. All operations you perform using this client apply to that AWS Region.
2. Provide request information by creating an instance of the `DeleteVaultRequest` class.

   You need to provide the vault name and account ID. If you don't provide an account ID, then account ID associated with the credentials you provide to sign the request is assumed. For more information, see Using the AWS SDK for .NET with Amazon Glacier.
3. Run the `DeleteVault` method by providing the request object as a parameter.

   Amazon Glacier (Amazon Glacier) deletes the vault only if it is empty. For more information, see Delete Vault (DELETE vault).

The following C# code snippet illustrates the preceding steps. The snippet retrieves metadata information of a vault that exists in the default AWS Region.

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

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

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

> **Note**
>
> For information about the underlying REST API, see Delete Vault (DELETE vault).

Example: Deleting a Vault Using the Low-Level API of the AWS SDK for .NET

For a working code example, see Example: Vault Operations Using the Low-Level API of the AWS SDK for .NET. The C# code example shows basic vault operations including create and delete vault.

Deleting a Vault in Amazon Glacier Using the REST API

To delete a vault using the REST API, see Delete Vault (DELETE vault).

Deleting an Empty Vault by Using the Amazon Glacier Console

> **Note**
>
> Before deleting a vault, you must delete all existing archives within the vault. You can do this by writing code to make a delete archive request by using either the REST API, the AWS SDK for Java, the AWS SDK for .NET, or by using the AWS Command Line Interface (AWS CLI). For information about deleting archives, see Step 5: Delete an Archive from a Vault in Amazon Glacier.

After your vault is empty, you can delete it by using the following steps.

To delete an empty vault by using the Amazon Glacier console

1. Sign into the AWS Management Console and open the Amazon Glacier console at Amazon Glacier Console.
2. Under **Select a Region**, choose the AWS Region where the vault exists.
3. In the left navigation pane, choose **Vaults**.
4. In the **Vaults** list, select the option button next to the name of the vault that you want to delete, and then choose **Delete** at the top of the page.
5. In the **Delete vault** dialog box, confirm that you want to delete the vault by choosing **Delete**.

> **Important**
>
> Deleting a vault can't be undone.

6. To verify that you've deleted the vault, open the **Vaults** list and enter the name of the vault that you deleted. If the vault can't be found, your deletion was successful.

Deleting a Vault in Amazon Glacier Using the AWS Command Line Interface

You can delete empty and nonempty vaults in Amazon Glacier (Amazon Glacier) using the AWS Command Line Interface (AWS CLI).

Topics

(Prerequisite) Setting Up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Installing the AWS Command Line Interface

   Configuring the AWS Command Line Interface

2. Verify your AWS CLI setup by entering the following commands at the command prompt. These commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try using the help command.

     ```
     aws help
     ```

   - To get a list of Amazon Glacier vaults on the configured account, use the `list-vaults` command. Replace *123456789012* with your AWS account ID.

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

- To see the current configuration data for the AWS CLI, use the `aws configure list` command.

```
aws configure list
```

Example: Deleting an Empty Vault Using the AWS CLI

- Use the `delete-vault` command to delete a vault that contains no archives.

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

Example: Deleting a Nonempty Vault Using the AWS CLI

Amazon Glacier deletes a vault only if there are no archives in the vault as of the last inventory it computed, and there have been no writes to the vault since the last inventory. Deleting a nonempty vault is a three-step process: retrieving archive IDs from a vault's inventory report, deleting each archive, and then deleting the vault.

1. Use the `initiate-job` command to start an inventory retrieval job.

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

   Expected output:

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

2. Use the `describe-job` command to check status of the previous retrieval job.
   ```

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

Expected output:

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

3. Wait for the job to complete.

   You must wait until the job output is ready for you to download. If you set a notification configuration on the vault or specified an Amazon Simple Notification Service (Amazon SNS) topic when you initiated the job, Amazon Glacier sends a message to the topic after it completes the job.

   You can set notification configuration for specific events on the vault. For more information, see Configuring Vault Notifications in Amazon Glacier. Amazon Glacier sends a message to the specified SNS topic anytime the specific event occurs.

4. When it's complete, use the `get-job-output` command to download the retrieval job to the file `output.json`.

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

   This command produces a file with the following fields.

   ```
   {
   ```

```
"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. Use the `delete-archive` command to delete each archive from a vault until none remain.

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

> **Note**
>
> If your archive ID starts with a hyphen or another special character, you must enclose the archive ID in quotation marks in order to run this command.

6. Use the `initiate-job` command to start a new inventory retrieval job.

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

7. When it's complete, use the `delete-vault` command to delete a vault with no archives.

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

Tagging Your Amazon Glacier Vaults

You can assign your own metadata to Amazon Glacier vaults in the form of tags. A *tag* is a key-value pair that you define for a vault. For basic information about tagging, including restrictions on tags, see Tagging Amazon Glacier Resources.

The following topics describe how you can add, list, and remove tags for vaults.

Topics

Tagging Vaults by Using the Amazon Glacier Console

You can add, list, and remove tags using the Amazon Glacier console, as described in the following procedures.

To view the tags for a vault

1. Sign in to the AWS Management Console and open the Amazon Glacier console at https://console.aws.amazon.com/glacier/home.
2. Under **Select a Region**, select an AWS Region from the Region selector.
3. In the left navigation pane, choose **Vaults**.
4. In the **Vaults** list, choose a vault.
5. Choose the **Vaults properties** tab. Scroll to the **Tags** section to view the tags associated with the vault.

To add a tag to a vault

You can associate up to 50 tags to a vault. Tags that are associated with a vault must have unique tag keys.

For more information about tag restrictions, see Tagging Amazon Glacier Resources.

1. Sign in to the AWS Management Console and open the Amazon Glacier console at https://console.aws.amazon.com/glacier/home.
2. Under **Select a Region**, select an AWS Region from the Region selector.
3. In the left navigation pane, choose **Vaults**.
4. In the **Vaults** list, choose the name of the vault that you want to add tags to.
5. Choose the **Vault properties** tab.
6. In the **Tags** section, choose **Add**. The **Add tags** page appears.

7. On the **Add tags** page, specify the tag key in the **Key** field, and optionally specify a tag value in the **Value** field.
8. Choose **Save changes**.

To edit a tag

1. Sign in to the AWS Management Console and open the Amazon Glacier console at https://console.aws.amazon.com/glacier/home.
2. Under **Select a Region**, select an AWS Region from the Region selector.
3. In the left navigation pane, choose **Vaults**.
4. In the **Vaults** list, choose a vault name.
5. Choose the **Vault properties** tab, and then scroll down to the **Tags** section.
6. Under **Tags**, select the check box next to the tags that you want to change, then choose **Edit**. The **Edit tags** page appears.
7. Update the tag key in the **Key** field, and optionally update the tag value in the **Value** field.
8. Choose **Save changes**.

To remove a tag from a vault

1. Sign in to the AWS Management Console and open the Amazon Glacier console at https://console.aws.amazon.com/glacier/home.
2. Under **Select a Region**, select an AWS Region from the Region selector.
3. In the left navigation pane, choose **Vaults**.
4. In the **Vaults** list, choose the name of the vault that you want to remove tags from.
5. Choose the **Vault properties** tab. Scroll down to the **Tags** section.
6. Under **Tags**, select the check box next to the tags that you want to remove, then choose **Delete**.
7. The **Delete tags** dialog box opens. To confirm that you want to delete the selected tags, choose **Delete**.

Tagging Vaults by Using the AWS CLI

Follow these steps to add, list, or remove tags by using the AWS Command Line Interface (AWS CLI).

Each tag is composed of a key and a value. Each vault can have up to 50 tags.

1. To add tags to a vault, use the `add-tags-to-vault` command.

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

   For more information on this vault operation, see Add Tags To Vault .

2. To list all the tags attached to a vault, use the `list-tags-for-vault` command.

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

   For more information on this vault operation, see List Tags For Vault.

3. To remove one or more tags from the set of tags attached to a vault, use the `remove-tags-from-vault` command.

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

   For more information on this vault operation, see Remove Tags From Vault.

Tagging Vaults by Using the Amazon Glacier API

You can add, list, and remove tags by using the Amazon Glacier API. For examples, see the following documentation:

Add Tags To Vault (POST tags add)

Adds or updates tags for the specified vault.

List Tags For Vault (GET tags)

Lists the tags for the specified vault.

Remove Tags From Vault (POST tags remove)

Removes tags from the specified vault.

Related Sections

- Tagging Amazon Glacier Resources

Amazon Glacier Vault Lock

The following topics describe how to lock a vault in Amazon Glacier and how to use Vault Lock policies.

Topics

- Vault Locking Overview
- Locking a Vault by Using the Amazon Glacier API
- Locking a Vault using the AWS Command Line Interface
- Locking a Vault by Using the Amazon Glacier Console

Vault Locking Overview

Amazon Glacier Vault Lock helps you to easily deploy and enforce compliance controls for individual Amazon Glacier vaults with a Vault Lock policy. You can specify controls such as "write once read many" (WORM) in a Vault Lock policy and lock the policy from future edits.

> **Important**
>
> After a Vault Lock policy is locked, the policy can no longer be changed or deleted.

Amazon Glacier enforces the controls set in the Vault Lock policy to help achieve your compliance objectives. For example, you can use Vault Lock policies to enforce data retention. You can deploy a variety of compliance controls in a Vault Lock policy by using the AWS Identity and Access Management (IAM) policy language. For more information about Vault Lock policies, see Vault Lock Policies.

A Vault Lock policy is different from a vault access policy. Both policies govern access controls to your vault. However, a Vault Lock policy can be locked to prevent future changes, which provides strong enforcement for your compliance controls. You can use the Vault Lock policy to deploy regulatory and compliance controls, which typically require tight controls on data access.

> **Important**
>
> We recommend that you first create a vault, complete a Vault Lock policy, and then upload your archives to the vault so that the policy will be applied to them.

In contrast, you use a vault access policy to implement access controls that are not compliance related, temporary, and subject to frequent modification. You can use Vault lock and vault access policies together. For example, you can implement time-based data-retention rules in the Vault Lock policy (deny deletes), and grant read access to designated third parties or your business partners (allow reads) in your vault access policy.

Locking a vault takes two steps:

1. Initiate the lock by attaching a Vault Lock policy to your vault, which sets the lock to an in-progress state and returns a lock ID. While the policy is in the in-progress state, you have 24 hours to validate your Vault Lock policy before the lock ID expires. To prevent your vault from exiting the in-progress state, you must complete the Vault Lock process within these 24 hours. Otherwise, your Vault Lock policy will be deleted.
2. Use the lock ID to complete the lock process. If the Vault Lock policy doesn't work as expected, you can stop the Vault Lock process and restart from the beginning. For information about how to use the Amazon Glacier API to lock a vault, see Locking a Vault by Using the Amazon Glacier API.

Locking a Vault by Using the Amazon Glacier API

To lock your vault with the Amazon Glacier API, you first call Initiate Vault Lock (POST lock-policy) with a Vault Lock policy that specifies the controls that you want to deploy. The `Initiate Vault Lock` operation attaches the policy to your vault, transitions the Vault Lock to the in-progress state, and returns a unique lock ID. After the Vault Lock enters the in-progress state, you have 24 hours to complete the lock by calling Complete Vault Lock (POST lockId) with the lock ID that was returned from the `Initiate Vault Lock` call.

Important

- We recommend that you first create a vault, complete a Vault Lock policy, and then upload your archives to the vault so that the policy will be applied to them.
- After the Vault Lock policy is locked, it cannot be changed or deleted.

If you don't complete the Vault Lock process within 24 hours after entering the in-progress state, your vault automatically exits the in-progress state, and the Vault Lock policy is removed. You can call `Initiate Vault Lock` again to install a new Vault Lock policy and transition into the in-progress state.

The in-progress state provides the opportunity to test your Vault Lock policy before you lock it. Your Vault Lock policy takes full effect during the in-progress state just as if the vault has been locked, except that you can remove the policy by calling Abort Vault Lock (DELETE lock-policy). To fine-tune your policy, you can repeat the `Abort Vault Lock/Initiate Vault Lock` combination as many times as necessary to validate your Vault Lock policy changes.

After you validate the Vault Lock policy, you can call Complete Vault Lock (POST lockId) with the most recent lock ID to complete the vault locking process. Your vault transitions to a locked state, where the Vault Lock policy is unchangeable and can no longer be removed by calling `Abort Vault Lock`.

Related Sections

- Vault Lock Policies
- Abort Vault Lock (DELETE lock-policy)
- Complete Vault Lock (POST lockId)
- Get Vault Lock (GET lock-policy)
- Initiate Vault Lock (POST lock-policy)

Locking a Vault using the AWS Command Line Interface

You can lock your vault using the AWS Command Line Interface. This will install a vault lock policy on the specified vault and return the lock ID. You must complete the vault locking process within 24 hours else the vault lock policy is removed from the vault.

(Prerequisite) Setting Up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Installing the AWS Command Line Interface

   Configuring the AWS Command Line Interface

2. Verify your AWS CLI setup by entering the following commands at the command prompt. These commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try using the help command.

     ```
     aws help
     ```

   - To get a list of Amazon Glacier vaults on the configured account, use the `list-vaults` command. Replace *123456789012* with your AWS account ID.

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

   - To see the current configuration data for the AWS CLI, use the `aws configure list` command.

     ```
     aws configure list
     ```

1. Use the `initiate-vault-lock` to install a vault lock policy and sets the lock state of the vault lock to `InProgress`.

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

2. The lock configuration is a JSON document as shown in the following example. Before using this command, replace the *VAULT_ARN* and *Principal* with the appropriate values for your use case.

   To find the ARN of the vault you wish to lock, you can use the `list-vaults` command.

```
{"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. After initiating the vault lock you should see the `lockId` returned.

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

To complete the vault lock You must run `complete-vault-lock` within 24 hours else the vault lock policy is removed from the vault.

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

Related Sections

- initiate-vault-lock in the *AWS CLI Command Reference*
- list-vaults in the *AWS CLI Command Reference*
- complete-vault-lock in the *AWS CLI Command Reference*
- Vault Lock Policies
- Abort Vault Lock (DELETE lock-policy)
- Complete Vault Lock (POST lockId)
- Get Vault Lock (GET lock-policy)
- Initiate Vault Lock (POST lock-policy)

Locking a Vault by Using the Amazon Glacier Console

Amazon Glacier Vault Lock helps you to easily deploy and enforce compliance controls for individual Amazon Glacier vaults with a Vault Lock policy. For more information about Amazon Glacier Vault Lock, see Amazon Glacier Access Control with Vault Lock Policies.

Important

- We recommend that you first create a vault, complete a Vault Lock policy, and then upload your archives to the vault so that the policy will be applied to them.
- After the Vault Lock policy is locked, it cannot be changed or deleted.

To initiate a Vault Lock policy on your vault by using the Amazon Glacier console

You initiate the lock by attaching a Vault Lock policy to your vault, which sets the lock to an in-progress state and returns a lock ID. While the policy is in the in-progress state, you have 24 hours to validate your Vault Lock policy before the lock ID expires.

1. Sign in to the AWS Management Console and open the Amazon Glacier console at https://console.aws.amazon.com/glacier/home.
2. Under **Select a Region**, select an AWS Region from the Region selector.
3. In the left navigation pane, choose **Vaults**.
4. On the **Vaults** page, choose **Create vault**.
5. Create a new vault.

   Important

   We recommend that you first create a vault, complete a Vault Lock policy, and then upload your archives to the vault so that the policy will be applied to them.

6. Choose your new vault from the **Vaults** list.
7. Choose the **Vault policies** tab.
8. In the **Vault Lock policy** section, choose **Initiate Vault Lock policy**.
9. On the **Initiate Vault Lock policy** page, specify the record retention controls in your Vault Lock policy in text format in the standard text box.

   Note

   You can specify the record retention controls in a Vault Lock policy in text format and initiate the Vault Lock by calling the `Initiate Vault Lock` API operation

or through the interactive UI in the Amazon Glacier console. For information about formatting your Vault Lock policy, see Amazon Glacier Vault Lock Policy Examples.

10. Choose **Save changes**.
11. In the **Record Vault Lock ID** dialog box, copy your **Lock ID** and save it in a safe place.

> **Important**
>
> After the Vault Lock policy has been initiated, you have 24 hours to validate the policy and complete the lock process. To complete the lock process, you must provide the lock ID. If it's not provided within 24 hours, the lock ID expires and your in-progress policy is deleted.

12. After saving your lock ID in a safe place, choose **Close**.
13. Test your Vault Lock policy within the next 24 hours. If the policy is working as intended, choose **Complete Vault Lock policy**.
14. In the **Complete Vault Lock** dialog box, select the check box to acknowledge that completing the Vault Lock policy process is irreversible.
15. Enter your provided **Lock ID** in the text box.
16. Choose **Complete Vault Lock**.

Working with Archives in Amazon Glacier

An archive is any object, such as a photo, video, or document, that you store in a vault. It is a base unit of storage in Amazon Glacier (Amazon Glacier). Each archive has a unique ID and an optional description. When you upload an archive, Amazon Glacier returns a response that includes an archive ID. This archive ID is unique in the AWS Region in which the archive is stored. The following is an example archive ID.

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

Archive IDs are 138 bytes long. When you upload an archive, you can provide an optional description. You can retrieve an archive using its ID but not its description.

Important

Amazon Glacier provides a management console. You can use the console to create and delete vaults. However, all other interactions with Amazon Glacier require that you use the AWS Command Line Interface (CLI) or write code. For example, to upload data, such as photos, videos, and other documents, you must either use the AWS CLI or write code to make requests, using either the REST API directly or by using the Amazon SDKs. For more information about using Amazon Glacier with the AWS CLI, go to AWS CLI Reference for Amazon Glacier. To install the AWS CLI, go to AWS Command Line Interface.

Topics

Archive Operations in Amazon Glacier

Amazon Glacier supports the following basic archive operations: upload, download, and delete. Downloading an archive is an asynchronous operation.

Uploading an Archive in Amazon Glacier

You can upload an archive in a single operation or upload it in parts. The API call you use to upload an archive in parts is referred as Multipart Upload. For more information, see Uploading an Archive in Amazon Glacier.

> **Important**
>
> Amazon Glacier provides a management console. You can use the console to create and delete vaults. However, all other interactions with Amazon Glacier require that you use the AWS Command Line Interface (CLI) or write code. For example, to upload data, such as photos, videos, and other documents, you must either use the AWS CLI or write code to make requests, using either the REST API directly or by using the Amazon SDKs. For more information about using Amazon Glacier with the AWS CLI, go to AWS CLI Reference for Amazon Glacier. To install the AWS CLI, go to AWS Command Line Interface.

Finding an Archive ID in Amazon Glacier

You can get the archive ID by downloading the vault inventory for the vault that contains the archive. For more information about downloading the vault inventory, see Downloading a Vault Inventory in Amazon Glacier.

Downloading an Archive in Amazon Glacier

Downloading an archive is an asynchronous operation. You must first initiate a job to download a specific archive. After receiving the job request, Amazon Glacier prepares your archive for download. After the job completes, you can download your archive data. Because of the asynchronous nature of the job, you can request Amazon Glacier to send a notification to an Amazon Simple Notification Service (Amazon SNS) topic when the job completes. You can specify an SNS topic for each individual job request or configure your vault to send a notification when

specific events occur. For more information about downloading an archive, see Downloading an Archive in Amazon Glacier.

Deleting an Archive in Amazon Glacier

Amazon Glacier provides an API call that you can use to delete one archive at a time. For more information, see Deleting an Archive in Amazon Glacier.

Updating an Archive in Amazon Glacier

After you upload an archive, you cannot update its content or its description. The only way you can update the archive content or its description is by deleting the archive and uploading another archive. Note that each time you upload an archive, Amazon Glacier returns to you a unique archive ID.

Maintaining Client-Side Archive Metadata

Except for the optional archive description, Amazon Glacier does not support any additional metadata for the archives. When you upload an archive Amazon Glacier assigns an ID, an opaque sequence of characters, from which you cannot infer any meaning about the archive. You might maintain metadata about the archives on the client-side. The metadata can include archive name and some other meaningful information about the archive.

> **Note**
>
> If you are an Amazon Simple Storage Service (Amazon S3) customer, you know that when you upload an object to a bucket, you can assign the object an object key such as `MyDocument.txt` or `SomePhoto.jpg`. In Amazon Glacier, you cannot assign a object key to the archives you upload.

If you maintain client-side archive metadata, note that Amazon Glacier maintains a vault inventory that includes archive IDs and any descriptions you provided during the archive upload. You might occasionally download the vault inventory to reconcile any issues in your client-side database you maintain for the archive metadata. However, Amazon Glacier takes vault inventory approximately daily. When you request a vault inventory, Amazon Glacier returns the last inventory it prepared, a point in time snapshot.

Uploading an Archive in Amazon Glacier

Amazon Glacier (Amazon Glacier) provides a management console, which you can use to create and delete vaults. However, you cannot upload archives to Amazon Glacier by using the management console. To upload data, such as photos, videos, and other documents, you must either use the AWS CLI or write code to make requests, by using either the REST API directly or by using the Amazon SDKs.

For information about using Amazon Glacier with the AWS CLI, go to AWS CLI Reference for Amazon Glacier. To install the AWS CLI, go to AWS Command Line Interface. The following **Uploading** topics describe how to upload archives to Amazon Glacier by using the Amazon SDK for Java, the Amazon SDK for .NET, and the REST API.

Topics

Options for Uploading an Archive to Amazon Glacier

Depending on the size of the data you are uploading, Amazon Glacier offers the following options:

- **Upload archives in a single operation** – In a single operation, you can upload archives from 1 byte to up to 4 GB in size. However, we encourage Amazon Glacier customers to use multipart upload to upload archives greater than 100 MB. For more information, see Uploading an Archive in a Single Operation.
- **Upload archives in parts** – Using the multipart upload API, you can upload large archives, up to about 40,000 GB (10,000 * 4 GB).

  The multipart upload API call is designed to improve the upload experience for larger archives. You can upload archives in parts. These parts can be uploaded independently, in any order, and in parallel. If a part upload fails, you only need to upload that part again and not the entire archive. You can use multipart upload for archives from 1 byte to about 40,000 GB in size. For more information, see Uploading Large Archives in Parts (Multipart Upload).

> **Important**
>
> The Amazon Glacier vault inventory is only updated once a day. When you upload an archive, you will not immediately see the new archive added to your vault (in the console or in your downloaded vault inventory list) until the vault inventory has been updated.

Using the AWS Snowball Edge Service

AWS Snowball Edge accelerates moving large amounts of data into and out of AWS using Amazon-owned devices, bypassing the internet. For more information, see AWS Snowball Edge detail page.

To upload existing data to Amazon Glacier (Amazon Glacier), you might consider using one of the AWS Snowball Edge device types to import data into Amazon S3, and then move it to the Amazon Glacier storage class for archival using lifecycle rules. When you transition Amazon S3 objects to the Amazon Glacier storage class, Amazon S3 internally uses Amazon Glacier for durable storage at lower cost. Although the objects are stored in Amazon Glacier, they remain Amazon S3 objects that you manage in Amazon S3, and you cannot access them directly through Amazon Glacier.

For more information about Amazon S3 lifecycle configuration and transitioning objects to the Amazon Glacier storage class, see Object Lifecycle Management and Transitioning Objects in the *Amazon Simple Storage Service User Guide*.

Uploading an Archive in a Single Operation

As described in Uploading an Archive in Amazon Glacier, you can upload smaller archives in a single operation. However, we encourage Amazon Glacier (Amazon Glacier) customers to use Multipart Upload to upload archives greater than 100 MB.

Topics

- Uploading an Archive in a Single Operation Using the AWS Command Line Interface
- Uploading an Archive in a Single Operation Using the AWS SDK for Java
- Uploading an Archive in a Single Operation Using the AWS SDK for .NET in Amazon Glacier
- Uploading an Archive in a Single Operation Using the REST API

Uploading an Archive in a Single Operation Using the AWS Command Line Interface

You can upload an archive in Amazon Glacier (Amazon Glacier) using the AWS Command Line Interface (AWS CLI).

Topics

(Prerequisite) Setting Up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Installing the AWS Command Line Interface

   Configuring the AWS Command Line Interface

2. Verify your AWS CLI setup by entering the following commands at the command prompt. These commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try using the help command.

     ```
     aws help
     ```

   - To get a list of Amazon Glacier vaults on the configured account, use the `list-vaults` command. Replace *123456789012* with your AWS account ID.

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

   - To see the current configuration data for the AWS CLI, use the `aws configure list` command.

     ```
     aws configure list
     ```

Example: Upload an Archive Using the AWS CLI

In order to upload an archive you must have a vault created. For more information about creating vaults, see Creating a Vault in Amazon Glacier.

1. Use the `upload-archive` command to add an archive to an existing vault. In the below example replace the `vault name` and `account ID`. For the body parameter specify a path to the file you wish to upload.

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

2. Expected output:

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

When finished the command will output the archive ID, checksum, and location in Amazon Glacier. For more information about the upload-archive command, see upload-archive in the *AWS CLI Command Reference*.

Uploading an Archive in a Single Operation Using the AWS SDK for Java

Both the high-level and low-level APIs provided by the Amazon SDK for Java provide a method to upload an archive.

Topics

- Uploading an Archive Using the High-Level API of the AWS SDK for Java
- Uploading an Archive in a Single Operation Using the Low-Level API of the AWS SDK for Java

Uploading an Archive Using the High-Level API of the AWS SDK for Java

The `ArchiveTransferManager` class of the high-level API provides the `upload` method, which you can use to upload an archive to a vault.

> **Note**
>
> You can use the `upload` method to upload small or large archives. Depending on the archive size you are uploading, this method determines whether to upload it in a single operation or use the multipart upload API to upload the archive in parts.

Example: Uploading an Archive Using the High-Level API of the AWS SDK for Java

The following Java code example uploads an archive to a vault (`examplevault`) in the US West (Oregon) Region (`us-west-2`). For a list of supported AWS Regions and endpoints, see Accessing Amazon Glacier.

For step-by-step instructions on how to run this example, see Running Java Examples for Amazon Glacier Using Eclipse. You need to update the code as shown with the name of the vault you want to upload to and the name of the file you want to upload.

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);
        }
    }
}
```

Uploading an Archive in a Single Operation Using the Low-Level API of the AWS SDK for Java

The low-level API provides methods for all the archive operations. The following are the steps to upload an archive using the AWS SDK for Java.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where you want to upload the archive. All operations you perform using this client apply to that AWS Region.
2. Provide request information by creating an instance of the `UploadArchiveRequest` class.

   In addition to the data you want to upload, you need to provide a checksum (SHA-256 tree hash) of the payload, the vault name, the content length of the data, and your account ID.

   If you don't provide an account ID, then the account ID associated with the credentials you provide to sign the request is assumed. For more information, see Using the AWS SDK for Java with Amazon Glacier.
3. Run the `uploadArchive` method by providing the request object as a parameter.

In response, Amazon Glacier (Amazon Glacier) returns an archive ID of the newly uploaded archive.

The following Java code snippet illustrates the preceding steps.

```
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());
```

Example: Uploading an Archive in a Single Operation Using the Low-Level API of the AWS SDK for Java

The following Java code example uses the AWS SDK for Java to upload an archive to a vault (`examplevault`). For step-by-step instructions on how to run this example, see Running Java Examples for Amazon Glacier Using Eclipse. You need to update the code as shown with the name of the vault you want to upload to and the name of the file you want to upload.

```
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);
        }
    }
}
```

Uploading an Archive in a Single Operation Using the AWS SDK for .NET in Amazon Glacier

Both the high-level and low-level APIs provided by the Amazon SDK for .NET provide a method to upload an archive in a single operation.

Topics

- Uploading an Archive Using the High-Level API of the AWS SDK for .NET
- Uploading an Archive in a Single Operation Using the Low-Level API of the AWS SDK for .NET

Uploading an Archive Using the High-Level API of the AWS SDK for .NET

The `ArchiveTransferManager` class of the high-level API provides the `Upload` method that you can use to upload an archive to a vault.

> **Note**
>
> You can use the `Upload` method to upload small or large files. Depending on the file size you are uploading, this method determines whether to upload it in a single operation or use the multipart upload API to upload the file in parts.

Example: Uploading an Archive Using the High-Level API of the AWS SDK for .NET

The following C# code example uploads an archive to a vault (`examplevault`) in the US West (Oregon) Region.

For step-by-step instructions on how to run this example, see Running Code Examples. You need to update the code as shown with the name of the file you want to upload.

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();
    }
  }
}
```

Uploading an Archive in a Single Operation Using the Low-Level API of the AWS SDK for .NET

The low-level API provides methods for all the archive operations. The following are the steps to upload an archive using the AWS SDK for .NET.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where you want to upload the archive. All operations you perform using this client apply to that AWS Region.
2. Provide request information by creating an instance of the `UploadArchiveRequest` class.

   In addition to the data you want to upload, You need to provide a checksum (SHA-256 tree hash) of the payload, the vault name, and your account ID.

   If you don't provide an account ID, then the account ID associated with the credentials you provide to sign the request is assumed. For more information, see Using the AWS SDK for .NET with Amazon Glacier.
3. Run the `UploadArchive` method by providing the request object as a parameter.

   In response, Amazon Glacier returns an archive ID of the newly uploaded archive.

Example: Uploading an Archive in a Single Operation Using the Low-Level API of the AWS SDK for .NET

The following C# code example illustrates the preceding steps. The example uses the AWS SDK for .NET to upload an archive to a vault (`examplevault`).

> **Note**
>
> For information about the underlying REST API to upload an archive in a single request, see Upload Archive (POST archive).

For step-by-step instructions on how to run this example, see Running Code Examples. You need to update the code as shown with the name of the file you want to upload.

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;
      }
    }
  }
}
```

Uploading an Archive in a Single Operation Using the REST API

You can use the Upload Archive API call to upload an archive in a single operation. For more information, see Upload Archive (POST archive).

Uploading Large Archives in Parts (Multipart Upload)

Topics

- Multipart Upload Process
- Quick Facts
- Uploading Large Archives by Using the AWS CLI
- Uploading Large Archives in Parts Using the Amazon SDK for Java
- Uploading Large Archives Using the AWS SDK for .NET

- Uploading Large Archives in Parts Using the REST API

Multipart Upload Process

As described in Uploading an Archive in Amazon Glacier, we encourage Amazon Glacier (Amazon Glacier) customers to use Multipart Upload to upload archives greater than 100 mebibytes (MiB).

1. **Initiate Multipart Upload**

   When you send a request to initiate a multipart upload, Amazon Glacier returns a multipart upload ID, which is a unique identifier for your multipart upload. Any subsequent multipart upload operations require this ID. This ID doesn't expire for at least 24 hours after Amazon Glacier completes the job.

   In your request to start a multipart upload, specify the part size in number of bytes. Each part you upload, except the last part, must be this size.

   > **Note**
   >
   > You don't need to know the overall archive size when using multipart uploads. This means that you can use multipart uploads in cases where you don't know the archive size when you start uploading the archive. You only need to decide the part size at the time you start the multipart upload.

   In the initiate multipart upload request, you can also provide an optional archive description.

2. **Upload Parts**

   For each part upload request, you must include the multipart upload ID you obtained in step 1. In the request, you must also specify the content range, in bytes, identifying the position of the part in the final archive. Amazon Glacier later uses the content range information to assemble the archive in proper sequence. Because you provide the content range for each part that you upload, it determines the part's position in the final assembly of the archive, and therefore you can upload parts in any order. You can also upload parts in parallel. If you upload a new part using the same content range as a previously uploaded part, the previously uploaded part is overwritten.

3. **Complete (or stop) Multipart Upload**

After uploading all the archive parts, you use the complete operation. Again, you must specify the upload ID in your request. Amazon Glacier creates an archive by concatenating parts in ascending order based on the content range you provided. Amazon Glacier response to a Complete Multipart Upload request includes an archive ID for the newly created archive. If you provided an optional archive description in the Initiate Multipart Upload request, Amazon Glacier associates it with the assembled archive. After you successfully complete a multipart upload, you cannot refer to the multipart upload ID. That means you cannot access parts associated with the multipart upload ID.

If you stop a multipart upload, you cannot upload any more parts using that multipart upload ID. All storage consumed by any parts associated with the stopped multipart upload is freed. If any part uploads were in-progress, they can still succeed or fail even after stopped.

Additional Multipart Upload Operations

Amazon Glacier (Amazon Glacier) provides the following additional multipart upload API calls.

- **List Parts**—Using this operation, you can list the parts of a specific multipart upload. It returns information about the parts that you have uploaded for a multipart upload. For each list parts request, Amazon Glacier returns information for up to 1,000 parts. If there are more parts to list for the multipart upload, the result is paginated and a marker is returned in the response at which to continue the list. You need to send additional requests to retrieve subsequent parts. Note that the returned list of parts doesn't include parts that haven't completed uploading.
- **List Multipart Uploads**—Using this operation, you can obtain a list of multipart uploads in progress. An in-progress multipart upload is an upload that you have initiated, but have not yet completed or stopped. For each list multipart uploads request, Amazon Glacier returns up to 1,000 multipart uploads. If there are more multipart uploads to list, then the result is paginated and a marker is returned in the response at which to continue the list. You need to send additional requests to retrieve the remaining multipart uploads.

Quick Facts

The following table provides multipart upload core specifications.

Item	Specification
Maximum archive size	10,000 x 4 gibibytes (GiB)
Maximum number of parts per upload	10,000
Part size	1 MiB to 4 GiB, last part can be < 1 MiB. You specify the size value in bytes. The part size must be a mebibyte (1024 kibibytes [KiB]) multiplied by a power of 2. For example, `1048576` (1 MiB), `2097152` (2 MiB), `4194304` (4 MiB), `8388608` (8 MiB).
Maximum number of parts returned for a list parts request	1,000
Maximum number of multipart uploads returned in a list multipart uploads request	1,000

Uploading Large Archives by Using the AWS CLI

You can upload an archive in Amazon Glacier (Amazon Glacier) by using the AWS Command Line Interface (AWS CLI). To improve the upload experience for larger archives, Amazon Glacier provides several API operations to support multipart uploads. By using these API operations, you can upload archives in parts. These parts can be uploaded independently, in any order, and in parallel. If a part upload fails, you need to upload only that part again, not the entire archive. You can use multipart uploads for archives from 1 byte to about 40,000 gibibytes (GiB) in size.

For more information about Amazon Glacier multipart uploads, see Uploading Large Archives in Parts (Multipart Upload).

Topics

- (Prerequisite) Setting Up the AWS CLI
- (Prerequisite) Install Python
- (Prerequisite) Create an Amazon Glacier Vault

- Example: Uploading Large Archives in Parts by Using the AWS CLI

(Prerequisite) Setting Up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Installing the AWS Command Line Interface

   Configuring the AWS Command Line Interface

2. Verify your AWS CLI setup by entering the following commands at the command prompt. These commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try using the help command.

     ```
     aws help
     ```

   - To get a list of Amazon Glacier vaults on the configured account, use the `list-vaults` command. Replace *123456789012* with your AWS account ID.

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

   - To see the current configuration data for the AWS CLI, use the `aws configure list` command.

     ```
     aws configure list
     ```

(Prerequisite) Install Python

To complete a multipart upload, you must calculate the SHA256 tree hash of the archive that you're uploading. Doing so is different than calculating the SHA256 tree hash of the file that you want to upload. To calculate the SHA256 tree hash of the archive that you're uploading, you can use Java, C# (with .NET), or Python. In this example, you will use Python. For instructions on using Java or C#, see Computing Checksums.

For more information about installing Python, see Install or update Python in the *Boto3 Developer Guide*.

(Prerequisite) Create an Amazon Glacier Vault

To use the following example, you must have at least one Amazon Glacier vault created. For more information about creating vaults, see Creating a Vault in Amazon Glacier.

Example: Uploading Large Archives in Parts by Using the AWS CLI

In this example, you will create a file and use multipart upload API operations to upload this file, in parts, to Amazon Glacier.

> **Important**
>
> Before starting this procedure, make sure that you've performed all of the prerequisite steps. To upload an archive, you must have a vault created, the AWS CLI configured, and be prepared to use Java, C#, or Python to calculate a SHA256 tree hash.

The following procedure uses the `initiate-multipart-upload`, `upload-multipart-part`, and `complete-multipart-upload` AWS CLI commands.

For more detailed information about each of these commands, see initiate-multipart-upload, upload-multipart-part, and complete-multipart-upload in the *AWS CLI Command Reference*.

1. Use the initiate-multipart-upload command to create a multipart upload resource. In your request, specify the part size in number of bytes. Each part that you upload, except the last part, will be this size. You don't need to know the overall archive size when initiating an upload. However, you will need the total size, in bytes, of each part when completing the upload on the final step.

   In the following command, replace the values for the `--vault-name` and `--account-ID` parameters with your own information. This command specifies that you will upload an archive with a part size of 1 mebibyte (MiB) (1024 x 1024 bytes) per file. Replace this `--part-size` parameter value if needed.

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

   Expected output:

   ```
   {
   ```

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

When finished, the command will output the multipart upload resource's upload ID and location in Amazon Glacier. You will use this upload ID in subsequent steps.

2. For this example, you can use the following commands to create a 4.4 MiB file, split it into 1 MiB chunks, and upload each chunk. To upload your own files, you can follow a similar procedure of splitting your data into chunks and uploading each part.

   Linux or macOS

   The following command creates a 4.4 MiB file, named `file_to_upload`, on Linux or macOS.

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

   Windows

   The following command creates a 4.4 MiB file, named `file_to_upload`, on Windows.

   ```
   fsutil file createnew file_to_upload 4608000
   ```

3. Next, you will split this file into 1 MiB chunks.

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

   You now have the following five chunks. The first four are 1 MiB, and the last is approximately 400 kibibytes (KiB).

   ```
   chunkaa
   chunkab
   chunkac
   chunkad
   chunkae
   ```

4. Use the upload-multipart-part command to upload a part of an archive. You can upload archive parts in any order. You can also upload them in parallel. You can upload up to 10,000 parts for a multipart upload.

In the following command, replace the values for the `--vault-name`, `--account-ID`, and `--upload-id` parameters. The upload ID must match the ID given as output of the `initiate-multipart-upload` command. The `--range` parameter specifies that you will upload a part with a size of 1 MiB (1024 x 1024 bytes). This size must match what you specified in the `initiate-multipart-upload` command. Adjust this size value if needed. The `--body` parameter specifies the name of the part that you're uploading.

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

If successful, the command will produce output that contains the checksum for the uploaded part.

5. Run the `upload-multipart-part` command again to upload the remaining parts of your multipart upload. Update the `--range` and `--body` parameter values for each command to match the part that you're uploading.

```
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**
>
> The final command's `--range` parameter value is smaller because the final part of our upload is less than 1 MiB. If successful, each command will produce output that contains the checksum for each uploaded part.

6. Next, you will assemble the archive and finish the upload. You must include the total size and SHA256 tree hash of the archive.

To calculate the SHA256 tree hash of the archive, you can use Java, C#, or Python. In this example, you will use Python. For instructions on using Java or C#, see Computing Checksums.

Create the Python file `checksum.py` and insert the following code. If needed, replace the name of the original file.

```
from botocore.utils import calculate_tree_hash

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

7. Run `checksum.py` to calculate the SHA256 tree hash. The following hash may not match your output.

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

8. Use the complete-multipart-upload command to finish the archive upload. Replace the values for the `--vault-name`, `--account-ID`, `--upload-ID`, and `--checksum` parameters. The `--archive` parameter value specifies the total size, in bytes, of the archive. This value must be the sum of all the sizes of the individual parts that you uploaded. Replace this value if needed.

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

When finished, the command will output the archive's ID, checksum, and location in Amazon Glacier.

Uploading Large Archives in Parts Using the Amazon SDK for Java

Both the high-level and low-level APIs provided by the Amazon SDK for Java provide a method to upload a large archive (see Uploading an Archive in Amazon Glacier).

- The high-level API provides a method that you can use to upload archives of any size. Depending on the file you are uploading, the method either uploads an archive in a single operation or uses the multipart upload support in Amazon Glacier (Amazon Glacier) to upload the archive in parts.

- The low-level API maps close to the underlying REST implementation. Accordingly, it provides a method to upload smaller archives in one operation and a group of methods that support multipart upload for larger archives. This section explains uploading large archives in parts using the low-level API.

For more information about the high-level and low-level APIs, see Using the AWS SDK for Java with Amazon Glacier.

Topics

- Uploading Large Archives in Parts Using the High-Level API of the AWS SDK for Java
- Upload Large Archives in Parts Using the Low-Level API of the AWS SDK for Java

Uploading Large Archives in Parts Using the High-Level API of the AWS SDK for Java

You use the same methods of the high-level API to upload small or large archives. Based on the archive size, the high-level API methods decide whether to upload the archive in a single operation or use the multipart upload API provided by Amazon Glacier. For more information, see Uploading an Archive Using the High-Level API of the AWS SDK for Java.

Upload Large Archives in Parts Using the Low-Level API of the AWS SDK for Java

For granular control of the upload you can use the low-level API where you can configure the request and process the response. The following are the steps to upload large archives in parts using the AWS SDK for Java.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where you want to save the archive. All operations you perform using this client apply to that AWS Region.
2. Initiate multipart upload by calling the `initiateMultipartUpload` method.

   You need to provide vault name in which you want to upload the archive, part size you want to use to upload archive parts, and an optional description. You provide this information by creating an instance of the `InitiateMultipartUploadRequest` class. In response, Amazon Glacier returns an upload ID.
3. Upload parts by calling the `uploadMultipartPart` method.

For each part you upload, You need to provide the vault name, the byte range in the final assembled archive that will be uploaded in this part, the checksum of the part data, and the upload ID.

4. Complete multipart upload by calling the `completeMultipartUpload` method.

   You need to provide the upload ID, the checksum of the entire archive, the archive size (combined size of all parts you uploaded), and the vault name. Amazon Glacier constructs the archive from the uploaded parts and returns an archive ID.

Example: Uploading a Large Archive in a Parts Using the AWS SDK for Java

The following Java code example uses the AWS SDK for Java to upload an archive to a vault (`examplevault`). For step-by-step instructions on how to run this example, see Running Java Examples for Amazon Glacier Using Eclipse. You need to update the code as shown with the name of the file you want to upload.

> **Note**
>
> This example is valid for part sizes from 1 MB to 1 GB. However, Amazon Glacier supports part sizes up to 4 GB.

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();
    }
}
```

Uploading Large Archives Using the AWS SDK for .NET

Both the high-level and low-level APIs provided by the Amazon SDK for .NET provide a method to upload large archives in parts (see Uploading an Archive in Amazon Glacier).

- The high-level API provides a method that you can use to upload archives of any size. Depending on the file you are uploading, the method either uploads archive in a single operation or uses the multipart upload support in Amazon Glacier (Amazon Glacier) to upload the archive in parts.
- The low-level API maps close to the underlying REST implementation. Accordingly, it provides a method to upload smaller archives in one operation and a group of methods that support multipart upload for larger archives. This section explains uploading large archives in parts using the low-level API.

For more information about the high-level and low-level APIs, see Using the AWS SDK for .NET with Amazon Glacier.

Topics

Uploading Large Archives in Parts Using the High-Level API of the AWS SDK for .NET

You use the same methods of the high-level API to upload small or large archives. Based on the archive size, the high-level API methods decide whether to upload the archive in a single operation or use the multipart upload API provided by Amazon Glacier. For more information, see Uploading an Archive Using the High-Level API of the AWS SDK for .NET.

Uploading Large Archives in Parts Using the Low-Level API of the AWS SDK for .NET

For granular control of the upload, you can use the low-level API, where you can configure the request and process the response. The following are the steps to upload large archives in parts using the AWS SDK for .NET.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where you want to save the archive. All operations you perform using this client apply to that AWS Region.
2. Initiate multipart upload by calling the `InitiateMultipartUpload` method.

   You need to provide the vault name to which you want to upload the archive, the part size you want to use to upload archive parts, and an optional description. You provide this information by creating an instance of the `InitiateMultipartUploadRequest` class. In response, Amazon Glacier returns an upload ID.
3. Upload parts by calling the `UploadMultipartPart` method.

   For each part you upload, You need to provide the vault name, the byte range in the final assembled archive that will be uploaded in this part, the checksum of the part data, and the upload ID.
4. Complete the multipart upload by calling the `CompleteMultipartUpload` method.

   You need to provide the upload ID, the checksum of the entire archive, the archive size (combined size of all parts you uploaded), and the vault name. Amazon Glacier constructs the archive from the uploaded parts and returns an archive ID.

Example: Uploading a Large Archive in Parts Using the Amazon SDK for .NET

The following C# code example uses the AWS SDK for .NET to upload an archive to a vault (`examplevault`). For step-by-step instructions on how to run this example, see Running Code Examples. You need to update the code as shown with the name of a file you want to upload.

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;
    }
  }
}
```

Uploading Large Archives in Parts Using the REST API

As described in Uploading Large Archives in Parts (Multipart Upload), multipart upload refers to a set of operations that enable you to upload an archive in parts and perform related operations. For more information about these operations, see the following API reference topics:

- Initiate Multipart Upload (POST multipart-uploads)
- Upload Part (PUT uploadID)

- Complete Multipart Upload (POST uploadID)
- Abort Multipart Upload (DELETE uploadID)
- List Parts (GET uploadID)
- List Multipart Uploads (GET multipart-uploads)

Downloading an Archive in Amazon Glacier

Amazon Glacier provides a management console, which you can use to create and delete vaults. However, you cannot download archives from Amazon Glacier by using the management console. To download data, such as photos, videos, and other documents, you must either use the AWS Command Line Interface (AWS CLI) or write code to make requests, by using either the REST API directly or by using the AWS SDKs.

For information about using Amazon Glacier with the AWS CLI, see the AWS CLI Reference for Amazon Glacier. To install the AWS CLI, see AWS Command Line Interface. The following topics describe how to download archives to Amazon Glacier by using the AWS SDK for Java, the AWS SDK for .NET, and the Amazon Glacier REST API.

Topics

- Retrieving Amazon Glacier Archives
- Downloading an Archive in Amazon Glacier Using the AWS SDK for Java
- Downloading an Archive in Amazon Glacier Using the AWS SDK for .NET
- Downloading a Large Archive Using Parallel Processing with Python
- Downloading an Archive by Using the REST API
- Downloading an Archive in Amazon Glacier Using the AWS CLI

Retrieving Amazon Glacier Archives

Retrieving an archive from Amazon Glacier is an asynchronous operation in which you first initiate a job, and then download the output after the job is completed. To initiate an archive retrieval job, you use the Initiate Job (POST jobs) REST API operation or the equivalent in the AWS CLI, or the AWS SDKs.

Topics

- Archive Retrieval Options

- Ranged Archive Retrievals

Retrieving an archive from Amazon Glacier is a two-step process. The following is an overview of this process.

To retrieve an archive

1. Initiate an archive retrieval job.

   a. Get the ID of the archive that you want to retrieve. You can get the archive ID from an inventory of the vault. You can get the archive ID with the REST API, AWS CLI, or AWS SDKs. For more information, see Downloading a Vault Inventory in Amazon Glacier.

   b. Initiate a job that requests Amazon Glacier to prepare an entire archive or a portion of the archive for subsequent download by using the Initiate Job (POST jobs) operation.

   When you initiate a job, Amazon Glacier returns a job ID in the response and runs the job asynchronously. (You cannot download the job output until after the job is completed, as described in Step 2.)

   > **Important**
   >
   > For Standard retrievals only, a data retrieval policy can cause your `Initiate Job` request to fail with a `PolicyEnforcedException` exception. For more information about data retrieval policies, see Amazon Glacier Data Retrieval Policies. For more information about the `PolicyEnforcedException` exception, see Error Responses.

   When required, you can restore large segments of the data stored in Amazon Glacier. For more information about restoring data from the Amazon Glacier storage classes, see Storage Classes for Archiving Objects in the *Amazon Simple Storage Service User Guide*.

2. After the job is completed, download the bytes by using the Get Job Output (GET output) operation.

   You can download all bytes or specify a byte range to download only a portion of the job output. For larger output, downloading the output in chunks helps in the event of a download failure, such as a network failure. If you get job output in a single request and there is a network failure, you have to restart downloading the output from the beginning. However,

if you download the output in chunks, in the event of any failure, you need only restart the download of the smaller portion and not the entire output.

Amazon Glacier must complete a job before you can get its output. After completion, a job does not expire for at least 24 hours, which means that you can download the output within the 24-hour period after the job is completed. The restore can expire any time after 24 hours from the completion of the job. To determine if your job is complete, check its status by using one of the following options:

- **Wait for a job-completion notification** – You can specify an Amazon Simple Notification Service (Amazon SNS) topic to which Amazon Glacier can post a notification after the job is completed. Amazon Glacier sends a notification only after it completes the job.

  You can specify an Amazon SNS topic for a job when you initiate the job. In addition to specifying an Amazon SNS topic in your job request, if your vault has notifications set for archive retrieval events, then Amazon Glacier also publishes a notification to that SNS topic. For more information, see Configuring Vault Notifications in Amazon Glacier.
- **Request job information explicitly** – You can also use the Amazon Glacier `Describe Job` API operation (Describe Job (GET JobID)) to periodically poll for job information. However, we recommend using Amazon SNS notifications.

> **Note**
>
> The information that you get by using an Amazon SNS notification is the same as what you get by calling the `Describe Job` API operation.

Archive Retrieval Options

When initiating a job to retrieve an archive, you can specify one of the following retrieval options, based on your access time and cost requirements. For information about retrieval pricing, see Amazon Glacier Pricing.

- **Expedited** – Expedited retrievals allow you to quickly access your data that's stored in the S3 Glacier Flexible Retrieval storage class or the S3 Intelligent-Tiering Archive Access tier when occasional urgent requests for restoring archives are required. For all but the largest archives (more than 250 MB), data accessed by using Expedited retrievals is typically made available

within 1–5 minutes. Provisioned capacity ensures that retrieval capacity for Expedited retrievals is available when you need it. For more information, see Provisioned Capacity.

- **Standard** – Standard retrievals allow you to access any of your archives within several hours. Standard retrievals are typically completed within 3–5 hours. Standard is the default option for retrieval requests that do not specify the retrieval option.
- **Bulk** – Bulk retrievals are the lowest-cost Amazon Glacier retrieval option, which you can use to retrieve large amounts, even petabytes, of data inexpensively in a day. Bulk retrievals are typically completed within 5–12 hours.

The following table summarizes the archive retrieval options. For information about pricing, see Amazon Glacier pricing.

To make an `Expedited`, `Standard`, or `Bulk` retrieval, set the `Tier` request element in the RestoreObject REST API operation request to the option that you want, or the equivalent in the AWS Command Line Interface (AWS CLI) or AWS SDKs. If you purchased provisioned capacity, all Expedited retrievals are automatically served through your provisioned capacity.

Provisioned Capacity

Provisioned capacity helps ensure that your retrieval capacity for Expedited retrievals is available when you need it. Each unit of capacity provides that at least three Expedited retrievals can be performed every 5 minutes and provides up to 150 megabytes per second (MBps) of retrieval throughput.

If your workload requires highly reliable and predictable access to a subset of your data in minutes, we recommend that you purchase provisioned retrieval capacity. Without provisioned capacity, Expedited retrievals are typically accepted, except for rare situations of unusually high demand. However, if you require access to Expedited retrievals under all circumstances, you must purchase provisioned retrieval capacity.

Purchasing Provisioned Capacity

You can purchase provisioned capacity units by using the Amazon Glacier console, the Purchase Provisioned Capacity (POST provisioned-capacity) REST API operation, the AWS SDKs, or the AWS CLI. For provisioned capacity pricing information, see Amazon Glacier Pricing.

A provisioned capacity unit lasts for one month, starting at the date and time of purchase.

If the start date is on the 31st day of a month, the expiration date is the last day of the next month. For example, if the start date is August 31, the expiration date is September 30. If the start date is January 31, the expiration date is February 28.

To purchase provisioned capacity by using the Amazon Glacier console

1. Sign in to the AWS Management Console and open the Amazon Glacier console at https://console.aws.amazon.com/glacier/home.
2. In the left navigation pane, choose **Data retrieval settings**.
3. Under **Provisioned capacity units (PCUs)**, choose **Purchase PCU**. The **Purchase PCU** dialog box appears.
4. If you want to purchase provisioned capacity, enter **`confirm`** in the **To confirm purchase** box.
5. Choose **Purchase PCU**.

Ranged Archive Retrievals

When you retrieve an archive from Amazon Glacier, you can optionally specify a range, or portion, of the archive to retrieve. The default is to retrieve the whole archive. Specifying a range of bytes can be helpful when you want to do the following:

- **Manage your data downloads** – Amazon Glacier allows retrieved data to be downloaded for 24 hours after the retrieval request is completed. Therefore, you might want to retrieve only portions of the archive so that you can manage the schedule of downloads within the given download window.
- **Retrieve a targeted part of a large archive** – For example, suppose you have previously aggregated many files and uploaded them as a single archive, and now you want to retrieve a few of the files. In this case, you can specify a range of the archive that contains the files that you are interested in by using one retrieval request. Or, you can initiate multiple retrieval requests, each with a range for one or more files.

When initiating a retrieval job using range retrievals, you must provide a range that is megabyte aligned. In other words, the byte range can start at zero (the beginning of your archive), or at any 1-MB interval thereafter (1 MB, 2 MB, 3 MB, and so on).

The end of the range can either be the end of your archive or any 1 MB interval greater than the beginning of your range. Furthermore, if you want to get checksum values when you download the data (after the retrieval job is completed), the range that you request in the job initiation must

also be tree-hash aligned. You can use checksums to help ensure that your data was not corrupted during transmission. For more information about megabyte alignment and tree-hash alignment, see Receiving Checksums When Downloading Data.

Downloading an Archive in Amazon Glacier Using the AWS SDK for Java

Both the high-level and low-level APIs provided by the Amazon SDK for Java provide a method to download an archive.

Topics

- Downloading an Archive Using the High-Level API of the AWS SDK for Java
- Downloading an Archive Using the Low-Level API of the AWS SDK for Java

Downloading an Archive Using the High-Level API of the AWS SDK for Java

The `ArchiveTransferManager` class of the high-level API provides the `download` method you can use to download an archive.

> **Important**
>
> The `ArchiveTransferManager` class creates an Amazon Simple Notification Service (Amazon SNS) topic, and an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to that topic. It then initiates the archive retrieval job and polls the queue for the archive to be available. When the archive is available, download begins. For information about retrieval times, see Archive Retrieval Options.

Example: Downloading an Archive Using the High-Level API of the AWS SDK for Java

The following Java code example downloads an archive from a vault (`examplevault`) in the US West (Oregon) Region (`us-west-2`).

For step-by-step instructions to run this sample, see Running Java Examples for Amazon Glacier Using Eclipse. You need to update the code as shown with an existing archive ID and the local file path where you want to save the downloaded archive.

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);
        }
    }
```

```
}
```

Downloading an Archive Using the Low-Level API of the AWS SDK for Java

The following are the steps to retrieve a vault inventory using the AWS SDK for Java low-level API.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region from where you want to download the archive. All operations you perform using this client apply to that AWS Region.
2. Initiate an `archive-retrieval` job by executing the `initiateJob` method.

   You provide job information, such as the archive ID of the archive you want to download and the optional Amazon SNS topic to which you want Amazon Glacier (Amazon Glacier) to post a job completion message, by creating an instance of the `InitiateJobRequest` class. Amazon Glacier returns a job ID in response. The response is available in an instance of the `InitiateJobResult` class.

```
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();
```

   You can optionally specify a byte range to request Amazon Glacier to prepare only a portion of the archive. For example, you can update the preceding request by adding the following statement to request Amazon Glacier to prepare only the 1 MB to 2 MB portion of the archive.

```
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. Wait for the job to complete.

   You must wait until the job output is ready for you to download. If you have either set a notification configuration on the vault identifying an Amazon Simple Notification Service (Amazon SNS) topic or specified an Amazon SNS topic when you initiated a job, Amazon Glacier sends a message to that topic after it completes the job.

   You can also poll Amazon Glacier by calling the `describeJob` method to determine the job completion status. Although, using an Amazon SNS topic for notification is the recommended approach.

4. Download the job output (archive data) by executing the `getJobOutput` method.

   You provide the request information such as the job ID and vault name by creating an instance of the `GetJobOutputRequest` class. The output that Amazon Glacier returns is available in the `GetJobOutputResult` object.

```
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.
```

   The preceding code snippet downloads the entire job output. You can optionally retrieve only a portion of the output, or download the entire output in smaller chunks by specifying the byte range in your `GetJobOutputRequest`.

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

In response to your `GetJobOutput` call, Amazon Glacier returns the checksum of the portion of the data you downloaded, if certain conditions are met. For more information, see Receiving Checksums When Downloading Data.

To verify there are no errors in the download, you can then compute the checksum on the client-side and compare it with the checksum Amazon Glacier sent in the response.

For an archive retrieval job with the optional range specified, when you get the job description, it includes the checksum of the range you are retrieving (SHA256TreeHash). You can use this value to further verify the accuracy of the entire byte range that you later download. For example, if you initiate a job to retrieve a tree-hash aligned archive range and then download output in chunks such that each of your `GetJobOutput` requests return a checksum, then you can compute checksum of each portion you download on the client-side and then compute the tree hash. You can compare it with the checksum Amazon Glacier returns in response to your describe job request to verify that the entire byte range you have downloaded is the same as the byte range that is stored in Amazon Glacier.

For a working example, see Example 2: Retrieving an Archive Using the Low-Level API of the AWS SDK for Java—Download Output in Chunks.

Example 1: Retrieving an Archive Using the Low-Level API of the AWS SDK for Java

The following Java code example downloads an archive from the specified vault. After the job completes, the example downloads the entire output in a single `getJobOutput` call. For an example of downloading output in chunks, see Example 2: Retrieving an Archive Using the Low-Level API of the AWS SDK for Java—Download Output in Chunks.

The example performs the following tasks:

- Creates an Amazon Simple Notification Service (Amazon SNS) topic.

  Amazon Glacier sends a notification to this topic after it completes the job.

- Creates an Amazon Simple Queue Service (Amazon SQS) queue.

  The example attaches a policy to the queue to enable the Amazon SNS topic to post messages to the queue.
- Initiates a job to download the specified archive.

  In the job request, the Amazon SNS topic that was created is specified so that Amazon Glacier can publish a notification to the topic after it completes the job.
- Periodically checks the Amazon SQS queue for a message that contains the job ID.

  If there is a message, parse the JSON and check if the job completed successfully. If it did, download the archive.
- Cleans up by deleting the Amazon SNS topic and the Amazon SQS queue that it created.

```
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));
    }
}
```

Example 2: Retrieving an Archive Using the Low-Level API of the AWS SDK for Java—Download Output in Chunks

The following Java code example retrieves an archive from Amazon Glacier. The code example downloads the job output in chunks by specifying byte range in a `GetJobOutputRequest` object.

```
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));
    }
}
```

Downloading an Archive in Amazon Glacier Using the AWS SDK for .NET

Both the high-level and low-level APIs provided by the Amazon SDK for .NET provide a method to download an archive.

Topics

Downloading an Archive Using the High-Level API of the AWS SDK for .NET

The `ArchiveTransferManager` class of the high-level API provides the `Download` method you can use to download an archive.

> **Important**
>
> The `ArchiveTransferManager` class creates an Amazon Simple Notification Service (Amazon SNS) topic, and an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to that topic. It then initiates the archive retrieval job and polls the queue for the archive to be available. When the archive is available, download begins. For information about retrieval times, see Archive Retrieval Options

Example: Downloading an Archive Using the High-Level API of the AWS SDK for .NET

The following C# code example downloads an archive from a vault (`examplevault`) in the US West (Oregon) Region.

For step-by-step instructions on how to run this example, see Running Code Examples. You need to update the code as shown with an existing archive ID and the local file path where you want to save the downloaded archive.

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

Downloading an Archive Using the Low-Level API of the AWS SDK for .NET

The following are the steps for downloading an Amazon Glacier (Amazon Glacier) archive using the low-level API of the AWS SDK for .NET.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region from where you want to download the archive. All operations you perform using this client apply to that AWS Region.
2. Initiate an `archive-retrieval` job by executing the `InitiateJob` method.

   You provide job information, such as the archive ID of the archive you want to download and the optional Amazon SNS topic to which you want Amazon Glacier to post a job completion message, by creating an instance of the `InitiateJobRequest` class. Amazon Glacier returns a job ID in response. The response is available in an instance of the `InitiateJobResponse` class.

```
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;
```

   You can optionally specify a byte range to request Amazon Glacier to prepare only a portion of the archive as shown in the following request. The request specifies Amazon Glacier to prepare only the 1 MB to 2 MB portion of the archive.

```
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. Wait for the job to complete.

   You must wait until the job output is ready for you to download. If you have either set a notification configuration on the vault identifying an Amazon Simple Notification Service (Amazon SNS) topic or specified an Amazon SNS topic when you initiated a job, Amazon Glacier sends a message to that topic after it completes the job. The code example given in the following section uses Amazon SNS for Amazon Glacier to publish a message.

   You can also poll Amazon Glacier by calling the `DescribeJob` method to determine the job completion status. Although, using an Amazon SNS topic for notification is the recommended approach .

4. Download the job output (archive data) by executing the `GetJobOutput` method.

   You provide the request information such as the job ID and vault name by creating an instance of the `GetJobOutputRequest` class. The output that Amazon Glacier returns is available in the `GetJobOutputResponse` object.

```
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);
  }
}
```

The preceding code snippet downloads the entire job output. You can optionally retrieve only a portion of the output, or download the entire output in smaller chunks by specifying the byte range in your `GetJobOutputRequest`.

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

In response to your `GetJobOutput` call, Amazon Glacier returns the checksum of the portion of the data you downloaded, if certain conditions are met. For more information, see Receiving Checksums When Downloading Data.

To verify there are no errors in the download, you can then compute the checksum on the client-side and compare it with the checksum Amazon Glacier sent in the response.

For an archive retrieval job with the optional range specified, when you get the job description, it includes the checksum of the range you are retrieving (SHA256TreeHash).You can use this value to further verify the accuracy of the entire byte range that you later download. For example, if you initiate a job to retrieve a tree-hash aligned archive range and then download output in chunks such that each of your `GetJobOutput` requests return a checksum, then you can compute checksum of each portion you download on the client-side and then compute the tree hash. You can compare it with the checksum Amazon Glacier returns in response to your describe job request to verify that the entire byte range you have downloaded is the same as the byte range that is stored in Amazon Glacier.

For a working example, see Example 2: Retrieving an Archive Using the Low-Level API of the AWS SDK for .NET—Download Output in Chunks.

Example 1: Retrieving an Archive Using the Low-Level API of the AWS SDK for .NET

The following C# code example downloads an archive from the specified vault. After the job completes, the example downloads the entire output in a single `GetJobOutput` call. For an example of downloading output in chunks, see Example 2: Retrieving an Archive Using the Low-Level API of the AWS SDK for .NET—Download Output in Chunks.

The example performs the following tasks:

- Sets up an Amazon Simple Notification Service (Amazon SNS) topic

  Amazon Glacier sends a notification to this topic after it completes the job.
- Sets up an Amazon Simple Queue Service (Amazon SQS) queue.

  The example attaches a policy to the queue to enable the Amazon SNS topic to post messages.
- Initiates a job to download the specified archive.

  In the job request, the example specifies the Amazon SNS topic so that Amazon Glacier can send a message after it completes the job.
- Periodically checks the Amazon SQS queue for a message.

  If there is a message, parse the JSON and check if the job completed successfully. If it did, download the archive. The code example uses the JSON.NET library (see JSON.NET) to parse the JSON.
- Cleans up by deleting the Amazon SNS topic and the Amazon SQS queue it created.

```
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\",&TCX5-2025-waiver;" +
        "    \"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);
      }
    }
  }
}
```

Example 2: Retrieving an Archive Using the Low-Level API of the AWS SDK for .NET—Download Output in Chunks

The following C# code example retrieves an archive from Amazon Glacier. The code example downloads the job output in chunks by specifying the byte range in a `GetJobOutputRequest` object.

```
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\",&TCX5-2025-waiver;" +
        "    \"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);
     }
   }
 }
```

```
}
```

Downloading a Large Archive Using Parallel Processing with Python

This topic describes how to download a large archive from Amazon S3 Glacier (S3 Glacier) using parallel processing with Python. This approach allows you to reliably download archives of any size by breaking them into smaller pieces that can be processed independently.

Overview

The Python script provided in this example performs the following tasks:

1. Sets up the necessary AWS resources (Amazon SNS topic and Amazon SQS queues) for notifications
2. Initiates an archive retrieval job with Amazon Glacier
3. Monitors an Amazon SQS queue for job completion notifications
4. Splits the large archive into manageable chunks
5. Downloads chunks in parallel using multiple worker threads
6. Saves each chunk to disk for later reassembly

Prerequisites

Before you begin, make sure you have:

- Python 3.6 or later installed
- AWS SDK for Python (Boto3) installed
- AWS credentials configured with appropriate permissions for Amazon Glacier, Amazon SNS, and Amazon SQS
- Sufficient disk space to store the downloaded archive chunks

Example: Downloading an Archive Using Parallel Processing with Python

The following Python script demonstrates how to download a large archive from Amazon Glacier using parallel processing:

```
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)
```

Using the Script

To use this script, follow these steps:

1. Replace the placeholder values in the script with your specific information:
   - *output_file_path*: Directory where chunk files will be saved
   - *vault_name*: Name of your S3 Glacier vault
   - *notify_queue_name*: Name for the job notification queue
   - *chunk_download_queue_name*: Name for the chunk download queue
   - *sns_topic_name*: Name for the SNS topic
   - *region*: AWS region where your vault is located
   - *archive_id*: ID of the archive to retrieve
2. Run the script:

   ```
   python download_large_archive.py
   ```
3. After all chunks are downloaded, you can combine them into a single file using a command like:

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

Important Considerations

When using this script, keep the following in mind:

- Archive retrieval from S3 Glacier can take several hours to complete, depending on the retrieval tier selected.
- The script runs indefinitely, continuously polling the queues. You may want to add a termination condition based on your specific requirements.
- Ensure you have sufficient disk space to store all the chunks of your archive.

- If the script is interrupted, you can restart it with `retrieve_archive=False` to continue downloading chunks without initiating a new retrieval job.
- Adjust the *`chunk_size`* and *`workers`* parameters based on your network bandwidth and system resources.
- Standard AWS charges apply for Amazon S3 retrievals, Amazon SNS, and Amazon SQS usage.

Downloading an Archive by Using the REST API

To download an archive by using the REST API

Downloading an archive is a two-step process.

1. Initiate a job of the `archive-retrieval` type. For more information, see Initiate Job (POST jobs).
2. After the job is completed, download the archive data. For more information, see Get Job Output (GET output).

Downloading an Archive in Amazon Glacier Using the AWS CLI

You can download archives in Amazon Glacier (Amazon Glacier) using the AWS Command Line Interface (AWS CLI).

Topics

- (Prerequisite) Setting Up the AWS CLI
- Example: Download an Archive Using the AWS CLI

(Prerequisite) Setting Up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Installing the AWS Command Line Interface

   Configuring the AWS Command Line Interface

2. Verify your AWS CLI setup by entering the following commands at the command prompt. These commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try using the help command.

     ```
     aws help
     ```

   - To get a list of Amazon Glacier vaults on the configured account, use the `list-vaults` command. Replace *123456789012* with your AWS account ID.

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

   - To see the current configuration data for the AWS CLI, use the `aws configure list` command.

     ```
     aws configure list
     ```

Example: Download an Archive Using the AWS CLI

> **Note**
>
> In order to download your archives you must know your archive ids. Steps 1-4 will retrieve your archive ids. If you already know the archive ids you wish to download skip to step 5.

1. Use the `initiate-job` command to start an inventory-retrieval job. The inventory report will list your archive ids.

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

   Expected output:

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

```
}
```

2. Use the `describe-job` command to check status of the previous job command.

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

Expected output:

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

3. Wait for the job to complete.

You must wait until the job output is ready for you to download. If you set a notification configuration on the vault or specified an Amazon Simple Notification Service (Amazon SNS) topic when you initiated the job, Amazon Glacier sends a message to the topic after it completes the job.

You can set notification configuration for specific events on the vault. For more information, see Configuring Vault Notifications in Amazon Glacier. Amazon Glacier sends a message to the specified SNS topic anytime the specific event occurs.

4. When it's complete, use the `get-job-output` command to download the retrieval job to the file `output.json`. This file will contain your archive ids.

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

This command produces a file with the following fields.

```
{
"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. Use the `initiate-job` command to start the retrieval process each archive from a vault. You will need to specify the job parameter as `archive-retrieval` as seen below.

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

6. Wait for the `archive-retrieval` job to complete. Use the `describe-job` command to check status of the previous command.

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

7. When the above job is complete use the `get-job-output` command to download your archive.

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

Deleting an Archive in Amazon Glacier

You cannot delete an archive using the Amazon Glacier (Amazon Glacier) management console. To delete an archive you must use the AWS Command Line Interface (CLI) or write code to make a delete request using either the REST API directly or the AWS SDK for Java and .NET wrapper

libraries. The following topics explain how to use the AWS SDK for Java and .NET wrapper libraries, the REST API, and the AWS CLI.

Topics

You can delete one archive at a time from a vault. To delete the archive you must provide its archive ID in your delete request. You can get the archive ID by downloading the vault inventory for the vault that contains the archive. For more information about downloading the vault inventory, see Downloading a Vault Inventory in Amazon Glacier.

After you delete an archive, you might still be able to make a successful request to initiate a job to retrieve the deleted archive, but the archive retrieval job will fail.

Archive retrievals that are in progress for an archive ID when you delete the archive might or might not succeed according to the following scenarios:

- If the archive retrieval job is actively preparing the data for download when Amazon Glacier receives the delete archive request, then the archival retrieval operation might fail.
- If the archive retrieval job has successfully prepared the archive for download when Amazon Glacier receives the delete archive request, then you will be able to download the output.

For more information about archive retrieval, see Downloading an Archive in Amazon Glacier.

This operation is idempotent. Deleting an already-deleted archive does not result in an error.

After you delete an archive, if you immediately download the vault inventory, it might include the deleted archive in the list because Amazon Glacier prepares vault inventory only about once a day.

> **Note**
>
> For automated deletion of vault archives, see Automated deletion of vault archives in Amazon S3 Glacier.

Deleting an Archive in Amazon Glacier Using the AWS SDK for Java

The following are the steps to delete an archive using the AWS SDK for Java low-level API.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where the archive you want to delete is stored. All operations you perform using this client apply to that AWS Region.
2. Provide request information by creating an instance of the `DeleteArchiveRequest` class.

   You need to provide an archive ID, a vault name, and your account ID. If you don't provide an account ID, then account ID associated with the credentials you provide to sign the request is assumed. For more information, see Using the AWS SDK for Java with Amazon Glacier.
3. Run the `deleteArchive` method by providing the request object as a parameter.

The following Java code snippet illustrates the preceding steps.

```
AmazonGlacierClient client;

DeleteArchiveRequest request = new DeleteArchiveRequest()
    .withVaultName("*** provide a vault name ***")
    .withArchiveId("*** provide an archive ID ***");

client.deleteArchive(request);
```

> **Note**
>
> For information about the underlying REST API, see Delete Archive (DELETE archive).

Example: Deleting an Archive Using the AWS SDK for Java

The following Java code example uses the AWS SDK for Java to delete an archive. For step-by-step instructions on how to run this example, see Running Java Examples for Amazon Glacier Using Eclipse. You need to update the code as shown with a vault name and the archive ID of the archive you want to delete.

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);
        }
    }
}
```

Deleting an Archive in Amazon Glacier Using the AWS SDK for .NET

Both the high-level and low-level APIs provided by the Amazon SDK for .NET provide a method to delete an archive.

Topics

Deleting an Archive Using the High-Level API of the AWS SDK for .NET

The `ArchiveTransferManager` class of the high-level API provides the `DeleteArchive` method you can use to delete an archive.

Example: Deleting an Archive Using the High-Level API of the AWS SDK for .NET

The following C# code example uses the high-level API of the AWS SDK for .NET to delete an archive. For step-by-step instructions on how to run this example, see Running Code Examples. You need to update the code as shown with the archive ID of the archive you want to delete.

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();
    }
  }
}
```

Deleting an Archive Using the Low-Level API AWS SDK for .NET

The following are the steps to delete an using the AWS SDK for .NET.

1. Create an instance of the `AmazonGlacierClient` class (the client).

   You need to specify an AWS Region where the archive you want to delete is stored. All operations you perform using this client apply to that AWS Region.
2. Provide request information by creating an instance of the `DeleteArchiveRequest` class.

   You need to provide an archive ID, a vault name, and your account ID. If you don't provide an account ID, then account ID associated with the credentials you provide to sign the request is assumed. For more information, see Using the AWS SDKs with Amazon Glacier.
3. Run the `DeleteArchive` method by providing the request object as a parameter.

Example: Deleting an Archive Using the Low-Level API of the AWS SDK for .NET

The following C# example illustrates the preceding steps. The example uses the low-level API of the AWS SDK for .NET to delete an archive.

> **Note**
>
> For information about the underlying REST API, see Delete Archive (DELETE archive).

For step-by-step instructions on how to run this example, see Running Code Examples. You need to update the code as shown with the archive ID of the archive you want to delete.

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);
    }
  }
}
```

Deleting an Amazon Glacier Archive Using the REST API

You can use the Delete Archive API to delete an archive.

- For information about the Delete Archive API, see Delete Archive (DELETE archive).
- For information about using the REST API, see API Reference for Amazon Glacier.

Deleting an Archive in Amazon Glacier Using the AWS Command Line Interface

You can delete archives in Amazon Glacier (Amazon Glacier) using the AWS Command Line Interface (AWS CLI).

Topics

(Prerequisite) Setting Up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Installing the AWS Command Line Interface

   Configuring the AWS Command Line Interface

2. Verify your AWS CLI setup by entering the following commands at the command prompt. These commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try using the help command.

     ```
     aws help
     ```

   - To get a list of Amazon Glacier vaults on the configured account, use the `list-vaults` command. Replace *123456789012* with your AWS account ID.

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

   - To see the current configuration data for the AWS CLI, use the `aws configure list` command.

```
aws configure list
```

Example: Deleting an Archive Using the AWS CLI

1. Use the initiate-job command to start an inventory retrieval job.

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

Expected output:

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

2. Use the describe-job command to check status of the previous retrieval job.

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

Expected output:

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

3. Wait for the job to complete.

   You must wait until the job output is ready for you to download. If you set a notification configuration on the vault or specified an Amazon Simple Notification Service (Amazon SNS) topic when you initiated the job, Amazon Glacier sends a message to the topic after it completes the job.

   You can set notification configuration for specific events on the vault. For more information, see Configuring Vault Notifications in Amazon Glacier. Amazon Glacier sends a message to the specified SNS topic anytime the specific event occurs.

4. When it's complete, use the get-job-output command to download the retrieval job to the file `output.json`.

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

   This command produces a file with the following fields.

   ```
   {
   "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. Use the `delete-archive` command to delete each archive from a vault until none remain.

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

Using the AWS SDKs with Amazon Glacier

AWS provides SDKs for you to develop applications for Amazon Glacier. The SDK libraries wrap the underlying Amazon Glacier API, simplifying your programming tasks. For example, for each request sent to Amazon Glacier, you must include a signature to authenticate your requests. When you use the SDK libraries, you need to provide only your AWS security credentials in your code and the libraries compute the necessary signature and include it in the request sent to Amazon Glacier. The AWS SDKs provide libraries that map to the underlying REST API and provide objects that you can use to easily construct requests and process responses.

Topics

The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services, including Amazon Glacier. For information about downloading the AWS CLI, see AWS Command Line Interface. For a list of the Amazon Glacier CLI commands, see the AWS CLI Command Reference.

AWS SDK Libraries for Java and .NET

The AWS SDKs for Java and .NET offer high-level and low-level wrapper libraries.

You can find examples of working with Amazon Glacier by using the AWS SDK for Java and the AWS SDK for .NET throughout this developer guide.

What Is the Low-Level API?

The low-level wrapper libraries map closely the underlying REST API (API Reference for Amazon Glacier) supported by Amazon Glacier. For each Amazon Glacier REST operations, the low-level API provides a corresponding method, a request object for you to provide request information and a response object for you to process Amazon Glacier response. The low-level wrapper libraries are the most complete implementation of the underlying Amazon Glacier operations.

For information about these SDK libraries, see Using the AWS SDK for Java with Amazon Glacier and Using the AWS SDK for .NET with Amazon Glacier.

What Is the High-Level API?

To further simplify application development, these libraries offer a higher-level abstraction for some of the operations. For example:

- Uploading an archive—To upload an archive using the low-level API in addition to the file name and the vault name where you want to save the archive, You need to provide a checksum (SHA-256 tree hash) of the payload. However, the high-level API computes the checksum for you.
- Downloading an archive or vault inventory—To download an archive using the low-level API you first initiate a job, wait for the job to complete, and then get the job output. You need to write additional code to set up an Amazon Simple Notification Service (Amazon SNS) topic for Amazon Glacier to notify you when the job is complete. You also need some polling mechanism to check if a job completion message was posted to the topic. The high-level API provides a method to download an archive that takes care of all these steps. You only specify an archive ID and a folder path where you want to save the downloaded data.

For information about these SDK libraries, see Using the AWS SDK for Java with Amazon Glacier and Using the AWS SDK for .NET with Amazon Glacier.

When to Use the High-Level and Low-Level API

In general, if the high-level API provides methods you need to perform an operation, you should use the high-level API because of the simplicity it provides. However, if the high-level API does not offer the functionality, you can use the low-level API. Additionally, the low-level API allows granular control of the operation such as retry logic in the event of a failure. For example, when uploading an archive the high-level API uses the file size to determine whether to upload the archive in a single operation or use the multipart upload API. The API also has built-in retry logic in case an upload fails. However, your application might need granular control over these decisions, in which case you can use the low-level API.

Using Amazon Glacier with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

SDK documentation	Code examples
AWS SDK for C++	AWS SDK for C++ code examples
AWS CLI	AWS CLI code examples
AWS SDK for Go	AWS SDK for Go code examples
AWS SDK for Java	AWS SDK for Java code examples
AWS SDK for JavaScript	AWS SDK for JavaScript code examples
AWS SDK for Kotlin	AWS SDK for Kotlin code examples
AWS SDK for .NET	AWS SDK for .NET code examples
AWS SDK for PHP	AWS SDK for PHP code examples
AWS Tools for PowerShell	AWS Tools for PowerShell code examples
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) code examples
AWS SDK for Ruby	AWS SDK for Ruby code examples
AWS SDK for Rust	AWS SDK for Rust code examples
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP code examples
AWS SDK for Swift	AWS SDK for Swift code examples

For examples specific to Amazon Glacier, see Code examples for Amazon Glacier using AWS SDKs.

Example availability

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

Using the AWS SDK for Java with Amazon Glacier

The AWS SDK for Java provides both high-level and low-level APIs for Amazon Glacier (Amazon Glacier) as described in Using the AWS SDKs with Amazon Glacier. For information about downloading the AWS SDK for Java, see Amazon SDK for Java.

> **Note**
>
> The AWS SDK for Java provides thread-safe clients for accessing Amazon Glacier. As a best practice, your applications should create one client and reuse the client between threads.

Topics

Using the Low-Level API

The low-level `AmazonGlacierClient` class provides all the methods that map to the underlying REST operations of Amazon Glacier (API Reference for Amazon Glacier). When calling any of these methods, you must create a corresponding request object and provide a response object in which the method can return the Amazon Glacier response to the operation.

For example, the `AmazonGlacierClient` class provides the `createVault` method to create a vault. This method maps to the underlying Create Vault REST operation (see Create Vault (PUT vault)). To use this method, you must create instances of the `CreateVaultResult` object that receives the Amazon Glacier response as shown in the following Java code snippet:

```
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);
```

All the low-level samples in the guide use this pattern.

> **Note**
>
> The preceding code segment specifies `AccountID` when creating the request. However, when using the AWS SDK for Java, the `AccountId` in the request is optional, and therefore all the low-level examples in this guide don't set this value. The `AccountId` is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can specify either the AWS account ID or optionally a '-', in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your Account ID, do not include hyphens in it. When using AWS SDK for Java, if you don't provide the account ID, the library sets the account ID to '-'.

Using the High-Level API

To further simplify your application development, the AWS SDK for Java provides the `ArchiveTransferManager` class that implements a higher-level abstraction for the some of the methods in the low-level API. It provides useful methods, such as the `upload` and `download` methods for archive operations.

For example, the following Java code snippet uses the `upload` high-level method to upload an archive.

```
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();
```

Note that any operations you perform apply to the AWS Region you specified when creating the `ArchiveTransferManager` object. If you don't specify any AWS Region, the AWS SDK for Java sets `us-east-1` as the default AWS Region.

All the high-level examples in this guide use this pattern.

> **Note**
>
> The high-level `ArchiveTransferManager` class can be constructed with an `AmazonGlacierClient` instance or an `AWSCredentials` instance.

Running Java Examples for Amazon Glacier Using Eclipse

The easiest way to get started with the Java code examples is to install the latest AWS Toolkit for Eclipse. For information on installing or updating to the latest toolkit, go to http://aws.amazon.com/eclipse. The following tasks guide you through the creation and testing of the Java code examples provided in this section.

General Process of Creating Java Code Examples

1	Create a default credentials profile for your AWS credentials as described in the AWS SDK for Java topic Providing AWS Credentials in the Amazon SDK for Java.
2	Create a new AWS Java project in Eclipse. The project is pre-configured with the AWS SDK for Java.
3	Copy the code from the section you are reading to your project.
4	Update the code by providing any required data. For example, if uploading a file, provide the file path and the bucket name.
5	Run the code. Verify that the object is created by using the AWS Management Console. For more information about the AWS Management Console, go to http://aws.amazon.com/console/.

Setting the Endpoint

By default, the AWS SDK for Java uses the endpoint `https://glacier.us-east-1.amazonaws.com`. You can set the endpoint explicitly as shown in the following Java code snippets.

The following snippet shows how to set the endpoint to the US West (Oregon) Region (`us-west-2`) in the low-level API.

Example

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

The following snippet shows how to set the endpoint to the US West (Oregon) Region in the high-level API.

```
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);
```

For a list of supported AWS Regions and endpoints, see Accessing Amazon Glacier.

Using the AWS SDK for .NET with Amazon Glacier

The AWS SDK for .NET API is available in `AWSSDK.dll`. For information about downloading the AWS SDK for .NET, go to Sample Code Libraries. As described in Using the AWS SDKs with Amazon Glacier, the AWS SDK for .NET provides both the high-level and low-level APIs.

> **Note**
>
> The low-level API and high-level API provide thread-safe clients for accessing Amazon Glacier. As a best practice, your applications should create one client and reuse the client between threads.

Topics

- Using the Low-Level API
- Using the High-Level API

- Running Code Examples
- Setting the Endpoint

Using the Low-Level API

The low-level `AmazonGlacierClient` class provides all the methods that map to the underlying REST operations of Amazon Glacier (Amazon Glacier) (API Reference for Amazon Glacier). When calling any of these methods, you must create a corresponding request object and provide a response object in which the method can return a Amazon Glacier response to the operation.

For example, the `AmazonGlacierClient` class provides the `CreateVault` method to create a vault. This method maps to the underlying Create Vault REST operation (see Create Vault (PUT vault)). To use this method, you must create instances of the `CreateVaultRequest` and `CreateVaultResponse` classes to provide request information and receive a Amazon Glacier response as shown in the following C# code snippet:

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

CreateVaultRequest request = new CreateVaultRequest()
{
  AccountId = "-",
  VaultName = "*** Provide vault name ***"
};

CreateVaultResponse response = client.CreateVault(request);
```

All the low-level samples in the guide use this pattern.

> **Note**
>
> The preceding code segment specifies `AccountId` when creating the request. However, when using the AWS SDK for .NET, the `AccountId` in the request is optional, and therefore all the low-level examples in this guide don't set this value. The `AccountId` is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can specify either the AWS account ID or optionally a '-', in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign

the request. If you specify your Account ID, do not include hyphens in it. When using AWS SDK for .NET, if you don't provide the account ID, the library sets the account ID to '-'.

Using the High-Level API

To further simplify your application development, the AWS SDK for .NET provides the `ArchiveTransferManager` class that implements a higher-level abstraction for some of the methods in the low-level API. It provides useful methods, such as `Upload` and `Download`, for the archive operations.

For example, the following C# code snippet uses the `Upload` high-level method to upload an archive.

```
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;
```

Note that any operations you perform apply to the AWS Region you specified when creating the `ArchiveTransferManager` object. All the high-level examples in this guide use this pattern.

Note

The high-level `ArchiveTransferManager` class still needs the low-level `AmazonGlacierClient` client, which you can pass either explicitly or the `ArchiveTransferManager` creates the client.

Running Code Examples

The easiest way to get started with the .NET code examples is to install the AWS SDK for .NET. For more information, go to Amazon SDK for .NET.

The following procedure outlines steps for you to test the code examples provided in this guide.

General Process of Creating .NET Code Examples (Using Visual Studio)

1	Create a credentials profile for your AWS credentials as described in the Amazon SDK for .NET topic Configuring AWS credentials.
2	Create a new Visual Studio project using the *AWS Empty Project* template.
3	Replace the code in the project file, `Program.cs` , with the code in the section you are reading.
4	Run the code. Verify that the object is created using the AWS Management Console. For more information about AWS Management Console, go to http://aws.amazon.com/console/.

Setting the Endpoint

By default, the AWS SDK for .NET sets the endpoint to the US West (Oregon) Region (`https://glacier.us-west-2.amazonaws.com`). You can set the endpoint to other AWS Regions as shown in the following C# snippets.

The following snippet shows how to set the endpoint to the US West (Oregon) Region (`us-west-2`) in the low-level API.

Example

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

The following snippet shows how to set the endpoint to the US West (Oregon) Region in the high-level API.

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

For a current list of supported AWS Regions and endpoints, see Accessing Amazon Glacier.

Code examples for Amazon Glacier using AWS SDKs

The following code examples show how to use Amazon Glacier with an AWS software development kit (SDK).

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Code examples

- Basic examples for Amazon Glacier using AWS SDKs
  - Hello Amazon Glacier
  - Actions for Amazon Glacier using AWS SDKs
    - Use AddTagsToVault with an AWS SDK or CLI
    - Use CreateVault with an AWS SDK or CLI
    - Use DeleteArchive with an AWS SDK or CLI
    - Use DeleteVault with an AWS SDK or CLI
    - Use DeleteVaultNotifications with an AWS SDK or CLI
    - Use DescribeJob with an AWS SDK or CLI
    - Use DescribeVault with an AWS SDK or CLI
    - Use GetJobOutput with an AWS SDK or CLI
    - Use GetVaultNotifications with an AWS SDK or CLI
    - Use InitiateJob with an AWS SDK or CLI
    - Use ListJobs with an AWS SDK or CLI
    - Use ListTagsForVault with an AWS SDK or CLI
    - Use ListVaults with an AWS SDK or CLI
    - Use SetVaultNotifications with an AWS SDK or CLI
    - Use UploadArchive with an AWS SDK or CLI

Basic examples for Amazon Glacier using AWS SDKs

The following code examples show how to use the basics of Amazon Glacier with AWS SDKs.

Examples

Hello Amazon Glacier

The following code example shows how to get started using Amazon Glacier.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

- For API details, see ListVaults in *AWS SDK for .NET API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Actions for Amazon Glacier using AWS SDKs

The following code examples demonstrate how to perform individual Amazon Glacier actions with AWS SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

These excerpts call the Amazon Glacier API and are code excerpts from larger programs that must be run in context. You can see actions in context in Scenarios for Amazon Glacier using AWS SDKs.

The following examples include only the most commonly used actions. For a complete list, see the Amazon Glacier API Reference.

Examples

- Use AddTagsToVault with an AWS SDK or CLI
- Use CreateVault with an AWS SDK or CLI
- Use DeleteArchive with an AWS SDK or CLI
- Use DeleteVault with an AWS SDK or CLI
- Use DeleteVaultNotifications with an AWS SDK or CLI
- Use DescribeJob with an AWS SDK or CLI
- Use DescribeVault with an AWS SDK or CLI
- Use GetJobOutput with an AWS SDK or CLI
- Use GetVaultNotifications with an AWS SDK or CLI
- Use InitiateJob with an AWS SDK or CLI
- Use ListJobs with an AWS SDK or CLI
- Use ListTagsForVault with an AWS SDK or CLI
- Use ListVaults with an AWS SDK or CLI
- Use SetVaultNotifications with an AWS SDK or CLI
- Use UploadArchive with an AWS SDK or CLI
- Use UploadMultipartPart with an AWS SDK or CLI

Use `AddTagsToVault` with an AWS SDK or CLI

The following code examples show how to use `AddTagsToVault`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /// <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;
   }
```

- For API details, see AddTagsToVault in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

The following command adds two tags to a vault named `my-vault`:

```
aws glacier add-tags-to-vault --account-id - --vault-name my-vault --
tags id=1234,date=july2015
```

Amazon Glacier requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

- For API details, see AddTagsToVault in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateVault` with an AWS SDK or CLI

The following code examples show how to use `CreateVault`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Archive a file, get notifications, and initiate a job

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/// <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;
    }
```

- For API details, see CreateVault in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

The following command creates a new vault named `my-vault`:

```
aws glacier create-vault --vault-name my-vault --account-id -
```

Amazon Glacier requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

- For API details, see CreateVault in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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);
        }
    }
}
```

- For API details, see CreateVault in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the client.

```
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 };
```

Create the vault.

```
// 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();
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see CreateVault in *AWS SDK for JavaScript API Reference*.

SDK for JavaScript (v2)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
// 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!");
  }
});
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see CreateVault in *AWS SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Creates a new vault for the user's account. As no value was supplied to the -AccountId parameter the cmdlets uses a default of "-" indicating the current account.

```
New-GLCVault -VaultName myvault
```

Output:

```
/01234567812/vaults/myvault
```

- For API details, see CreateVault in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Creates a new vault for the user's account. As no value was supplied to the -AccountId parameter the cmdlets uses a default of "-" indicating the current account.

```
New-GLCVault -VaultName myvault
```

Output:

```
/01234567812/vaults/myvault
```

- For API details, see CreateVault in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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
```

- For API details, see CreateVault in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteArchive` with an AWS SDK or CLI

The following code examples show how to use `DeleteArchive`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Get archive content and delete the archive

CLI

AWS CLI

To delete an archive from a vault

The following `delete-archive` example removes the specified archive from `example_vault`.

```
aws glacier delete-archive \
    --account-id 111122223333  \
    --vault-name example_vault \
    --archive-id Sc0u9ZP8yaWkmh-XGlIvAVprtLhaLCGnNwNl5I5x9HqPIkX5mjc0DrId3Ln-
Gi_k2HzmlIDZUz117KSdVMdMXLuFWi9PJUitxWO73edQ43eTlMWkH0pd9zVSAuV_XXZBVhKhyGhJ7w
```

This command produces no output.

- For API details, see DeleteArchive in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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);
        }
    }
}
```

- For API details, see DeleteArchive in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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
```

- For API details, see DeleteArchive in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteVault` with an AWS SDK or CLI

The following code examples show how to use `DeleteVault`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Get archive content and delete the archive

CLI

AWS CLI

The following command deletes a vault named `my-vault`:

```
aws glacier delete-vault --vault-name my-vault --account-id -
```

This command does not produce any output. Amazon Glacier requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

- For API details, see DeleteVault in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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);
        }
    }
}
```

- For API details, see DeleteVault in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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
```

- For API details, see DeleteVault in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DeleteVaultNotifications` with an AWS SDK or CLI

The following code examples show how to use `DeleteVaultNotifications`.

CLI

AWS CLI

To remove the SNS notifications for a vault

The following `delete-vault-notifications` example removes notifications sent by Amazon Simple Notification Service (Amazon SNS) for the specified vault.

```
aws glacier delete-vault-notifications \
    --account-id 111122223333 \
    --vault-name example_vault
```

This command produces no output.

- For API details, see DeleteVaultNotifications in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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
```

- For API details, see DeleteVaultNotifications in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeJob` with an AWS SDK or CLI

The following code examples show how to use `DescribeJob`.

CLI

AWS CLI

The following command retrieves information about an inventory retrieval job on a vault named `my-vault`:

```
aws glacier describe-job --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61ds04CnMW
```

Output:

```
{
```

```
    "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"
}
```

The job ID can be found in the output of `aws glacier initiate-job` and `aws glacier list-jobs`. Amazon Glacier requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

- For API details, see DescribeJob in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Returns details of the specified job. When the job completes successfully the Read-GCJobOutput cmdlet can be used to retrieve the contents of the job (an archive or inventory list) to the local file system.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Output:

```
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
```

- For API details, see DescribeJob in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Returns details of the specified job. When the job completes successfully the Read-GCJobOutput cmdlet can be used to retrieve the contents of the job (an archive or inventory list) to the local file system.

```
Get-GLCJob -VaultName myvault -JobId "op1x...JSbthM"
```

Output:

```
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
```

- For API details, see DescribeJob in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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
```

- For API details, see DescribeJob in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DescribeVault` with an AWS SDK or CLI

The following code examples show how to use `DescribeVault`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/// <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;
    }
```

- For API details, see DescribeVault in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

The following command retrieves data about a vault named `my-vault`:

```
aws glacier describe-vault --vault-name my-vault --account-id -
```

Amazon Glacier requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

- For API details, see DescribeVault in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetJobOutput` with an AWS SDK or CLI

The following code examples show how to use `GetJobOutput`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Get archive content and delete the archive

CLI

AWS CLI

The following command saves the output from a vault inventory job to a file in the current directory named `output.json`:

```
aws glacier get-job-output --account-id - --vault-name my-
vault --job-id zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW output.json
```

The `job-id` is available in the output of `aws glacier list-jobs`. Note that the output file name is a positional argument that is not prefixed by an option name. Amazon Glacier requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

Output:

```
{
    "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
```

- For API details, see GetJobOutput in *AWS CLI Command Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Downloads the archive content that was scheduled for retrieval in the specified job and stores the contents into a file on disk. The download validates the checksum for you, if one is available. If desired the entire response including the checksum can be returned by specifying `-Select '*'`.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- For API details, see GetJobOutput in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Downloads the archive content that was scheduled for retrieval in the specified job and stores the contents into a file on disk. The download validates the checksum for you, if one is available. If desired the entire response including the checksum can be returned by specifying `-Select '*'`.

```
Read-GLCJobOutput -VaultName myvault -JobId "HSWjArc...Zq2XLiW" -FilePath "c:
\temp\blue.bin"
```

- For API details, see GetJobOutput in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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
```

- For API details, see GetJobOutput in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetVaultNotifications` with an AWS SDK or CLI

The following code examples show how to use `GetVaultNotifications`.

CLI

AWS CLI

The following command gets a description of the notification configuration for a vault named `my-vault`:

```
aws glacier get-vault-notifications --account-id - --vault-name my-vault
```

Output:

```
{
    "vaultNotificationConfig": {
        "Events": [
            "InventoryRetrievalCompleted",
            "ArchiveRetrievalCompleted"
        ],
        "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault"
    }
}
```

If no notifications have been configured for the vault, an error is returned. Amazon Glacier requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

- For API details, see GetVaultNotifications in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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
```

- For API details, see GetVaultNotifications in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `InitiateJob` with an AWS SDK or CLI

The following code examples show how to use `InitiateJob`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Archive a file, get notifications, and initiate a job

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Retrieve an archive from a vault. This example uses the ArchiveTransferManager class. For API details see 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}%");
        }
    }
```

- For API details, see InitiateJob in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

The following command initiates a job to get an inventory of the vault `my-vault`:

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

Output:

```
{
    "location": "/0123456789012/vaults/my-vault/jobs/
zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW",
    "jobId": "zbxcm3Z_3z5UkoroF7SuZKrxgGoDc3RloGduS7Eg-
RO47Yc6FxsdGBgf_Q2DK5Ejh18CnTS5XW4_XqlNHS61dsO4CnMW"
}
```

Amazon Glacier requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

The following command initiates a job to retrieve an archive from the vault `my-vault`:

```
aws glacier initiate-job --account-id - --vault-name my-vault --job-
parameters file://job-archive-retrieval.json
```

`job-archive-retrieval.json` is a JSON file in the local folder that specifies the type of job, archive ID, and some optional parameters:

```
{
  "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"
}
```

Archive IDs are available in the output of `aws glacier upload-archive` and `aws glacier get-job-output`.

Output:

```
{
```

```
    "location": "/011685312445/vaults/mwunderl/jobs/l7IL5-
EkXyEY9Ws95fClzIbk2O5uLYaFdAYOi-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav",
    "jobId": "l7IL5-EkXy2O5uLYaFdAYOiEY9Ws95fClzIbk-
azsX_Z8V6NH4yERHzars8wTKYQMX6nBDI9cMNHzyZJO59-8N9aHWav"
}
```

See Initiate Job in the *Amazon Glacier API Reference* for details on the job parameters format.

- For API details, see InitiateJob in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Retrieve a vault inventory.

```
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);

        }
    }
}
```

- For API details, see [InitiateJob](#) in *AWS SDK for Java 2.x API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Starts a job to retrieve an archive from the specified vault owned by the user. The status of the job can be checked using the Get-GLCJob cmdlet. When the job completes successfully the Read-GCJobOutput cmdlet can be used to retrieve the contents of the archive to the local file system.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

Output:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- For API details, see InitiateJob in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Starts a job to retrieve an archive from the specified vault owned by the user. The status of the job can be checked using the Get-GLCJob cmdlet. When the job completes successfully the Read-GCJobOutput cmdlet can be used to retrieve the contents of the archive to the local file system.

```
Start-GLCJob -VaultName myvault -JobType "archive-retrieval" -JobDescription
 "archive retrieval" -ArchiveId "o9O9j...TX-TpIhQJw"
```

Output:

```
JobId            JobOutputPath Location
-----            ------------- --------
op1x...JSbthM                  /012345678912/vaults/test/jobs/
op1xe...I4HqCHkSJSbthM
```

- For API details, see InitiateJob in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Retrieve a vault inventory.

```
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
```

Retrieve an archive from a vault.

```
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
```

- For API details, see InitiateJob in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListJobs` with an AWS SDK or CLI

The following code examples show how to use `ListJobs`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code examples:

- Archive a file, get notifications, and initiate a job
- Get archive content and delete the archive

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /// <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;
    }
```

- For API details, see ListJobs in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

The following command lists in-progress and recently completed jobs for a vault named `my-vault`:

```
aws glacier list-jobs --account-id - --vault-name my-vault
```

Output:

```
{
    "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 requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

- For API details, see ListJobs in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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
```

- For API details, see ListJobs in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListTagsForVault` with an AWS SDK or CLI

The following code examples show how to use `ListTagsForVault`.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
    /// <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;
    }
```

- For API details, see ListTagsForVault in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

The following command lists the tags applied to a vault named `my-vault`:

```
aws glacier list-tags-for-vault --account-id - --vault-name my-vault
```

Output:

```
{
    "Tags": {
        "date": "july2015",
        "id": "1234"
    }
}
```

Amazon Glacier requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

- For API details, see ListTagsForVault in *AWS CLI Command Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListVaults` with an AWS SDK or CLI

The following code examples show how to use `ListVaults`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Archive a file, get notifications, and initiate a job

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
/// <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;
}
```

- For API details, see ListVaults in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

The following command lists the vaults in the default account and region:

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

Output:

```
{
    "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 requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

- For API details, see ListVaults in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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);
        }
    }
}
```

- For API details, see ListVaults in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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
```

- For API details, see ListVaults in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `SetVaultNotifications` with an AWS SDK or CLI

The following code examples show how to use `SetVaultNotifications`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Archive a file, get notifications, and initiate a job

CLI

AWS CLI

The following command configures SNS notifications for a vault named `my-vault`:

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

`notificationconfig.json` is a JSON file in the current folder that specifies an SNS topic and the events to publish:

```
{
  "SNSTopic": "arn:aws:sns:us-west-2:0123456789012:my-vault",
  "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"]
}
```

Amazon Glacier requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

- For API details, see SetVaultNotifications in *AWS CLI Command Reference*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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
```

- For API details, see SetVaultNotifications in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UploadArchive` with an AWS SDK or CLI

The following code examples show how to use `UploadArchive`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Archive a file, get notifications, and initiate a job

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
   /// <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;
       }
   }
```

- For API details, see UploadArchive in *AWS SDK for .NET API Reference*.

CLI

AWS CLI

The following command uploads an archive in the current folder named `archive.zip` to a vault named `my-vault`:

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

Output:

```
{
    "archiveId": "kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--
zM_mw6k76ZFGEIWQX-ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw",
    "checksum":
 "969fb39823836d81f0cc028195fcdbcbbe76cdde932d4646fa7de5f21e18aa67",
    "location": "/0123456789012/vaults/my-vault/archives/
kKB7ymWJVpPSwhGP6ycSOAekp9ZYe_--zM_mw6k76ZFGEIWQX-
ybtRDvc2VkPSDtfKmQrj0IRQLSGsNuDp-
AJVlu2ccmDSyDUmZwKbwbpAdGATGDiB3hHO0bjbGehXTcApVud_wyDw"
}
```

Amazon Glacier requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

To retrieve an uploaded archive, initiate a retrieval job with the aws glacier initiate-job command.

- For API details, see UploadArchive in *AWS CLI Command Reference*.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
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();
    }
}
```

- For API details, see UploadArchive in *AWS SDK for Java 2.x API Reference*.

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create the client.

```
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 };
```

Upload the archive.

```
// 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();
```

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see UploadArchive in *AWS SDK for JavaScript API Reference*.

SDK for JavaScript (v2)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

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

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see UploadArchive in *AWS SDK for JavaScript API Reference*.

PowerShell

Tools for PowerShell V4

Example 1: Uploads a single file to the specified vault, returning the archive ID and computed checksum.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Output:

```
FilePath                  ArchiveId             Checksum
--------                  ---------             --------
C:\temp\blue.bin          o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

Example 2: Uploads the contents of a folder hierarchy to the specified vault in the user's account. For each file uploaded the cmdlet emits the filename, corresponding archive ID and the computed checksum of the archive.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Output:

```
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
```

- For API details, see UploadArchive in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: Uploads a single file to the specified vault, returning the archive ID and computed checksum.

```
Write-GLCArchive -VaultName myvault -FilePath c:\temp\blue.bin
```

Output:

```
FilePath                    ArchiveId              Checksum
--------                    ---------              --------
C:\temp\blue.bin            o9O9jUUs...TTX-TpIhQJw 79f3e...f4395b
```

Example 2: Uploads the contents of a folder hierarchy to the specified vault in the user's account. For each file uploaded the cmdlet emits the filename, corresponding archive ID and the computed checksum of the archive.

```
Write-GLCArchive -VaultName myvault -FolderPath . -Recurse
```

Output:

```
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
```

- For API details, see [UploadArchive](#) in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see UploadArchive in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UploadMultipartPart` with an AWS SDK or CLI

The following code examples show how to use `UploadMultipartPart`.

CLI

AWS CLI

The following command uploads the first 1 MiB (1024 x 1024 bytes) part of an archive:

```
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 requires an account ID argument when performing operations, but you can use a hyphen to specify the in-use account.

The body parameter takes a path to a part file on the local filesystem. The range parameter takes an HTTP content range indicating the bytes that the part occupies in the completed archive. The upload ID is returned by the `aws glacier initiate-multipart-upload` command and can also be obtained by using `aws glacier list-multipart-uploads`.

For more information on multipart uploads to Amazon Glacier using the AWS CLI, see Using Amazon Glacier in the *AWS CLI User Guide*.

- For API details, see UploadMultipartPart in *AWS CLI Command Reference*.

JavaScript

SDK for JavaScript (v2)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a multipart upload of 1 megabyte chunks of a Buffer object.

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

- For more information, see AWS SDK for JavaScript Developer Guide.
- For API details, see UploadMultipartPart in *AWS SDK for JavaScript API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Scenarios for Amazon Glacier using AWS SDKs

The following code examples show you how to implement common scenarios in Amazon Glacier with AWS SDKs. These scenarios show you how to accomplish specific tasks by calling multiple functions within Amazon Glacier or combined with other AWS services. Each scenario includes a link to the complete source code, where you can find instructions on how to set up and run the code.

Scenarios target an intermediate level of experience to help you understand service actions in context.

Examples

- Archive a file to Amazon Glacier, get notifications, and initiate a job using an AWS SDK
- Get Amazon Glacier archive content and delete the archive using an AWS SDK

Archive a file to Amazon Glacier, get notifications, and initiate a job using an AWS SDK

The following code example shows how to:

- Create an Amazon Glacier vault.
- Configure the vault to publish notifications to an Amazon SNS topic.
- Upload an archive file to the vault.
- Initiate an archive retrieval job.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a class that wraps Amazon Glacier operations.

```
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
```

Call functions on the wrapper class to create a vault and upload a file, then configure the vault to publish notifications and initiate a job to retrieve the archive.

```
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.")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - CreateVault
  - InitiateJob
  - ListJobs
  - ListVaults

- SetVaultNotifications
- UploadArchive

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Get Amazon Glacier archive content and delete the archive using an AWS SDK

The following code example shows how to:

- List jobs for an Amazon Glacier vault and get job status.
- Get the output of a completed archive retrieval job.
- Delete an archive.
- Delete a vault.

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Create a class that wraps Amazon Glacier operations.

```
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
```

Call functions on the wrapper class to get archive content from a completed job, then delete the archive.

```
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)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - DeleteArchive
  - DeleteVault
  - GetJobOutput

- ListJobs

For a complete list of AWS SDK developer guides and code examples, see Using Amazon Glacier with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Security in Amazon Glacier

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. The effectiveness of our security is regularly tested and verified by third-party auditors as part of the AWS compliance programs. To learn about the compliance programs that apply to Amazon Glacier (Amazon Glacier), see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This documentation will help you understand how to apply the shared responsibility model when using Amazon Glacier. The following topics show you how to configure Amazon Glacier to meet your security and compliance objectives. You'll also learn how to use other AWS services that can help you to monitor and secure your Amazon Glacier resources.

Topics

Data Protection in Amazon Glacier

Amazon Glacier (Amazon Glacier) provides highly durable cloud storage for data archiving and long-term backup. Amazon Glacier is designed to deliver 99.999999999 percent durability and

provides comprehensive security and compliance capabilities that can help you meet stringent regulatory requirements. Amazon Glacier redundantly stores data in multiple AWS Availability Zones (AZ) and on multiple devices within each AZ. To increase durability, Amazon Glacier synchronously stores your data across multiple AZs before confirming a successful upload.

For more information about the AWS global cloud infrastructure, see Global Infrastructure.

For data protection purposes, we recommend that you protect AWS account credentials and give individual users, groups, or roles only the permissions necessary to fulfill their job duties.

If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

Topics

Data Encryption

Data protection refers to protecting data while in-transit (as it travels to and from Amazon Glacier) and at rest (while it is stored in AWS data centers). You can protect data in transit that is uploaded directly to Amazon Glacier using Secure Sockets Layer (SSL) or client-side encryption.

You can also access Amazon Glacier through Amazon S3. Amazon S3 supports lifecycle configuration on an Amazon S3 bucket, which enables you to transition objects to the Amazon Glacier storage class for archival. Data in transit between Amazon S3 and Amazon Glacier via lifecycle policies is encrypted using SSL.

Data at rest stored in Amazon Glacier is automatically server-side encrypted using 256-bit Advanced Encryption Standard (AES-256) with keys maintained by AWS. If you prefer to manage your own keys, you can also use client-side encryption before storing data in Amazon Glacier. For more information about how to setup default encryption for Amazon S3, see Amazon S3 Default Encryption in the *Amazon Simple Storage Service User Guide*.

Key Management

Server-side encryption addresses data encryption at rest—that is, Amazon Glacier encrypts your data as it writes it to its data centers and decrypts it for you when you access it. As long as you authenticate your request and you have access permissions, there is no difference in the way you access encrypted or unencrypted data.

Data at rest stored in Amazon Glacier is automatically server-side encrypted using AES-256, using keys maintained by AWS. As an additional safeguard, AWS encrypts the key itself with a root key that we regularly rotate.

Internetwork Traffic Privacy

Access to Amazon Glacier via the network is through AWS published APIs. Clients must support Transport Layer Security (TLS) 1.2. We recommend TLS 1.3 or later. Clients must also support cipher suites with Perfect Forward Secrecy (PFS), such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). Most modern systems such as Java 7 and later support these modes. Additionally, you must sign requests using an access key ID and a secret access key that are associated with an IAM principal, or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

VPC Endpoints

A virtual private cloud (VPC) endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Although Amazon Glacier does not support VPC endpoints directly, you can take advantage of Amazon Simple Storage Service (Amazon S3) VPC endpoints if you access Amazon Glacier as a storage tier integrated with Amazon S3.

For more information about Amazon S3 lifecycle configuration and transitioning objects to the Amazon Glacier storage class, see Object Lifecycle Management and Transitioning Objects in the *Amazon Simple Storage Service User Guide*. For more information about VPC endpoints, see VPC Endpoints in the *Amazon VPC User Guide*.

Identity and Access Management for Amazon Glacier

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon Glacier resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting Amazon Glacier identity and access)
- **Service administrator** - determine user access and submit permission requests (see How Amazon Glacier works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for Amazon Glacier)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon Glacier works with IAM

Before you use IAM to manage access to Amazon Glacier, learn what IAM features are available to use with Amazon Glacier.

IAM features you can use with Amazon Glacier

IAM feature	Amazon Glacier support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No

IAM feature	Amazon Glacier support
ABAC (tags in policies)	No
Temporary credentials	Yes
Principal permissions	No
Service roles	No
Service-linked roles	No

To get a high-level view of how Amazon Glacier and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Amazon Glacier

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Amazon Glacier

To view examples of Amazon Glacier identity-based policies, see Identity-based policy examples for Amazon Glacier.

Resource-based policies within Amazon Glacier

Supports resource-based policies: Yes

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that

support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

The Amazon Glacier service supports only one type of resource-based policy called a *vault policy*, which is attached to a vault. This policy defines which principals can perform actions on the vault.

Amazon Glacier vault policies manage permissions in the following ways:

- Manage user permissions in an account using a single vault policy, instead of more than one individual user policies.
- Manage cross-account permissions as an alternative to using IAM roles.

Resource-based policy examples within Amazon Glacier

To view examples of Amazon Glacier resource-based policies, see Resource-based policy examples for Amazon Glacier.

Policy actions for Amazon Glacier

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Amazon Glacier actions, see Actions defined by Amazon Glacier in the *Service Authorization Reference*.

Policy actions in Amazon Glacier use the following prefix before the action:

```
glacier
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
            "glacier:CreateVault",
            "glacier:DescribeVault",
            "glacier:ListVaults"
          ]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "glacier:GetVault*"
```

To view examples of Amazon Glacier identity-based policies, see Identity-based policy examples for Amazon Glacier.

Policy resources for Amazon Glacier

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of Amazon Glacier resource types and their ARNs, see Resources defined by Amazon Glacier in the *Service Authorization Reference*. To learn which actions you can specify the ARN of each resource, see Actions defined by Amazon Glacier.

In Amazon Glacier, the primary resource is a *vault*. Amazon Glacier supports policies only at the vault level. That is, in an IAM policy, the `Resource` value that you specify can be a specific vault or a set of vaults in a specific AWS Region. Amazon Glacier doesn't support archive-level permissions.

For all Amazon Glacier actions, `Resource` specifies the vault on which you want to grant the permissions. These resources have unique Amazon Resource Names (ARNs) associated with them as shown in the following table, and you can use a wildcard character (*) in the ARN to match vault names that start with the same prefix.

Amazon Glacier provides a set of operations to work with the Amazon Glacier resources. For information on the available operations, see API Reference for Amazon Glacier.

Some Amazon Glacier API actions support multiple resources. For example, `glacier:AddTagsToVault` accesses examplevault1 and examplevault2, so a principal must have permissions to access both resources. To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [
""arn:aws:glacier:us-west-2:123456789012:vaults/examplevault1",",
""arn:aws:glacier:us-west-2:123456789012:vaults/examplevault2","
]
```

Policy condition keys for Amazon Glacier

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Amazon Glacier condition keys, see Condition keys for Amazon Glacier in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon Glacier.

For examples of using the glacier–specific condition keys, see Vault Lock Policies.

ACLs in Amazon Glacier

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with Amazon Glacier

Supports ABAC (tags in policies): No

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with Amazon Glacier

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

Cross-service principal permissions for Amazon Glacier

Supports forward access sessions (FAS): No

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for Amazon Glacier

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Amazon Glacier functionality. Edit service roles only when Amazon Glacier provides guidance to do so.

Service-linked roles for Amazon Glacier

Supports service-linked roles: No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a Yes in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for Amazon Glacier

By default, users and roles don't have permission to create or modify Amazon Glacier resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by Amazon Glacier, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Glacier in the *Service Authorization Reference*.

The following is an example policy that grants permissions for three Amazon Glacier vault-related actions (`glacier:CreateVault`, `glacier:DescribeVault` and `glacier:ListVaults`) on a resource, using the Amazon Resource Name (ARN) that identifies all of the vaults in the `us-west-2` AWS Region. ARNs uniquely identify AWS resources. For more information about ARNs used with Amazon Glacier, see Policy resources for Amazon Glacier.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
         "glacier:CreateVault",
         "glacier:DescribeVault",
         "glacier:ListVaults"
         ],
         "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/*"
      }
   ]
}
```

The policy grants permissions to create, list, and obtain descriptions of vaults in the `us-west-2` Region. The wildcard character (*) at the end of the ARN means that this statement can match any vault name.

Important

When you grant permissions to create a vault using the `glacier:CreateVault` operation, you must specify a wildcard character (*) because you don't know the vault name until after you create the vault.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon Glacier resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the Amazon Glacier console

To access the Amazon Glacier console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon Glacier resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

The Amazon Glacier console provides an integrated environment for you to create and manage Amazon Glacier vaults. At a minimum IAM identities that you create must be granted permissions for the `glacier:ListVaults` action to view the Amazon Glacier console as shown in the following example.

JSON

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

```
        }
```

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see AWS Managed Policies in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to Amazon Glacier:

- **AmazonGlacierReadOnlyAccess** – Grants read only access to Amazon Glacier through the AWS Management Console.
- **AmazonGlacierFullAccess** – Grants full access to Amazon Glacier through the AWS Management Console.

You can also create your own custom IAM policies to allow permissions for Amazon Glacier API actions and resources. You can attach these custom policies to the custom IAM roles that you create for your Amazon Glacier vaults.

Both of the Amazon Glacier AWS Managed policies discussed in the next section grant permissions for `glacier:ListVaults`.

For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "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": "*"
        }
    ]
}
```

Customer Managed Policy Examples

In this section, you can find example user policies that grant permissions for various Amazon Glacier actions. These policies work when you are using Amazon Glacier REST API, the Amazon SDKs, the AWS CLI, or, if applicable, the Amazon Glacier management console.

> **Note**
>
> All examples use the US West (Oregon) Region (`us-west-2`) and contain fictitious account IDs.

Examples

- Example 1: Allow a User to Download Archives from a Vault
- Example 2: Allow a User to Create a Vault and Configure Notifications

- Example 3: Allow a User to Upload Archives to a Specific Vault
- Example 4: Allow a User Full Permissions on a Specific Vault

Example 1: Allow a User to Download Archives from a Vault

To download an archive, you first initiate a job to retrieve the archive. After the retrieval job is complete, you can download the data. The following example policy grants permissions for the `glacier:InitiateJob` action to initiate a job (which allows the user to retrieve an archive or a vault inventory from the vault), and permissions for the `glacier:GetJobOutput` action to download the retrieved data. The policy also grants permissions to perform the `glacier:DescribeJob` action so that the user can get the job status. For more information, see Initiate Job (POST jobs).

The policy grants these permissions on a vault named `examplevault`. You can get the vault ARN from the Amazon Glacier console, or programmatically by calling either the Describe Vault (GET vault) or the List Vaults (GET vaults) API actions.

Example 2: Allow a User to Create a Vault and Configure Notifications

The following example policy grants permissions to create a vault in the us-west-2 Region as specified in the `Resource` element and configure notifications. For more information about working with notifications, see Configuring Vault Notifications in Amazon Glacier. The policy also grants permissions to list vaults in the AWS Region and get a specific vault description.

> **Important**
>
> When you grant permissions to create a vault using the `glacier:CreateVault` operation, you must specify a wildcard character (*) in the `Resource` value because you don't know the vault name until after you create the vault.

Example 3: Allow a User to Upload Archives to a Specific Vault

The following example policy grants permissions to upload archives to a specific vault in the us-west-2 Region. These permissions allow a user to upload an archive all at once using the Upload Archive (POST archive) API operation or in parts using the Initiate Multipart Upload (POST multipart-uploads) API operation.

Example 4: Allow a User Full Permissions on a Specific Vault

The following example policy grants permissions for all Amazon Glacier actions on a vault named `examplevault`.

Resource-based policy examples for Amazon Glacier

A Amazon Glacier vault can have one vault access policy and one Vault Lock policy associated with it. A Amazon Glacier *vault access policy* is a resource-based policy that you can use to manage permissions to your vault. A *Vault Lock policy* is vault access policy that can be locked. After you lock a Vault Lock policy, the policy can't be changed. You can use a Vault Lock Policy to enforce compliance controls.

Topics

- Vault Access Policies
- Vault Lock Policies

Vault Access Policies

An Amazon Glacier vault access policy is a resource-based policy that you can use to manage permissions to your vault.

You can create one vault access policy for each vault to manage *permissions*. You can modify permissions in a vault access policy at any time. Amazon Glacier also supports a Vault Lock policy on each vault that, after you lock it, cannot be altered. For more information about working with Vault Lock policies, see Vault Lock Policies.

Examples

- Example 1: Grant Cross-Account Permissions for Specific Amazon Glacier Actions
- Example 2: Grant Cross-Account Permissions for MFA Delete Operations

Example 1: Grant Cross-Account Permissions for Specific Amazon Glacier Actions

The following example policy grants cross-account permissions to two AWS accounts for a set of Amazon Glacier operations on a vault named `examplevault`.

> **Note**
>
> The account that owns the vault is billed for all costs associated with the vault. All requests, data transfer, and retrieval costs made by allowed external accounts are billed to the account that owns the vault.

Example 2: Grant Cross-Account Permissions for MFA Delete Operations

You can use multi-factor authentication (MFA) to protect your Amazon Glacier resources. To provide an extra level of security, MFA requires users to prove physical possession of an MFA device by providing a valid MFA code. For more information about configuring MFA access, see Configuring MFA-Protected API Access in the *IAM User Guide*.

The example policy grants an AWS account with temporary credentials permission to delete archives from a vault named examplevault, provided the request is authenticated with an MFA device. The policy uses the `aws:MultiFactorAuthPresent` condition key to specify this additional requirement. For more information, see Available Keys for Conditions in the *IAM User Guide*.

Vault Lock Policies

An Amazon Glacier (Amazon Glacier) vault can have one resource-based vault access policy and one Vault Lock policy attached to it. A *Vault Lock policy* is a vault access policy that you can lock. Using a Vault Lock policy can help you enforce regulatory and compliance requirements. Amazon Glacier provides a set of API operations for you to manage the Vault Lock policies, see Locking a Vault by Using the Amazon Glacier API.

As an example of a Vault Lock policy, suppose that you are required to retain archives for one year before you can delete them. To implement this requirement, you can create a Vault Lock policy that denies users permissions to delete an archive until the archive has existed for one year. You can test this policy before locking it down. After you lock the policy, the policy becomes immutable. For more information about the locking process, see Vault Lock Policies. If you want to manage other user permissions that can be changed, you can use the vault access policy (see Vault Access Policies).

You can use the Amazon Glacier API, Amazon SDKs, AWS CLI, or the Amazon Glacier console to create and manage Vault Lock policies. For a list of Amazon Glacier actions allowed for vault resource-based policies, see API Permissions Reference.

Examples

- Example 1: Deny Deletion Permissions for Archives Less Than 365 Days Old
- Example 2: Deny Deletion Permissions Based on a Tag

Example 1: Deny Deletion Permissions for Archives Less Than 365 Days Old

Suppose that you have a regulatory requirement to retain archives for up to one year before you can delete them. You can enforce that requirement by implementing the following Vault Lock policy. The policy denies the `glacier:DeleteArchive` action on the examplevault vault if the archive being deleted is less than one year old. The policy uses the Amazon Glacier-specific condition key `ArchiveAgeInDays` to enforce the one-year retention requirement.

Example 2: Deny Deletion Permissions Based on a Tag

Suppose that you have a time-based retention rule that an archive can be deleted if it is less than a year old. At the same time, suppose that you need to place a legal hold on your archives to prevent deletion or modification for an indefinite duration during a legal investigation. In this case, the legal hold takes precedence over the time-based retention rule specified in the Vault Lock policy.

To put these two rules in place, the following example policy has two statements:

- The first statement denies deletion permissions for everyone, locking the vault. This lock is performed by using the `LegalHold` tag.
- The second statement grants deletion permissions when the archive is less than 365 days old. But even when archives are less than 365 days old, no one can delete them when the condition in the first statement is met.

Troubleshooting Amazon Glacier identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon Glacier and IAM.

Topics

- I am not authorized to perform an action in Amazon Glacier
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Amazon Glacier resources

I am not authorized to perform an action in Amazon Glacier

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `glacier:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 glacier:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `glacier:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Amazon Glacier.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon Glacier. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Amazon Glacier resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon Glacier supports these features, see How Amazon Glacier works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

API Permissions Reference

When you are setting up How Amazon Glacier works with IAM and writing a permissions policy that you can attach to an IAM identity (identity-based policies) or a resource (resource-based policies), you can use the following table as a reference. The list includes each Amazon Glacier API operation, the corresponding actions for which you can grant permissions to perform the action, and the AWS resource for which you can grant the permissions.

You specify the actions in the policy's `Action` element, and you specify the resource value in the policy's `Resource` element. Also, you can use the IAM policy language `Condition` element to specify when a policy should take effect.

To specify an action, use the `glacier:` prefix followed by the API operation name (for example, `glacier:CreateVault`). For most Amazon Glacier actions, `Resource` is the vault on which you want to grant the permissions. You specify a vault as the `Resource` value by using the vault ARN.

To express conditions, you use predefined condition keys. For more information, see Resource-based policies within Amazon Glacier.

The following table lists actions that can be used with identity-based policies and resource-based policies.

> **Note**
>
> Some actions can only be used with identity-based policies. These actions are marked by an asterisk (*) after the name of the API operation in the first column.

Amazon Glacier API and Required Permissions for Actions

Abort Multipart Upload (DELETE uploadID)

Required Permissions (API Actions): `glacier:AbortMultipartUpload`

Resources: `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/vault-name`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/example*`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/*`

Amazon Glacier Condition Keys:

Abort Vault Lock (DELETE lock-policy)

Required Permissions (API Actions): `glacier:AbortVaultLock`

Resources:

Amazon Glacier Condition Keys:

Add Tags To Vault (POST tags add)

Required Permissions (API Actions):`glacier:AddTagsToVault`

Resources: `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/vault-name`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/example*`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/*`

Amazon Glacier Condition Keys: `glacier:ResourceTag/`*`TagKey`*

Complete Multipart Upload (POST uploadID)

Required Permissions (API Actions):`glacier:CompleteMultipartUpload`

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys: glacier:ResourceTag/*TagKey*

Complete Vault Lock (POST lockId)

Required Permissions (API Actions):glacier:CompleteVaultLock

Resources:

Amazon Glacier Condition Keys: glacier:ResourceTag/*TagKey*

Create Vault (PUT vault) *

Required Permissions (API Actions):glacier:CreateVault

Resources:

Amazon Glacier Condition Keys:

Delete Archive (DELETE archive)

Required Permissions (API Actions):glacier:DeleteArchive

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys: glacier:ArchiveAgeInDays, glacier:ResourceTag/*TagKey*

Delete Vault (DELETE vault)

Required Permissions (API Actions):glacier:DeleteVault

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys: glacier:ResourceTag/*TagKey*

Delete Vault Access Policy (DELETE access-policy)

Required Permissions (API Actions):glacier:DeleteVaultAccessPolicy

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys: glacier:ResourceTag/*TagKey*

Delete Vault Notifications (DELETE notification-configuration)

Required Permissions (API Actions):glacier:DeleteVaultNotifications

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys: glacier:ResourceTag/*TagKey*

Describe Job (GET JobID)

Required Permissions (API Actions):glacier:DescribeJob

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys:

Describe Vault (GET vault)

Required Permissions (API Actions):glacier:DescribeVault

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys:

Get Data Retrieval Policy (GET policy) *

Required Permissions (API Actions):glacier:GetDataRetrievalPolicy

Resources: arn:aws:glacier:*region*:*account-id*:policies/retrieval-limit-policy

Amazon Glacier Condition Keys:

Get Job Output (GET output)

Required Permissions (API Actions):glacier:GetJobOutput

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys:

Get Vault Access Policy (GET access-policy)

Required Permissions (API Actions):glacier:GetVaultAccessPolicy

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys:

Get Vault Lock (GET lock-policy)

Required Permissions (API Actions):glacier:GetVaultLock

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys:

Get Vault Notifications (GET notification-configuration)

Required Permissions (API Actions):glacier:GetVaultNotifications

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys:

Initiate Job (POST jobs)

Required Permissions (API Actions):glacier:InitiateJob

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys: glacier:ArchiveAgeInDays, glacier:ResourceTag/*TagKey*

Initiate Multipart Upload (POST multipart-uploads)

Required Permissions (API Actions):glacier:InitiateMultipartUpload

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys: glacier:ResourceTag/*TagKey*

Initiate Vault Lock (POST lock-policy)

Required Permissions (API Actions):glacier:InitiateVaultLock

Resources:

Amazon Glacier Condition Keys: glacier:ResourceTag/*TagKey*

List Jobs (GET jobs)

Required Permissions (API Actions):glacier:ListJobs

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys:

List Multipart Uploads (GET multipart-uploads)

Required Permissions (API Actions):glacier:ListMultipartUploads

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys:

List Parts (GET uploadID)

Required Permissions (API Actions):glacier:ListParts

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys:

List Tags For Vault (GET tags)

Required Permissions (API Actions):glacier:ListTagsForVault

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys:

List Vaults (GET vaults)

Required Permissions (API Actions):glacier:ListVaults

Resources:

Amazon Glacier Condition Keys:

Remove Tags From Vault (POST tags remove)

Required Permissions (API Actions):glacier:RemoveTagsFromVault

Resources: arn:aws:glacier:*region*:*account-id*:vaults/vault-name, arn:aws:glacier:*region*:*account-id*:vaults/example*, arn:aws:glacier:*region*:*account-id*:vaults/*

Amazon Glacier Condition Keys: glacier:ResourceTag/*TagKey*

Set Data Retrieval Policy (PUT policy) *

Required Permissions (API Actions):glacier:SetDataRetrievalPolicy

Resources:arn:aws:glacier:*region*:*account-id*:policies/retrieval-limit-policy

Amazon Glacier Condition Keys:

Set Vault Access Policy (PUT access-policy)

Required Permissions (API Actions):`glacier:SetVaultAccessPolicy`

Resources: `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/vault-name`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/example*`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/*`

Amazon Glacier Condition Keys: `glacier:ResourceTag/`*`TagKey`*

Set Vault Notification Configuration (PUT notification-configuration)

Required Permissions (API Actions):`glacier:SetVaultNotifications`

Resources: `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/vault-name`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/example*`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/*`

Amazon Glacier Condition Keys: `glacier:ResourceTag/`*`TagKey`*

Upload Archive (POST archive)

Required Permissions (API Actions):`glacier:UploadArchive`

Resources: `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/vault-name`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/example*`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/*`

Amazon Glacier Condition Keys: `glacier:ResourceTag/`*`TagKey`*

Upload Part (PUT uploadID)

Required Permissions (API Actions):`glacier:UploadMultipartPart`

Resources: `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/vault-name`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/example*`, `arn:aws:glacier:`*`region`*`:`*`account-id`*`:vaults/*`

Amazon Glacier Condition Keys: `glacier:ResourceTag/`*`TagKey`*

Logging and Monitoring in Amazon Glacier

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon Glacier (Amazon Glacier) and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily identify and debug the source of a failure if one occurs. AWS provides the following tools for monitoring your Amazon Glacier resources and responding to potential incidents:

Amazon CloudWatch Alarms

When using Amazon Glacier via Amazon S3, you can use Amazon CloudWatch alarms to watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon SNS topic or AWS Auto Scaling policy. CloudWatch alarms do not invoke actions because they are in a particular state. Rather the state must have changed and been maintained for a specified number of periods. For more information, see Monitoring Metrics with Amazon CloudWatch.

AWS CloudTrail Logs

CloudTrail provides a record of actions taken by a user, role, or an AWS service in Amazon Glacier. CloudTrail captures all API calls for Amazon Glacier as events, including calls from the Amazon Glacier console and from code calls to the Amazon Glacier APIs. For more information, see Logging Amazon Glacier API Calls with AWS CloudTrail.

AWS Trusted Advisor

Trusted Advisor draws upon best practices learned from serving hundreds of thousands of AWS customers. Trusted Advisor inspects your AWS environment and then makes recommendations when opportunities exist to save money, improve system availability and performance, or help close security gaps. All AWS customers have access to five Trusted Advisor checks. Customers with a Business or Enterprise support plan can view all Trusted Advisor checks.

For more information, see AWS Trusted Advisor in the *Support User Guide*.

Compliance Validation for Amazon Glacier

The security and compliance of Amazon Glacier (Amazon Glacier) is assessed by third-party auditors as part of multiple AWS compliance programs, including the following:

- System and Organization Controls (SOC)
- Payment Card Industry Data Security Standard (PCI DSS)
- Federal Risk and Authorization Management Program (FedRAMP)
- Health Insurance Portability and Accountability Act (HIPAA)

AWS provides a frequently updated list of AWS services in scope of specific compliance programs at AWS Services in Scope by Compliance Program.

Third-party audit reports are available for you to download using AWS Artifact. For more information, see Downloading Reports in AWS Artifact in the *AWS Artifact User Guide*.

For more information about AWS compliance programs, see AWS Compliance Programs.

Your compliance responsibility when using Amazon Glacier is determined by the sensitivity of your data, your organization's compliance objectives, and applicable laws and regulations. If your use of Amazon Glacier is subject to compliance with standards like HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- Amazon Glacier Vault Lock allows you to easily deploy and enforce compliance controls for individual Amazon Glacier vaults with a vault lock policy. You can specify controls such as "write once read many" (WORM) in a vault lock policy and lock the policy from future edits. After the policy is locked, it can no longer be changed. Vault lock policies can help you comply with regulatory frameworks such as SEC17a-4 and HIPAA.
- Security and Compliance Quick Start Guides discuss architectural considerations and steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance outlines how companies use AWS to help them meet HIPAA requirements.
- The AWS Well-Architected Tool (AWS WA Tool) is a service in the cloud that provides a consistent process for you to review and measure your architecture using AWS best practices. The AWS WA Tool provides recommendations for making your workloads more reliable, secure, efficient, and cost-effective.

- AWS Compliance Resources provide several different workbooks and guides that might apply to your industry and location.
- AWS Config can help you assess how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub provides you with a comprehensive view of your security state within AWS and helps you check your compliance with security industry standards and best practices.

Resilience in Amazon Glacier

The AWS global infrastructure is built around Regions and Availability Zones. AWS Regions provide multiple, physically separated and isolated Availability Zones that are connected with low latency, high throughput, and highly redundant networking. These Availability Zones offer you an effective way to design and operate applications and databases. They are more highly available, fault tolerant, and scalable than traditional single data center infrastructures or multi-data center infrastructures. Amazon Glacier redundantly stores data in multiple devices spanning a minimum of three Availability Zones. To increase durability, Amazon Glacier synchronously stores your data across multiple AZs before confirming a successful upload.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure Security in Amazon Glacier

As a managed service, Amazon Glacier (Amazon Glacier) is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes.

Access to Amazon Glacier via the network is through AWS published APIs. Clients must support Transport Layer Security (TLS) 1.2. We recommend TLS 1.3 or later. Clients must also support cipher suites with Perfect Forward Secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). Most modern systems such as Java 7 and later support these modes. Additionally, requests must be signed using an access key ID and a secret access key that is associated with an IAM principal, or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

VPC Endpoints

A virtual private cloud (VPC) endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Although Amazon Glacier does not support VPC endpoints directly, you can take advantage of Amazon S3 VPC endpoints if you access Amazon Glacier as a storage tier integrated with Amazon S3.

For more information about Amazon S3 lifecycle configuration and transitioning objects to the Amazon Glacier storage class, see Object Lifecycle Management and Transitioning Objects in the *Amazon Simple Storage Service User Guide*. For more information about VPC endpoints, see VPC Endpoints in the *Amazon VPC User Guide*.

Amazon Glacier Data Retrieval Policies

With Amazon Glacier data retrieval policies, you can easily set data retrieval quotas and manage the data retrieval activities across your AWS account in each AWS Region. For more information about Amazon Glacier data retrieval charges, see Amazon Glacier pricing.

> **Important**
>
> A data retrieval policy applies only to Standard retrievals and manages retrieval requests made directly to Amazon Glacier.
> For more information about the Amazon Glacier storage classes, see Storage classes for archiving objects and Transitioning objects in the *Amazon Simple Storage Service User Guide*.

Topics

Choosing an Amazon Glacier Data Retrieval Policy

You can choose from three types of Amazon Glacier data retrieval policies: No Retrieval Limit, Free Tier Only, and Max Retrieval Rate.

No Retrieval Limit is the default data retrieval policy that's used for retrievals. If you use the No Retrieval Limit policy, no retrieval quota is set, and all valid data retrieval requests are accepted.

By using a Free Tier Only policy, you can keep your retrievals within your daily AWS Free Tier allowance and not incur any data retrieval costs. If you want to retrieve more data than is in your AWS Free Tier allowance, you can use a Max Retrieval Rate policy to set a bytes-per-hour retrieval-rate quota. The Max Retrieval Rate policy ensures that the peak retrieval rate from all retrieval jobs across your account in an AWS Region does not exceed the bytes-per-hour quota that you set.

With both the Free Tier Only and Max Retrieval Rate policies, data retrieval requests that exceed the retrieval quotas that you specified are not accepted. If you use a Free Tier Only policy, Amazon

Glacier synchronously rejects retrieval requests that exceed your AWS Free Tier allowance. If you use a Max Retrieval Rate policy, Amazon Glacier rejects retrieval requests that cause the peak retrieval rate of the in-progress jobs to exceed the bytes-per-hour quota set by the policy. These policies help you simplify data retrieval cost management.

The following are some useful facts about data retrieval policies:

- Data retrieval policy settings do not change the 3- to 5-hour period that it takes to retrieve data from Amazon Glacier by using Standard retrievals.
- Setting a new data retrieval policy does not affect previously accepted retrieval jobs that are already in progress.
- If a retrieval job request is rejected because of a data retrieval policy, you are not charged for the job or the request.
- You can set one data retrieval policy for each AWS Region, which will govern all data retrieval activities in the AWS Region under your account. A data retrieval policy is specific to a particular AWS Region because data retrieval costs vary across AWS Regions. For more information, see Amazon Glacier pricing.

Free Tier Only Policy

You can set a data retrieval policy to Free Tier Only to ensure that your retrievals always stay within your AWS Free Tier allowance, so that you don't incur data retrieval charges. If a retrieval request is rejected, you receive an error message stating that the request has been denied by the current data retrieval policy.

You can set the data retrieval policy to Free Tier Only on a per-Region basis. After the policy is set, you cannot retrieve more data in a day than your prorated daily AWS Free Tier retrieval allowance for that AWS Region. You also do not incur data retrieval fees.

You can also switch to a Free Tier Only policy after you have incurred data retrieval charges within a month. In that case, the Free Tier Only policy takes effect for new retrieval requests, but does not affect past requests. You will be billed for the previously incurred charges.

Max Retrieval Rate Policy

You can set your data retrieval policy to Max Retrieval Rate to control the peak retrieval rate by specifying a data retrieval quota that has a bytes-per-hour maximum. When you set the data retrieval policy to Max Retrieval Rate, a new retrieval request is rejected if it would cause the peak

retrieval rate of the in-progress jobs to exceed the bytes-per-hour quota that's specified by the policy. If a retrieval job request is rejected, you receive an error message stating that the request has been denied by the current data retrieval policy.

Setting your data retrieval policy to the Max Retrieval Rate policy can affect how much of your AWS Free Tier allowance that you can use in a day. For example, suppose you set Max Retrieval Rate to 1 MB per hour. This is less than the AWS Free Tier policy rate. To ensure that you make good use of the daily AWS Free Tier allowance, you can first set your policy to Free Tier Only, and then switch to the Max Retrieval Rate policy later if you need to. For more information about how your retrieval allowance is calculated, go to Amazon Glacier FAQs.

No Retrieval Limit Policy

If your data retrieval policy is set to No Retrieval Limit, all valid data retrieval requests are accepted and your data retrieval costs will vary based on your usage.

Using the Amazon Glacier Console to Set Up a Data Retrieval Policy

To create a data retrieval policy by using the Amazon Glacier console

1. Sign in to the AWS Management Console and open the Amazon Glacier console at https://console.aws.amazon.com/glacier/home.
2. Under **Select a Region**, choose an AWS Region from the dropdown menu. You can configure a data retrieval policy for each AWS Region.
3. In the left navigation pane, choose **Data retrieval settings**.
4. Choose **Edit**. The **Edit data retrieval policies** page appears.
5. Under **Data retrieval policies**, choose a policy.

   You can select one of the three data retrieval policies: **No retrieval limit**, **Free Tier only**, or **Specify a max retrieval rate**.

   - If you choose **No retrieval limit**, all valid data retrieval requests are accepted.
   - If you choose **Free Tier only**, data retrieval requests that exceed the AWS Free Tier are not accepted.
   - If you choose **Specify a max retrieval rate**, data retrieval requests are rejected if they would cause the peak retrieval rate of the in-progress jobs to exceed the max retrieval rate that

you specify. You must specify a gigabytes (GB) per hour value in the **GB/hour** box under **Max retrieval rate**. When you enter a value for **GB/hour**, the console calculates an estimated cost for you.

6. Choose **Save changes**.

Using the Amazon Glacier API to Set Up a Data Retrieval Policy

You can view and set a data retrieval policy by using the Amazon Glacier REST API or by using the AWS SDKs.

Using the Amazon Glacier REST API to Set Up a Data Retrieval Policy

You can view and set a data retrieval policy by using the Amazon Glacier REST API. You can view an existing data retrieval policy by using the Get Data Retrieval Policy (GET policy) operation. You set a data retrieval policy by using the Set Data Retrieval Policy (PUT policy) operation.

When using the PUT policy operation, you select the data retrieval policy type by setting the JSON `Strategy` field value to `BytesPerHour`, `FreeTier`, or None. `BytesPerHour` is equivalent to choosing **Specify a max retrieval rate** in the console, `FreeTier` to choosing **Free Tier only**, and None to choosing **No retrieval limit**.

When you use the Initiate Job (POST jobs) operation to initiate a data retrieval job that will exceed the maximum retrieval rate set in your data retrieval policy, the `Initiate Job` operation stops and throws an exception.

Using the AWS SDKs to Set Up a Data Retrieval Policy

AWS provides SDKs for you to develop applications for Amazon Glacier. These SDKs provide libraries that map to the underlying REST API and provide objects that enable you to easily construct requests and process responses. For more information, see Using the AWS SDKs with Amazon Glacier.

Tagging Amazon Glacier Resources

A *tag* is a label that you assign to an AWS resource. Each tag consists of a *key* and a *value*, both of which you define. You can assign the tags that you define to Amazon Glacier (Amazon Glacier) vault resources. Using tags is a simple yet powerful way to manage AWS resources and organize data, including billing data.

Topics

Tagging Basics

You use the Amazon Glacier console, AWS Command Line Interface (AWS CLI), or Amazon Glacier API to complete the following tasks:

- Adding tags to a vault
- Listing the tags for a vault
- Removing tags from a vault

For information about how to add, list, and remove tags, see Tagging Your Amazon Glacier Vaults.

You can use tags to categorize your vaults. For example, you can categorize vaults by purpose, owner, or environment. Because you define the key and value for each tag, you can create a custom set of categories to meet your specific needs. For example, you might define a set of tags that helps you track vaults by owner and purpose for the vault. Following are a few examples of tags:

- Owner: Name
- Purpose: Video archives
- Environment: Production

Tag Restrictions

Basic tag restrictions are as follows:

- The maximum number of tags for a resource (vault) is 50.
- Tag keys and values are case-sensitive.

Tag key restrictions are as follows:

- Within a set of tags for a vault, each tag key must be unique. If you add a tag with a key that's already in use, your new tag overwrites the existing key-value pair.
- Tag keys cannot start with `aws:` because this prefix is reserved for use by AWS. AWS can create tags that begin with this prefix on your behalf, but you can't edit or delete them.
- Tag keys must be from 1 to 128 Unicode characters in length.
- Tag keys must consist of the following characters: Unicode letters, digits, spaces, and the following special characters: `_ . / = + - @`.

Tag value restrictions are as follows:

- Tag values must be from 0 to 255 Unicode characters in length.
- Tag values can be blank. Otherwise, they must consist of the following characters: Unicode letters, digits, spaces, and any of the following special characters: `_ . / = + - @`.

Tracking Costs Using Tagging

You can use tags to categorize and track your AWS costs. When you apply tags to any AWS resources, including vaults, your AWS cost allocation report includes usage and costs aggregated by tags. You can apply tags that represent business categories (such as cost centers, application names, and owners) to organize your costs across multiple services. For more information, see Use Cost Allocation Tags for Custom Billing Reports in the *AWS Billing User Guide*.

Managing Access Control with Tagging

You can use tags as a condition in an access policy statement. For example, you can set up a legal hold tag and include it as a condition in a data retention policy that states that “archive deletion

from everyone will be denied if the legal hold tag value is set to `True`." You can deploy the data retention policy and set the legal hold tag to `False` under normal conditions. If your data must be put on hold to assist an investigation, you can easily turn on the legal hold by setting the tag value to `True` and removing the hold in a similar way later on. For more information, see Controlling Access Using Tags in the *IAM User Guide*.

Related Sections

- Tagging Your Amazon Glacier Vaults

Logging Amazon Glacier API Calls with AWS CloudTrail

Amazon Glacier (Amazon Glacier) is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon Glacier. CloudTrail captures all API calls for Amazon Glacier as events, including calls from the Amazon Glacier console and from code calls to the Amazon Glacier APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon Glacier. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon Glacier, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Amazon Glacier Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon Glacier, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon Glacier, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all AWS Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All Amazon Glacier actions are logged by CloudTrail and are documented in the API Reference for Amazon Glacier. For example, calls to the Create Vault (PUT vault), Delete Vault (DELETE vault), and List Vaults (GET vaults) actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or other credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Understanding Amazon Glacier Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the Create Vault (PUT vault), Delete Vault (DELETE vault), List Vaults (GET vaults), and Describe Vault (GET vault) actions.


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

API Reference for Amazon Glacier

Amazon Glacier supports a set of operations—specifically, a set of RESTful API calls—that enable you to interact with the service.

You can use any programming library that can send HTTP requests to send your REST requests to Amazon Glacier. When sending a REST request, Amazon Glacier requires that you authenticate every request by signing the request. Additionally, when uploading an archive, you must also compute the checksum of the payload and include it in your request. For more information, see Signing Requests.

If an error occurs, you need to know what Amazon Glacier sends in an error response so that you can process it. This section provides all this information, in addition to documenting the REST operations, so that you can make REST API calls directly.

You can either use the REST API calls directly or use the Amazon SDKs that provide wrapper libraries. These libraries sign each request you send and compute the checksum of the payload in your request. Therefore, using the Amazon SDKs simplifies your coding task. This developer guide provides working examples of basic Amazon Glacier operations using the AWS SDK for Java and .NET. For more information see, Using the AWS SDKs with Amazon Glacier.

Topics

Common Request Headers

Amazon Glacier (Amazon Glacier) REST requests include headers that contain basic information about the request. The following table describes headers that can be used by all Amazon Glacier REST requests.

Header Name	Description	Required
`Authorization`	The header that is required to sign requests. Amazon Glacier requires Signature Version 4. For more information, see Signing Requests. Type: String	Yes
`Content-Length`	The length of the request body (without the headers). Type: String Condition: Required only for the Upload Archive (POST archive) API.	Conditional
`Date`	The date that can be used to create the signature contained in the `Authorization` header. If the `Date` header is to be used for signing it must be specified in the ISO 8601 basic format. In this case, the `x-amz-date` header is not needed. Note that when `x-amz-date` is present, it always overrides the value of the `Date` header. If the Date header is not used for signing, it can be one of the full date formats specified by RFC 2616, section 3.3. For example, the following date/time `Wed, 10 Feb 2017 12:00:00 GMT` is a valid date/time header for use with Amazon Glacier.	Conditional

Header Name	Description	Required
	If you are using the `Date` header for signing, then it must be in the ISO 8601 basic `YYYYMMDD'T'HHMMSS'Z'` format. Type: String Condition: If `Date` is specified but is not in ISO 8601 basic format, then you must also include the `x-amz-date` header. If `Date` is specified in ISO 8601 basic format, then this is sufficient for signing requests and you do not need the `x-amz-date` header. For more information, see Handling Dates in Signature Version 4 in the *Amazon Web Services Glossary*.	
`Host`	This header specifies the service endpoint to which you send your requests. The value must be of the form `"glacier.`*`region`*`.amazonaws.com "`, where *`region`* is replaced with an AWS Region designation such as `us-west-2 .` Type: String	Yes

Header Name	Description	Required
x-amz-content-sha256	The computed SHA256 checksum of an entire payload that is uploaded with either Upload Archive (POST archive) or Upload Part (PUT uploadID). This header is not the same as the x-amz-sha256-tree-hash header, though, for some small payloads the values are the same. When x-amz-content-sha256 is required, both x-amz-content-sha256 and x-amz-sha256-tree-hash must be specified. Type: String Condition: Required for streaming API, Upload Archive (POST archive) and Upload Part (PUT uploadID).	Conditional
x-amz-date	The date used to create the signature in the Authorization header. The format must be ISO 8601 basic in the YYYYMMDD'T'HHMMSS'Z' format. For example, the following date/time 20170210T120000Z is a valid x-amz-date for use with Amazon Glacier. Type: String Condition: x-amz-date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, then x-amz-date is not needed. When x-amz-date is present, it always overrides the value of the Date header. For more information, see Handling Dates in Signature Version 4 in the *Amazon Web Services Glossary*.	Conditional

Header Name	Description	Required
`x-amz-glacier-version`	The Amazon Glacier API version to use. The current version is `2012-06-01` . Type: String	Yes
`x-amz-sha256-tree-hash`	The computed SHA256 tree-hash checksum for an uploaded archive (Upload Archive (POST archive)) or archive part (Upload Part (PUT uploadID)). For more information about calculating this checksum, see Computing Checksums. Type: String Default: None Condition: Required for Upload Archive (POST archive) and Upload Part (PUT uploadID).	Conditional

Common Response Headers

The following table describes response headers that are common to most API responses.

Name	Description
`Content-Length`	The length in bytes of the response body. Type: String
`Date`	The date and time Amazon Glacier (Amazon Glacier) responded, for example, `Wed, 10 Feb 2017 12:00:00 GMT`. The format of the date must be one of the full date formats specified by RFC 2616, section 3.3. Note that `Date` returned may drift slightly from other dates, so for example, the date returned from an Upload Archive (POST archive) request may not match the date shown for the archive in an inventory list for the vault.

Name	Description
	Type: String
`x-amzn-RequestId`	A value created by Amazon Glacier that uniquely identifies your request. In the event that you have a problem with Amazon Glacier, AWS can use this value to troubleshoot the problem. It is recommended that you log these values. Type: String
`x-amz-sha256-tree-hash`	The SHA256 tree-hash checksum of the archive or inventory body. For more information about calculating this checksum, see Computing Checksums. Type: String

Signing Requests

Amazon Glacier requires that you authenticate every request you send by signing the request. To sign a request, you calculate a digital signature using a cryptographic hash function. A cryptographic hash is a function that returns a unique hash value based on the input. The input to the hash function includes the text of your request and your secret access key. The hash function returns a hash value that you include in the request as your signature. The signature is part of the `Authorization` header of your request.

After receiving your request, Amazon Glacier recalculates the signature using the same hash function and input that you used to sign the request. If the resulting signature matches the signature in the request, Amazon Glacier processes the request. Otherwise, the request is rejected.

Amazon Glacier supports authentication using AWS Signature Version 4. The process for calculating a signature can be broken into three tasks:

- Task 1: Create a Canonical Request

  Rearrange your HTTP request into a canonical format. Using a canonical form is necessary because Amazon Glacier uses the same canonical form when it recalculates a signature to compare with the one you sent.

- Task 2: Create a String to Sign

  Create a string that you will use as one of the input values to your cryptographic hash function. The string, called the *string to sign*, is a concatenation of the name of the hash algorithm, the request date, a *credential scope* string, and the canonicalized request from the previous task. The *credential scope* string itself is a concatenation of date, AWS Region, and service information.

- Task 3: Create a Signature

  Create a signature for your request by using a cryptographic hash function that accepts two input strings: your *string to sign* and a *derived key*. The *derived key* is calculated by starting with your secret access key and using the *credential scope* string to create a series of hash-based message authentication codes (HMACs). Note that the hash function used in this signing step is not the tree-hash algorithm used in Amazon Glacier APIs that upload data.

Topics

Example Signature Calculation

The following example walks you through the details of creating a signature for Create Vault (PUT vault). The example could be used as a reference to check your signature calculation method. For more information, see Signing AWS API requests in the *IAM User Guide*.

The example assumes the following:

- The time stamp of the request is `Fri, 25 May 2012 00:24:53 GMT`.
- The endpoint is US East (N. Virginia) Region `us-east-1`.

The general request syntax (including the JSON body) is:

```
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
```

The canonical form of the request calculated for Task 1: Create a Canonical Request is:

```
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
```

The last line of the canonical request is the hash of the request body. Also, note the empty third line in the canonical request. This is because there are no query parameters for this API.

The *string to sign* for Task 2: Create a String to Sign is:

```
AWS4-HMAC-SHA256
20120525T002453Z
20120525/us-east-1/glacier/aws4_request
5f1da1a2d0feb614dd03d71e87928b8e449ac87614479332aced3a701f916743
```

The first line of the *string to sign* is the algorithm, the second line is the time stamp, the third line is the *credential scope*, and the last line is a hash of the canonical request from Task 1: Create a Canonical Request. The service name to use in the credential scope is `glacier`.

For Task 3: Create a Signature, the *derived key* can be represented as:

```
derived key = HMAC(HMAC(HMAC(HMAC("AWS4" + YourSecretAccessKey,"20120525"),"us-
east-1"),"glacier"),"aws4_request")
```

If the secret access key, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`, is used, then the calculated signature is:

```
3ce5b2f2fffac9262b4da9256f8d086b4aaf42eba5f111c21681a65a127b7c2a
```

The final step is to construct the `Authorization` header. For the demonstration access key `AKIAIOSFODNN7EXAMPLE`, the header (with line breaks added for readability) is:

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-east-1/
glacier/aws4_request,
SignedHeaders=host;x-amz-date;x-amz-glacier-version,
Signature=3ce5b2f2fffac9262b4da9256f8d086b4aaf42eba5f111c21681a65a127b7c2a
```

Calculating Signatures for the Streaming Operations

Upload Archive (POST archive) and Upload Part (PUT uploadID) are streaming operations that require you to include an additional header `x-amz-content-sha256` when signing and sending your request. The signing steps for the streaming operations are exactly the same as those for other operations, with the addition of the streaming header.

The calculation of the streaming header `x-amz-content-sha256` is based on the SHA256 hash of the entire content (payload) that is to be uploaded. Note that this calculation is different from the SHA256 tree hash (Computing Checksums). Besides trivial cases, the SHA 256 hash value of the payload data will be different from the SHA256 tree hash of the payload data.

If the payload data is specified as a byte array, you can use the following Java code snippet to calculate the SHA256 hash.

```
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();
}
```

Similarly, in C# you can calculate the SHA256 hash of the payload data as shown in the following code snippet.

```
public static byte[] CalculateSHA256Hash(byte[] payload)
{
    SHA256 sha256 = System.Security.Cryptography.SHA256.Create();
    byte[] hash = sha256.ComputeHash(payload);

    return hash;
}
```

Example Signature Calculation for Streaming API

The following example walks you through the details of creating a signature for Upload Archive (POST archive), one of the two streaming APIs in Amazon Glacier. The example assumes the following:

- The time stamp of the request is `Mon, 07 May 2012 00:00:00 GMT`.
- The endpoint is the US East (N. Virginia) Region, us-east-1.
- The content payload is a string "Welcome to Amazon Glacier."

The general request syntax (including the JSON body) is shown in the example below. Note that the `x-amz-content-sha256` header is included. In this simplified example, the `x-amz-sha256-tree-hash` and `x-amz-content-sha256` are the same value. However, for archive uploads greater than 1 MB, this is not the case.

```
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
```

The canonical form of the request calculated for Task 1: Create a Canonical Request is shown below. Note that the streaming header `x-amz-content-sha256` is included with its value. This means you must read the payload and calculate the SHA256 hash first and then compute the signature.

```
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
```

The remainder of the signature calculation follows the steps outlined in Example Signature Calculation. The `Authorization` header using the secret access key `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY` and the access key `AKIAIOSFODNN7EXAMPLE` is shown below (with line breaks added for readability):

```
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
```

Computing Checksums

When uploading an archive, you must include both the `x-amz-sha256-tree-hash` and `x-amz-content-sha256` headers. The `x-amz-sha256-tree-hash` header is a checksum of the payload in your request body. This topic describes how to calculate the `x-amz-sha256-tree-hash` header. The `x-amz-content-sha256` header is a hash of the entire payload and is required for authorization. For more information, see Example Signature Calculation for Streaming API.

The payload of your request can be an:

- **Entire archive—** When uploading an archive in a single request using the Upload Archive API, you send the entire archive in the request body. In this case, you must include the checksum of the entire archive.
- **Archive part—** When uploading an archive in parts using the multipart upload API, you send only a part of the archive in the request body. In this case, you include the checksum of the archive part. And after you upload all the parts, you send a Complete Multipart Upload request, which must include the checksum of the entire archive.

The checksum of the payload is a SHA-256 tree hash. It is called a tree hash because in the process of computing the checksum you compute a tree of SHA-256 hash values. The hash value at the root is the checksum for the entire archive.

> **Note**
>
> This section describes a way to compute the SHA-256 tree hash. However, you may use any procedure as long as it produces the same result.

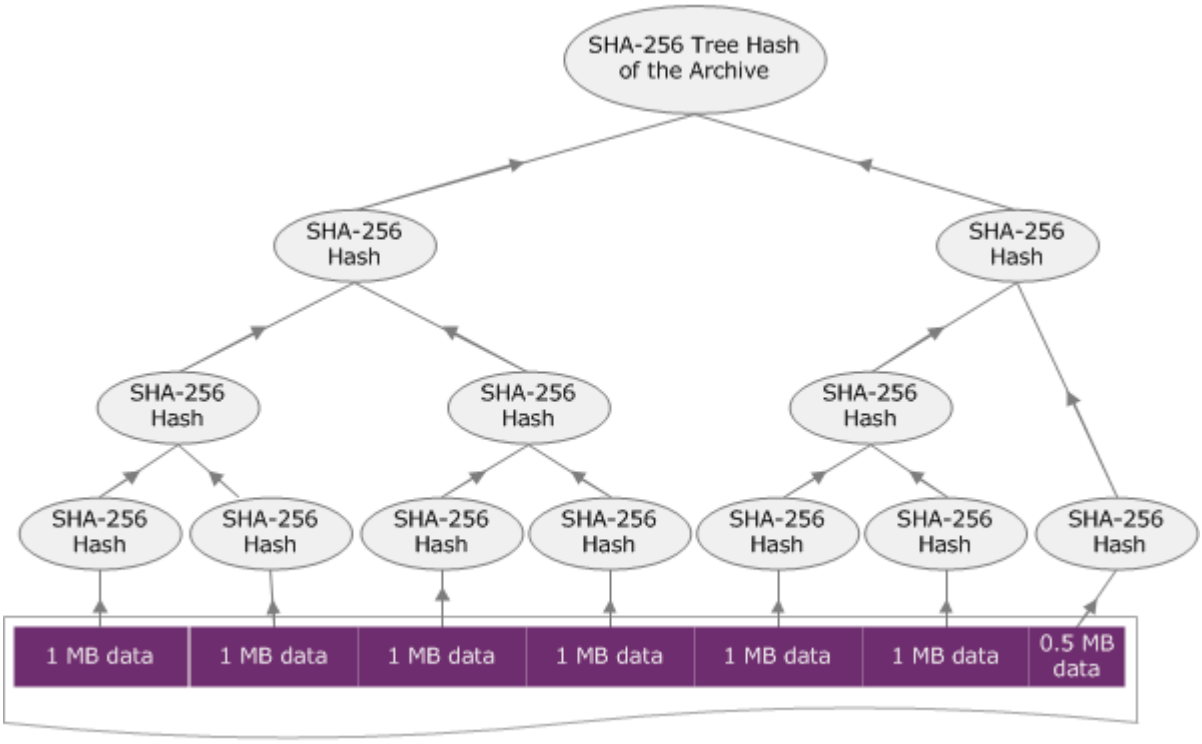
You compute the SHA-256 tree hash as follows:

1. For each 1 MB chunk of payload data, compute the SHA-256 hash. The last chunk of data can be less than 1 MB. For example, if you are uploading a 3.2 MB archive, you compute the SHA-256 hash values for each of the first three 1 MB chunks of data, and then compute the SHA-256 hash of the remaining 0.2 MB data. These hash values form the leaf nodes of the tree.
2. Build the next level of the tree.
   a. Concatenate two consecutive child node hash values and compute the SHA-256 hash of the concatenated hash values. This concatenation and generation of the SHA-256 hash produces a parent node for the two child nodes.
   b. If only one child node remains, promote that hash value to the next level in the tree.
3. Repeat step 2 until the resulting tree has a root. The root of the tree provides a hash of the entire archive and a root of the appropriate subtree provides the hash for the part in a multipart upload.

Topics

Tree Hash Example 1: Uploading an archive in a single request

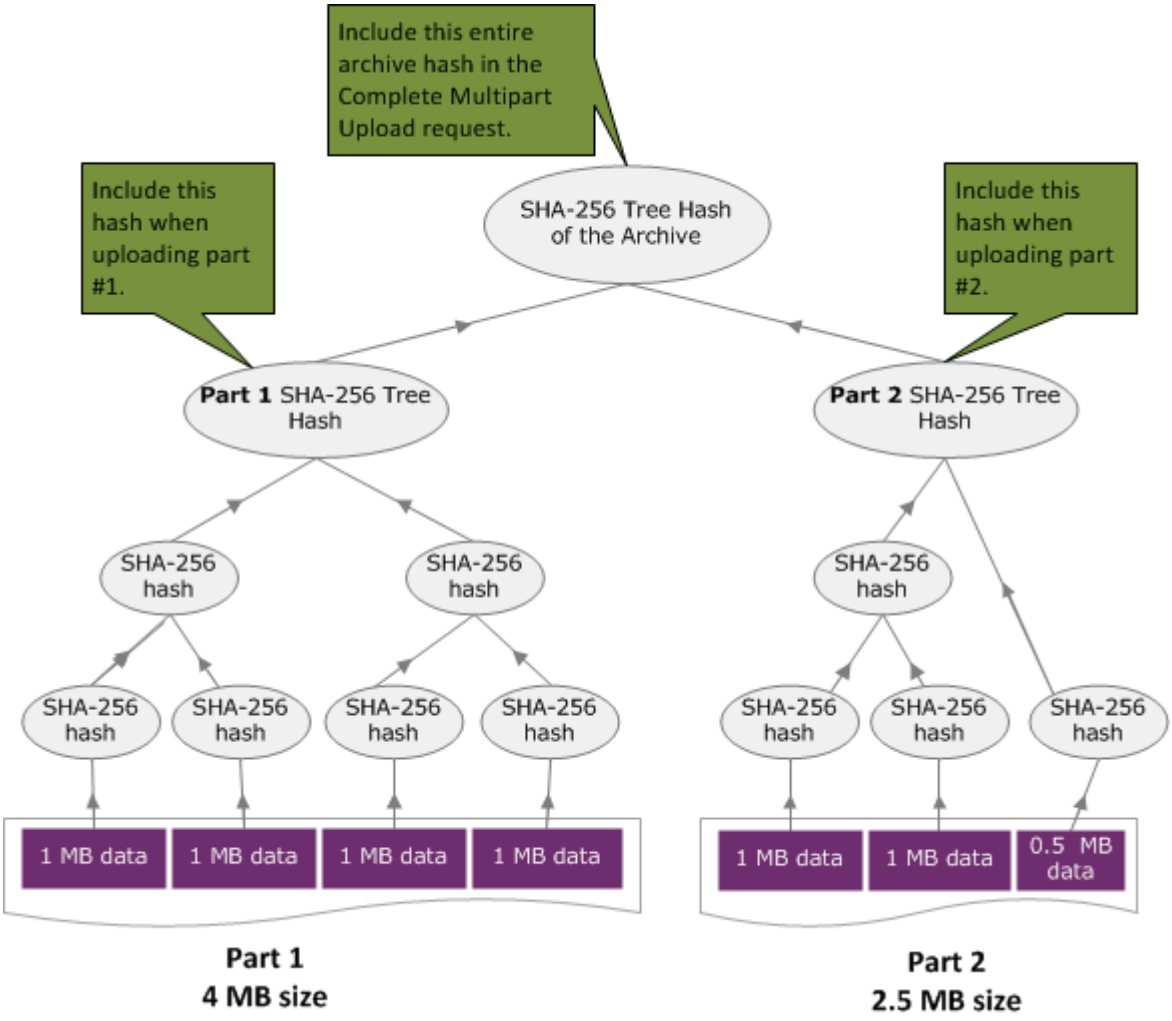
When you upload an archive in a single request using the Upload Archive API (see Upload Archive (POST archive)), the request payload includes the entire archive. Accordingly, you must include the tree hash of the entire archive in the `x-amz-sha256-tree-hash` request header. Suppose you want to upload a 6.5 MB archive. The following diagram illustrates the process of creating the SHA-256 hash of the archive. You read the archive and compute the SHA-256 hash for each 1 MB chunk. You also compute the hash for the remaining 0.5 MB data and then build the tree as outlined in the preceding procedure.



6.5 MB Archive

Tree Hash Example 2: Uploading an archive using a multipart upload

The process of computing the tree hash when uploading an archive using multipart upload is the same when uploading the archive in a single request. The only difference is that in a multipart upload you upload only a part of the archive in each request (using the Upload Part (PUT uploadID) API), and therefore you provide the checksum of only the part in the `x-amz-sha256-tree-hash` request header. However, after you upload all parts, you must send the Complete Multipart Upload (see Complete Multipart Upload (POST uploadID)) request with a tree hash of the entire archive in the `x-amz-sha256-tree-hash` request header.



Computing the Tree Hash of a File

The algorithms shown here are selected for demonstration purposes. You can optimize the code as needed for your implementation scenario. If you are using an Amazon SDK to program against Amazon Glacier (Amazon Glacier), the tree hash calculation is done for you and you only need to provide the file reference.

Example 1: Java Example

The following example shows how to calculate the SHA256 tree hash of a file using Java. You can run this example by either supplying a file location as an argument or you can use the `TreeHashExample.computeSHA256TreeHash` method directly from your code.

```
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 Example

The following example shows how to calculate the SHA256 tree hash of a file. You can run this example by supplying a file location as an argument.

```
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;
        }
    }
}
```

Receiving Checksums When Downloading Data

When you retrieve an archive using the Initiate Job API (see Initiate Job (POST jobs)), you can optionally specify a range to retrieve of the archive. Similarly, when you download your data using the Get Job Output API (see Get Job Output (GET output)), you can optionally specify a range of data to download. There are two characteristics of these ranges that are important to understand when you are retrieving and downloading your archive's data. The range to retrieve is required to be *megabyte aligned* to the archive. Both the range to retrieve and the range to download must be *tree hash aligned* in order to receive checksum values when you download your data. The definition of these two types of range alignments are as follows:

- Megabyte aligned - A range [*StartByte*, *EndBytes*] is megabyte (1024*1024) aligned when *StartBytes* is divisible by 1 MB and *EndBytes* plus 1 is divisible by 1 MB or is equal to the end of the archive specified (archive byte size minus 1). A range used in the Initiate Job API, if specified, is required to be megabyte aligned.
- Tree-hash aligned - A range [*StartBytes*, *EndBytes*] is tree hash aligned with respect to an archive if and only if the root of the tree hash built over the range is equivalent to a node in the tree hash of the whole archive. Both the range to retrieve and range to download must be tree hash aligned in order to receive checksum values for the data you download. For an example of ranges and their relationship to the archive tree hash, see Tree Hash Example: Retrieving an archive range that is tree-hash aligned.

  Note that a range that is tree-hash aligned is also megabyte aligned. However, a megabyte aligned range is not necessarily tree-hash aligned.

The following cases describe when you receive a checksum value when you download your archive data:

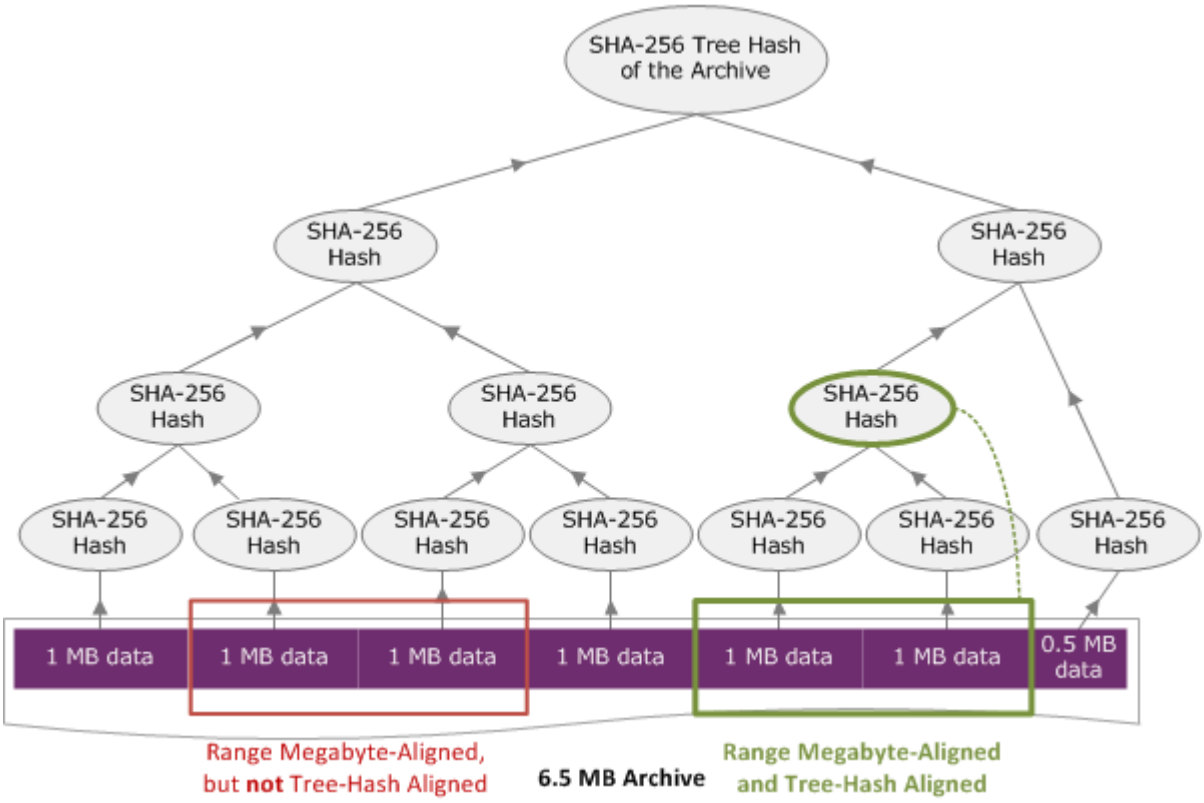
- If you do not specify a range to retrieve in the Initiate Job request and you download the whole archive in the Get Job Request.
- If you do not specify a range to retrieve in the Initiate Job request and you do specify a tree-hash aligned range to download in the Get Job Request.
- If you specify a tree-hash aligned range to retrieve in the Initiate Job request and you download the whole range in the Get Job Request.

- If you specify a tree-hash aligned range to retrieve in the Initiate Job request and you specify a tree-hash aligned range to download in the Get Job Request.

If you specify a range to retrieve in the Initiate Job request that is not tree hash aligned, then you can still get your archive data but no checksum values are returned when you download data in the Get Job Request.

Tree Hash Example: Retrieving an archive range that is tree-hash aligned

Suppose you have a 6.5 MB archive in your vault and you want to retrieve 2 MB of the archive. How you specify the 2 MB range in the Initiate Job request determines if you receive data checksum values when you download your data. The following diagram illustrates two 2 MB ranges for the 6.5 MB archive that you could download. Both ranges are megabyte aligned, but only one is tree-hash aligned.



Tree-Hash Aligned Range Specification

This section gives the exact specification for what constitutes a tree-hash aligned range. Tree-hash aligned ranges are important when you are downloading a portion of an archive and you specify

the range of data to retrieve and the range to download from the retrieved data. If both of these ranges are tree-hash aligned, then you will receive checksum data when you download the data.

A range [*A*, *B*] is *tree-hash aligned* with respect to an archive if and only if when a new tree hash is built over [*A*, *B*], the root of the tree hash of that range is equivalent to a node in the tree hash of the whole archive. You can see this shown in the diagram in Tree Hash Example: Retrieving an archive range that is tree-hash aligned. In this section, we provide the specification for tree-hash alignment.

Consider [*P*, *Q*) as the range query for an archive of *N* megabytes (MB) and *P* and *Q* are multiples of one MB. Note that the actual inclusive range is [*P* MB, *Q* MB – 1 byte], but for simplicity, we show it as [*P*, *Q*). With these considerations, then

- If *P* is an odd number, there is only one possible tree-hash aligned range—that is [*P*, *P* + 1 MB).
- If *P* is an even number and *k* is the maximum number, where *P* can be written as $2k * X$, then there are at most *k* tree-hash aligned ranges that start with *P*. *X* is an integer greater than 0. The tree-hash aligned ranges fall in the following categories:
  - For each *i*, where (0 <= *i* <= *k*) and where $P + 2^i < N$, then [*P*, $Q + 2^i$) is a tree-hash aligned range.
  - *P* = 0 is the special case where *A* = 2[lgN]*0

Error Responses

In the event of an error, the API returns one of the following exceptions:

Code	Description	HTTP Status Code	Type
`AccessDeniedException`	Returned if there was an attempt to access a resource not allowed by an AWS Identity and Access Management (IAM) policy, or the incorrect AWS account ID was used in the request URI. For more information, see Identity and	`403 Forbidden`	Client

Code	Description	HTTP Status Code	Type
	Access Management for Amazon Glacier.		
BadRequest	Returned if the request cannot be processed.	400 Bad Request	Client
ExpiredTokenException	Returned if the security token used in the request has expired.	403 Forbidden	Client
InsufficientCapacityException	Returned if there is insufficient capacity to process the expedited request. This error only applies to expedited retrievals and not to standard or bulk retrievals.	503 Service Unavailable	Server
InvalidParameterValueException	Returned if a parameter of the request is incorrectly specified.	400 Bad Request	Client
InvalidSignatureException	Returned if the request signature is invalid.	403 Forbidden	Client
LimitExceededException	Returned if the request results in one of the following limits being exceeded, a vault limit, a tags limit, or the provisioned capacity limit.	400 Bad Request	Client
MissingAuthenticationTokenException	Returned if no authentication data is found for the request.	400 Bad Request	Client
MissingParameterValueException	Returned if a required header or parameter is missing from the request.	400 Bad Request	Client

Code	Description	HTTP Status Code	Type
PolicyEnforcedException	Returned if a retrieval job will exceed the current data policy's retrieval rate limit. For more information about data retrieval policies, see Amazon Glacier Data Retrieval Policies.	400 Bad Request	Client
ResourceNotFoundException	Returned if the specified resource such as a vault, upload ID, or job ID does not exist.	404 Not Found	Client
RequestTimeoutException	Returned if uploading an archive and Amazon Glacier (Amazon Glacier) times out while receiving the upload.	408 Request Timeout	Client
SerializationException	Returned if the body of the request is invalid. If including a JSON payload, check that it is well-formed.	400 Bad Request	Client
ServiceUnavailableException	Returned if the service cannot complete the request.	500 Internal Server Error	Server
ThrottlingException	Returned if you need to reduce your rate of requests to Amazon Glacier.	400 Bad Request	Client
UnrecognizedClientException	Returned if the Access Key ID or security token is invalid.	400 Bad Request	Client

Various Amazon Glacier APIs return the same exception, but with different exception messages to help you troubleshoot the specific error encountered.

Amazon Glacier returns error information in the response body. The following examples show some of the error responses.

Example 1: Describe Job request with a job ID that does not exist

Suppose you send a Describe Job (GET JobID) request for a job that does not exist. That is, you specify a job ID that does not exist.

```
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
```

In response, Amazon Glacier returns the following error response.

```
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"
  }
```

Where:

Code

One of the general exceptions.

Type: String

Message

A generic description of the error condition specific to the API that returns the error.

Type: String

Type

The source of the error. The field can be one of the following values: `Client`, `Server`, or Unknown.

Type: String.

Note the following in the preceding response:

- For the error response, Amazon Glacier returns status code values of `4xx` and `5xx`. In this example, the status code is `404 Not Found`.
- The `Content-Type` header value `application/json` indicates JSON in the body
- The JSON in the body provides the error information.

In the previous request, instead of a bad job ID, suppose you specify a vault that does not exist. The response returns a different message.

```
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"
}
```

Example 2: List Jobs request with an invalid value for the request parameter

In this example you send a List Jobs (GET jobs) request to retrieve vault jobs with a specific `statuscode`, and you provide an incorrect `statuscode` value `finished`, instead of the acceptable values `InProgress`, `Succeeded`, or `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
```

Amazon Glacier returns the `InvalidParameterValueException` with an appropriate message.

```
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"
}
```

Vault Operations

The following are the vault operations available in Amazon Glacier.

Topics

- Initiate Vault Lock (POST lock-policy)
- List Tags For Vault (GET tags)
- List Vaults (GET vaults)
- Remove Tags From Vault (POST tags remove)
- Set Vault Access Policy (PUT access-policy)
- Set Vault Notification Configuration (PUT notification-configuration)

Abort Vault Lock (DELETE lock-policy)

Description

This operation stops the vault locking process if the vault lock is not in the `Locked` state. If the vault lock is in the `Locked` state when this operation is requested, the operation returns an `AccessDeniedException` error. Stopping the vault locking process removes the vault lock policy from the specified vault.

A vault lock is put into the `InProgress` state by calling Initiate Vault Lock (POST lock-policy). A vault lock is put into the `Locked` state by calling Complete Vault Lock (POST lockId). You can get the state of a vault lock by calling Get Vault Lock (GET lock-policy). For more information about the vault locking process, see Amazon Glacier Vault Lock. For more information about vault lock policies, see Vault Lock Policies.

This operation is idempotent. You can successfully invoke this operation multiple times, if the vault lock is in the `InProgress` state or if there is no policy associated with the vault.

Requests

To delete the vault lock policy, send an HTTP `DELETE` request to the URI of the vault's `lock-policy` subresource.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

If the policy is successfully deleted, Amazon Glacier returns an `HTTP 204 No Content` response.

Syntax

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example demonstrates how to stop the vault locking process.

Example Request

In this example, a DELETE request is sent to the `lock-policy` subresource of the vault named **`examplevault`**.

```
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
```

Example Response

If the policy is successfully deleted Amazon Glacier returns an `HTTP 204 No Content` response, as shown in the following example.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

Related Sections

- Complete Vault Lock (POST lockId)
- Get Vault Lock (GET lock-policy)
- Initiate Vault Lock (POST lock-policy)

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Add Tags To Vault (POST tags add)

This operation adds the specified tags to a vault. Each tag is composed of a key and a value. Each vault can have up to 50 tags. If your request would cause the tag limit for the vault to be exceeded, the operation throws the `LimitExceededException` error.

If a tag already exists on the vault under a specified key, the existing key value will be overwritten. For more information about tags, see Tagging Amazon Glacier Resources.

Request Syntax

To add tags to a vault, send an HTTP POST request to the tags URI as shown in the following syntax example.

```
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

The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS

account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

Request Parameters

Name	Description	Required
`operation=add`	A single query string parameter `operation` with a value of `add` to distinguish it from Remove Tags From Vault (POST tags remove).	Yes

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

The request body contains the following JSON fields.

Tags

The tags to add to the vault. Each tag is composed of a key and a value. The value can be an empty string.

Type: String to String map

Length constraints: Minimum length of 1. Maximum length 10.

Required: Yes

Responses

If the operation request is successful, the service returns an HTTP `204 No Content` response.

Syntax

```
HTTP/1.1 204 No Content
```

```
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example Request

The following example sends an HTTP POST request with the tags to add to the vault.

```
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"
    }
}
```

Example Response

If the request was successful Amazon Glacier returns a `HTTP 204 No Content` as shown in the following example.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

Related Sections

- List Tags For Vault (GET tags)
- Remove Tags From Vault (POST tags remove)

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Create Vault (PUT vault)

Description

This operation creates a new vault with the specified name. The name of the vault must be unique within an AWS Region for an AWS account. You can create up to 1,000 vaults per account. For information on creating more vaults, go to the Amazon Glacier product detail page.

You must use the following guidelines when naming a vault.

- Names can be between 1 and 255 characters long.
- Allowed characters are a–z, A–Z, 0–9, '_' (underscore), '-' (hyphen), and '.' (period).

This operation is idempotent, you can send the same request multiple times and it has no further effect after the first time Amazon Glacier (Amazon Glacier) creates the specified vault.

Requests

Syntax

To create a vault, send an HTTP PUT request to the URI of the vault to be created.

```
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**
>
> The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

The request body for this operation must be empty (0 bytes).

Responses

Syntax

```
HTTP/1.1 201 Created
```

```
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
```

Response Headers

A successful response includes the following response headers, in addition to the response headers that are common to all operations. For more information about common response headers, see Common Response Headers.

Name	Description
Location	The relative URI path of the vault that was created. Type: String

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example Request

The following example sends an HTTP PUT request to create a vault named examplevault.

```
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
```

Example Response

Amazon Glacier creates the vault and returns the relative URI path of the vault in the `Location` header. The account ID is always displayed in the `Location` header regardless of whether the account ID or a hyphen ('-') was specified in the request.

```
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
```

Related Sections

- List Vaults (GET vaults)
- Delete Vault (DELETE vault)
- Identity and Access Management for Amazon Glacier

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Complete Vault Lock (POST lockId)

Description

This operation completes the vault locking process by transitioning the vault lock from the `InProgress` state to the `Locked` state, which causes the vault lock policy to become unchangeable. A vault lock is put into the `InProgress` state by calling Initiate Vault Lock (POST lock-policy). You can obtain the state of the vault lock by calling Get Vault Lock (GET lock-policy). For more information about the vault locking process, see Amazon Glacier Vault Lock.

This operation is idempotent. This request is always successful if the vault lock is in the `Locked` state and the provided lock ID matches the lock ID originally used to lock the vault.

If an invalid lock ID is passed in the request when the vault lock is in the `Locked` state, the operation returns an `AccessDeniedException` error. If an invalid lock ID is passed in the request when the vault lock is in the `InProgress` state, the operation throws an `InvalidParameter` error.

Requests

To complete the vault locking process, send an HTTP `POST` request to the URI of the vault's `lock-policy` subresource with a valid lock ID.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

The `lockId` value is the lock ID obtained from a Initiate Vault Lock (POST lock-policy) request.

Request Parameters

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

If the operation request is successful, the service returns an HTTP `204 No Content` response.

Syntax

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example Request

The following example sends an HTTP POST request with the lock ID to complete the vault locking process.

```
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
```

Example Response

If the request was successful, Amazon Glacier (Amazon Glacier) returns an HTTP `204 No Content` response, as shown in the following example.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

Related Sections

- Abort Vault Lock (DELETE lock-policy)
- Get Vault Lock (GET lock-policy)
- Initiate Vault Lock (POST lock-policy)

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Delete Vault (DELETE vault)

Description

This operation deletes a vault. Amazon Glacier (Amazon Glacier) will delete a vault only if there are no archives in the vault as per the last inventory and there have been no writes to the vault since the last inventory. If either of these conditions is not satisfied, the vault deletion fails (that is, the vault is not removed) and Amazon Glacier returns an error.

You can use the Describe Vault (GET vault) operation that provides vault information, including the number of archives in the vault; however, the information is based on the vault inventory Amazon Glacier last generated.

This operation is idempotent.

> **Note**
>
> When you delete a vault, the vault access policy attached to the vault is also deleted. For more information about vault access policies, see Vault Access Policies.

Requests

To delete a vault, send a DELETE request to the vault resource URI.

Syntax

```
DELETE /AccountId/vaults/VaultName HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example Request

The following example deletes a vault named `examplevault`. The example request is a `DELETE` request to the URI of the resource (the vault) to delete.

```
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
```

Example Response

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

Related Sections

- Create Vault (PUT vault)
- List Vaults (GET vaults)
- Initiate Job (POST jobs)
- Identity and Access Management for Amazon Glacier

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Delete Vault Access Policy (DELETE access-policy)

Description

This operation deletes the access policy associated with the specified vault. The operation is eventually consistent—that is, it might take some time for Amazon Glacier (Amazon Glacier) to completely remove the access policy, and you might still see the effect of the policy for a short time after you send the delete request.

This operation is idempotent. You can invoke delete multiple times, even if there is no policy associated with the vault. For more information about vault access policies, see Vault Access Policies.

Requests

To delete the current vault access policy, send an HTTP `DELETE` request to the URI of the vault's `access-policy` subresource.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

In response, Amazon Glacier returns `204 No Content` if the policy is successfully deleted.

Syntax

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example demonstrates how to delete a vault access policy.

Example Request

In this example, a DELETE request is sent to the `access-policy` subresource of the vault named **`examplevault`**.

```
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
```

Example Response

In response, if the policy is successfully deleted Amazon Glacier returns a `204 No Content` as shown in the following example.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

Related Sections

- Get Vault Access Policy (GET access-policy)
- Set Vault Access Policy (PUT access-policy)

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Delete Vault Notifications (DELETE notification-configuration)

Description

This operation deletes the notification configuration set for a vault Set Vault Notification Configuration (PUT notification-configuration). The operation is eventually consistent—that is, it might take some time for Amazon Glacier (Amazon Glacier) to completely disable the notifications, and you might still receive some notifications for a short time after you send the delete request.

Requests

To delete a vault's notification configuration, send a `DELETE` request to the vault's `notification-configuration` subresource.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example demonstrates how to remove notification configuration for a vault.

Example Request

In this example, a DELETE request is sent to the `notification-configuration` subresource of the vault called `examplevault`.

```
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
```

Example Response

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

Related Sections

- Get Vault Notifications (GET notification-configuration)
- Set Vault Notification Configuration (PUT notification-configuration)
- Identity and Access Management for Amazon Glacier

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Describe Vault (GET vault)

Description

This operation returns information about a vault, including the vault Amazon Resource Name (ARN), the date the vault was created, the number of archives contained within the vault, and the total size of all the archives in the vault. The number of archives and their total size are as of the last vault inventory Amazon Glacier (Amazon Glacier) generated (see Working with Vaults in Amazon Glacier). Amazon Glacier generates vault inventories approximately daily. This means that if you add or remove an archive from a vault, and then immediately send a Describe Vault request, the response might not reflect the changes.

Requests

To get information about a vault, send a GET request to the URI of the specific vault resource.

Syntax

```
GET /AccountId/vaults/VaultName HTTP/1.1
```

```
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
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
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

CreationDate

The UTC date when the vault was created.

Type: A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

LastInventoryDate

The UTC date when Amazon Glacier completed the last vault inventory. For information about initiating an inventory for a vault, see Initiate Job (POST jobs).

Type: A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

NumberOfArchives

The number of archives in the vault as per the last vault inventory. This field will return null if an inventory has not yet run on the vault, for example, if you just created the vault.

Type: Number

SizeInBytes

The total size in bytes of the archives in the vault including any per-archive overhead, as of the last inventory date. This field will return `null` if an inventory has not yet run on the vault, for example, if you just created the vault.

Type: Number

VaultARN

The Amazon Resource Name (ARN) of the vault.

Type: String

VaultName

The vault name that was specified at creation time. The vault name is also included in the vault's ARN.

Type: String

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example Request

The following example demonstrates how to get information about the vault named `examplevault`.

```
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
```

Example Response

```
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"
```

```
}
```

Related Sections

- Create Vault (PUT vault)
- List Vaults (GET vaults)
- Delete Vault (DELETE vault)
- Initiate Job (POST jobs)
- Identity and Access Management for Amazon Glacier

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Get Vault Access Policy (GET access-policy)

Description

This operation retrieves the `access-policy` subresource set on the vault—for more information on setting this subresource, see Set Vault Access Policy (PUT access-policy). If there is no access policy set on the vault, the operation returns a `404 Not found` error. For more information about vault access policies, see Vault Access Policies.

Requests

To return the current vault access policy, send an HTTP `GET` request to the URI of the vault's `access-policy` subresource.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

In response, Amazon Glacier (Amazon Glacier) returns the vault access policy in JSON format in the body of the response.

Syntax

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length

{
  "Policy": "string"
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

Policy

The vault access policy as a JSON string, which uses "\" as an escape character.

Type: String

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example demonstrates how to get a vault access policy.

Example Request

In this example, a GET request is sent to the URI of a vault's `access-policy` subresource.

```
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
```

Example Response

If the request was successful, Amazon Glacier returns the vault access policy as a JSON string in the body of the response. The returned JSON string uses "\" as an escape character, as shown in the Set Vault Access Policy (PUT access-policy) examples. However, the following example shows the returned JSON string without escape characters for readability.

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

Related Sections

- Delete Vault Access Policy (DELETE access-policy)
- Set Vault Access Policy (PUT access-policy)

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Get Vault Lock (GET lock-policy)

Description

This operation retrieves the following attributes from the `lock-policy` subresource set on the specified vault:

- The vault lock policy set on the vault.
- The state of the vault lock, which is either `InProgess` or `Locked`.
- When the lock ID expires. The lock ID is used to complete the vault locking process.
- When the vault lock was initiated and put into the `InProgress` state.

A vault lock is put into the `InProgress` state by calling Initiate Vault Lock (POST lock-policy). A vault lock is put into the `Locked` state by calling Complete Vault Lock (POST lockId). You can stop the vault locking process by calling Abort Vault Lock (DELETE lock-policy). For more information about the vault locking process, see Amazon Glacier Vault Lock.

If there is no vault lock policy set on the vault, the operation returns a `404 Not found` error. For more information about vault lock policies, see Vault Lock Policies.

Requests

To return the current vault lock policy and other attributes, send an HTTP `GET` request to the URI of the vault's `lock-policy` subresource as shown in the following syntax example.

Syntax

```
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

The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon

Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

In response, Amazon Glacier (Amazon Glacier) returns the vault access policy in JSON format in the body of the response.

Syntax

```
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"
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

Policy

The vault lock policy as a JSON string, which uses "\" as an escape character.

Type: String

State

The state of the vault lock.

Type: String

Valid values: `InProgress|Locked`

ExpirationDate

The UTC date and time at which the lock ID expires. This value can be `null` if the vault lock is in a `Locked` state.

Type: A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

CreationDate

The UTC date and time at which the vault lock was put into the `InProgress` state.

Type: A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example demonstrates how to get a vault lock policy.

Example Request

In this example, a `GET` request is sent to the URI of a vault's `lock-policy` subresource.

```
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
```

Example Response

If the request was successful, Amazon Glacier returns the vault access policy as a JSON string in the body of the response. The returned JSON string uses "\" as an escape character, as shown in the Initiate Vault Lock (POST lock-policy) example request. However, the following example shows the returned JSON string without escape characters for readability.

```
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"
}
```

Related Sections

- Abort Vault Lock (DELETE lock-policy)
- Complete Vault Lock (POST lockId)
- Initiate Vault Lock (POST lock-policy)

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Get Vault Notifications (GET notification-configuration)

Description

This operation retrieves the `notification-configuration` subresource set on the vault (see Set Vault Notification Configuration (PUT notification-configuration). If notification configuration for a vault is not set, the operation returns a `404 Not Found` error. For more information about vault notifications, see Configuring Vault Notifications in Amazon Glacier.

Requests

To retrieve the notification configuration information, send a `GET` request to the URI of a vault's `notification-configuration` subresource.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
HTTP/1.1 200 OK
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Content-Type: application/json
Content-Length: length
{
```

```
  "Events": [
    String,
    ...
  ],
  "SNSTopic": String
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

Events

A list of one or more events for which Amazon Glacier (Amazon Glacier) will send a notification to the specified Amazon SNS topic. For information about vault events for which you can configure a vault to publish notifications, see Set Vault Notification Configuration (PUT notification-configuration).

Type: Array

SNSTopic

The Amazon Simple Notification Service (Amazon SNS) topic Amazon Resource Name (ARN). For more information, see Getting Started with Amazon SNS in the *Amazon Simple Notification Service Getting Started Guide*.

Type: String

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example demonstrates how to retrieve the notification configuration for a vault.

Example Request

In this example, a GET request is sent to the `notification-configuration` subresource of a vault.

```
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
```

Example Response

A successful response shows the audit logging configuration document in the body of the response in JSON format. In this example, the configuration shows that notifications for two events (`ArchiveRetrievalCompleted` and `InventoryRetrievalCompleted`) are sent to the Amazon SNS topic `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"
}
```

Related Sections

- Delete Vault Notifications (DELETE notification-configuration)
- Set Vault Notification Configuration (PUT notification-configuration)
- Identity and Access Management for Amazon Glacier

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Initiate Vault Lock (POST lock-policy)

Description

This operation initiates the vault locking process by doing the following:

- Installing a vault lock policy on the specified vault.
- Setting the lock state of vault lock to `InProgress`.
- Returning a lock ID, which is used to complete the vault locking process.

You can set one vault lock policy for each vault and this policy can be up to 20 KB in size. For more information about vault lock policies, see Vault Lock Policies.

You must complete the vault locking process within 24 hours after the vault lock enters the `InProgress` state. After the 24 hour window ends, the lock ID expires, the vault automatically exits the `InProgress` state, and the vault lock policy is removed from the vault. You call Complete Vault Lock (POST lockId) to complete the vault locking process by setting the state of the vault lock to `Locked`.

> **Note**
>
> After a vault lock is in the `Locked` state, you cannot initiate a new vault lock for the vault.

You can stop the vault locking process by calling Abort Vault Lock (DELETE lock-policy). You can get the state of the vault lock by calling Get Vault Lock (GET lock-policy). For more information about the vault locking process, see Amazon Glacier Vault Lock.

If this operation is called when the vault lock is in the `InProgress` state, the operation returns an `AccessDeniedException` error. When the vault lock is in the `InProgress` state you must call Abort Vault Lock (DELETE lock-policy) before you can initiate a new vault lock policy.

Requests

To initiate the vault locking process, send an HTTP POST request to the URI of the `lock-policy` subresource of the vault, as shown in the following syntax example.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

The request body contains the following JSON fields.

Policy

The vault lock policy as a JSON string, which uses "\" as an escape character.

Type: String

Required: Yes

Responses

Amazon Glacier (Amazon Glacier) returns an `HTTP 201 Created` response, if the policy is accepted.

Syntax

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-lock-id: lockId
```

Response Headers

A successful response includes the following response headers, in addition to the response headers that are common to all operations. For more information about common response headers, see Common Response Headers.

Name	Description
`x-amz-lock-id`	The lock ID, which is used to complete the vault locking process. Type: String

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example Request

The following example sends an HTTP PUT request to the URI of the vault's `lock-policy` subresource. The `Policy` JSON string uses "\" as an escape character.

```
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\"}}}]}"}
```

Example Response

If the request was successful, Amazon Glacier returns an `HTTP 201 Created` response, as shown in the following example.

```
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
```

Related Sections

- Abort Vault Lock (DELETE lock-policy)
- Complete Vault Lock (POST lockId)

- Get Vault Lock (GET lock-policy)

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

List Tags For Vault (GET tags)

This operation lists all the tags attached to a vault. The operation returns an empty map if there are no tags. For more information about tags, see Tagging Amazon Glacier Resources.

Request Syntax

To list the tags for a vault, send an HTTP GET request to the tags URI as shown in the following syntax example.

```
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**
>
> The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

If the operation is successful, the service sends back an HTTP `200 OK` response.

Response Syntax

```
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"
      }
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

Tags

The tags attached to the vault. Each tag is composed of a key and a value.

Type: String to String map

Required: Yes

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example: List Tags For a Vault

The following example lists the tags for a vault.

Example Request

In this example, a GET request is sent to retrieve a list of tags from the specified vault.

```
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
```

Example Response

If the request was successful, Amazon Glacier (Amazon Glacier) returns a `HTTP 200 OK` with a list of tags for the vault as shown in the following example.

```
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"
      }
}
```

Related Sections

- Add Tags To Vault (POST tags add)

- Remove Tags From Vault (POST tags remove)

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

List Vaults (GET vaults)

Description

This operation lists all vaults owned by the calling user's account. The list returned in the response is ASCII-sorted by vault name.

By default, this operation returns up to 10 items per request. If there are more vaults to list, the `marker` field in the response body contains the vault Amazon Resource Name (ARN) at which to continue the list with a new List Vaults request; otherwise, the `marker` field is `null`. In your next List Vaults request you set the `marker` parameter to the value Amazon Glacier (Amazon Glacier) returned in the responses to your previous List Vaults request. You can also limit the number of vaults returned in the response by specifying the `limit` parameter in the request.

Requests

To get a list of vaults, you send a `GET` request to the *vaults* resource.

Syntax

```
GET /AccountId/vaults HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation uses the following request parameters.

Name	Description	Required
`limit`	The maximum number of vaults to be returned. The default limit is 10. The number of vaults returned might be fewer than the specified limit, but the number of returned vaults never exceeds the limit. Type: String Constraints: Minimum integer value of 1. Maximum integer value of 10.	No
`marker`	A string used for pagination. `marker` specifies the vault ARN after which the listing of vaults should begin. (The vault specified by `marker` is not included in the returned list.) Get the `marker` value from a previous List Vaults response. You need to include the `marker` only if you are continuing the pagination of results started in a previous List Vaults request. Specifying an empty value ("") for the marker returns a list of vaults starting from the first vault. Type: String Constraints: None	No

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
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
   },
   ...
  ]
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

CreationDate

The date the vault was created, in Coordinated Universal Time (UTC).

Type: String. A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

LastInventoryDate

The date of the last vault inventory, in Coordinated Universal Time (UTC). This field can be null if an inventory has not yet run on the vault, for example, if you just created the vault. For information about initiating an inventory for a vault, see Initiate Job (POST jobs).

Type: A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

Marker

The `vaultARN` that represents where to continue pagination of the results. You use the `marker` in another List Vaults request to obtain more vaults in the list. If there are no more vaults, this value is `null`.

Type: String

NumberOfArchives

The number of archives in the vault as of the last inventory date.

Type: Number

SizeInBytes

The total size, in bytes, of all the archives in the vault including any per-archive overhead, as of the last inventory date.

Type: Number

VaultARN

The Amazon Resource Name (ARN) of the vault.

Type: String

VaultList

An array of objects, with each object providing a description of a vault.

Type: Array

VaultName

The vault name.

Type: String

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example: List All Vaults

The following example lists vaults. Because the `marker` and `limit` parameters are not specified in the request, up to 10 vaults are returned.

Example Request

```
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
```

Example Response

The `Marker` is `null` indicating there are no more vaults to list.

```
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"
   }
  ]
}
```

Example: Partial List of Vaults

The following example returns two vaults starting at the vault specified by the `marker`.

Example Request

```
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
```

Example Response

Two vaults are returned in the list. The `Marker` contains the vault ARN to continue pagination in another List Vaults request.

```
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"
   }
  ]
}
```

Related Sections

- Create Vault (PUT vault)
- Delete Vault (DELETE vault)
- Initiate Job (POST jobs)
- Identity and Access Management for Amazon Glacier

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Remove Tags From Vault (POST tags remove)

This operation removes one or more tags from the set of tags attached to a vault. For more information about tags, see Tagging Amazon Glacier Resources.

This operation is idempotent. The operation will be successful, even if there are no tags attached to the vault.

Request Syntax

To remove tags from a vault, send an HTTP POST request to the tags URI as shown in the following syntax example.

```
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

The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

Request Parameters

Name	Description	Required
`operation=remove`	A single query string parameter `operation` with a value of `remove` to distinguish it from Add Tags To Vault (POST tags add).	Yes

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

The request body contains the following JSON fields.

TagKeys

A list of tag keys. Each corresponding tag is removed from the vault.

Type: array of Strings

Length constraint: Minimum of 1 item in the list. Maximum of 10 items in the list.

Required: Yes

Responses

If the action is successful, the service sends back an HTTP `204 No Content` response with an empty HTTP body.

Syntax

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example Request

The following example sends an HTTP POST request to remove the specified tags.

```
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"
   ]
}
```

Example Response

If the request was successful Amazon Glacier (Amazon Glacier) returns a `HTTP 204 No Content` as shown in the following example.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
```

```
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

Related Sections

- Add Tags To Vault (POST tags add)
- List Tags For Vault (GET tags)

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Set Vault Access Policy (PUT access-policy)

Description

This operation configures an access policy for a vault and will overwrite an existing policy. To configure a vault access policy, send a PUT request to the `access-policy` subresource of the vault. You can set one access policy per vault and the policy can be up to 20 KB in size. For more information about vault access policies, see Vault Access Policies.

Requests

Syntax

To set a vault access policy, send an HTTP PUT request to the URI of the vault's `access-policy` subresource as shown in the following syntax example.

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

The request body contains the following JSON fields.

Policy

The vault access policy as a JSON string, which uses "\" as an escape character.

Type: String

Required: Yes

Responses

In response, Amazon Glacier returns `204 No Content` if the policy is accepted.

Syntax

```
HTTP/1.1 204 No Content
```

```
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example Request

The following example sends an HTTP PUT request to the URI of the vault's `access-policy` subresource. The `Policy` JSON string uses "\" as an escape character.

```
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\"}]}"}
```

Example Response

If the request was successful, Amazon Glacier (Amazon Glacier) returns a `HTTP 204 No Content` as shown in the following example.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

Related Sections

- Delete Vault Access Policy (DELETE access-policy)
- Get Vault Access Policy (GET access-policy)

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Set Vault Notification Configuration (PUT notification-configuration)

Description

Retrieving an archive and a vault inventory are asynchronous operations in Amazon Glacier (Amazon Glacier) for which you must first initiate a job and wait for the job to complete before you can download the job output. You can configure a vault to post a message to an Amazon Simple Notification Service (Amazon SNS) topic when these jobs complete. You can use this operation to set notification configuration on the vault. For more information, see Configuring Vault Notifications in Amazon Glacier.

To configure vault notifications, send a PUT request to the `notification-configuration` subresource of the vault. A notification configuration is specific to a vault; therefore, it is also referred to as a vault subresource. The request should include a JSON document that provides an Amazon Simple Notification Service (Amazon SNS) topic and the events for which you want Amazon Glacier to send notifications to the topic.

You can configure a vault to publish a notification for the following vault events:

- **`ArchiveRetrievalCompleted`—** This event occurs when a job that was initiated for an archive retrieval is completed (Initiate Job (POST jobs)). The status of the completed job can be `Succeeded` or `Failed`. The notification sent to the SNS topic is the same output as returned from Describe Job (GET JobID).
- **`InventoryRetrievalCompleted`—** This event occurs when a job that was initiated for an inventory retrieval is completed (Initiate Job (POST jobs)). The status of the completed job can be `Succeeded` or `Failed`. The notification sent to the SNS topic is the same output as returned from Describe Job (GET JobID).

Amazon SNS topics must grant permission to the vault to be allowed to publish notifications to the topic.

Requests

To set notification configuration on your vault, send a PUT request to the URI of the vault's `notification-configuration` subresource. You specify the configuration in the request body. The configuration includes the Amazon SNS topic name and an array of events that trigger notification to each topic.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

The JSON in the request body contains the following fields.

Events

An array of one or more events for which you want Amazon Glacier to send notification.

Valid Values: `ArchiveRetrievalCompleted` | `InventoryRetrievalCompleted`

Required: yes

Type: Array

SNSTopic

The Amazon SNS topic ARN. For more information, go to Getting Started with Amazon SNS in the *Amazon Simple Notification Service Getting Started Guide*.

Required: yes

Type: String

Responses

In response, Amazon Glacier (Amazon Glacier) returns `204 No Content` if the notification configuration is accepted.

Syntax

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example demonstrates how to configure vault notification.

Example Request

The following request sets the `examplevault` notification configuration so that notifications for two events (`ArchiveRetrievalCompleted` and `InventoryRetrievalCompleted`) are sent to the Amazon SNS topic `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"
}
```

Example Response

A successful response returns a `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
```

Related Sections

- Get Vault Notifications (GET notification-configuration)
- Delete Vault Notifications (DELETE notification-configuration)
- Identity and Access Management for Amazon Glacier

See Also

For more information about using this API in one of the language-specific Amazon SDKs, see the following:

- AWS Command Line Interface

Archive Operations

The following are the archive operations available for use in Amazon Glacier.

Topics

- Delete Archive (DELETE archive)
- Upload Archive (POST archive)

Delete Archive (DELETE archive)

Description

This operation deletes an archive from a vault. You can delete one archive at a time from a vault. To delete the archive you must provide its archive ID in the delete request. You can get the archive ID by downloading the vault inventory for the vault that contains the archive. For more information about downloading the vault inventory, see Downloading a Vault Inventory in Amazon Glacier.

After you delete an archive, you might still be able to make a successful request to initiate a job to retrieve the deleted archive, but the archive retrieval job will fail.

Archive retrievals that are in progress for an archive ID when you delete the archive might or might not succeed according to the following scenarios:

- If the archive retrieval job is actively preparing the data for download when Amazon Glacier (Amazon Glacier) receives the delete archive request, the archival retrieval operation might fail.
- If the archive retrieval job has successfully prepared the archive for download when Amazon Glacier receives the delete archive request, you will be able to download the output.

For more information about archive retrieval, see Downloading an Archive in Amazon Glacier.

This operation is idempotent. Attempting to delete an already-deleted archive does not result in an error.

Requests

To delete an archive you send a DELETE request to the archive resource URI.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example demonstrates how to delete an archive from the vault named `examplevault`.

Example Request

The ID of the archive to be deleted is specified as a subresource of `archives`.

```
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
```

Example Response

If the request is successful, Amazon Glacier responds with `204 No Content` to indicate that the archive is deleted.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

Related Sections

- Initiate Multipart Upload (POST multipart-uploads)
- Upload Archive (POST archive)
- Identity and Access Management for Amazon Glacier

Upload Archive (POST archive)

Description

This operation adds an archive to a vault. For a successful upload, your data is durably persisted. In response, Amazon Glacier (Amazon Glacier) returns the archive ID in the `x-amz-archive-id` header of the response. You should save the archive ID returned so that you can access the archive later.

You must provide a SHA256 tree hash of the data you are uploading. For information about computing a SHA256 tree hash, see Computing Checksums.

> **Note**
>
> The SHA256 tree hash is only required for the Upload Archive (POST archive) action when using the API. It is not required when using the AWS CLI.

When uploading an archive, you can optionally specify an archive description of up to 1,024 printable ASCII characters. Amazon Glacier returns the archive description when you either retrieve the archive or get the vault inventory. Amazon Glacier does not interpret the description in any way. An archive description does not need to be unique. You cannot use the description to retrieve or sort the archive list.

Except for the optional archive description, Amazon Glacier does not support any additional metadata for the archives. The archive ID is an opaque sequence of characters from which you cannot infer any meaning about the archive. So you might maintain metadata about the archives on the client-side. For more information, see Working with Archives in Amazon Glacier.

Archives are immutable. After you upload an archive, you cannot edit the archive or its description.

Requests

To upload an archive, you use the HTTP POST method and scope the request to the `archives` subresource of the vault in which you want to save the archive. The request must include the archive payload size, checksum (SHA256 tree hash), and can optionally include a description of the archive.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This implementation of the operation does not use request parameters.

Request Headers

This operation uses the following request headers, in addition to the request headers that are common to all operations. For more information about the common request headers, see Common Request Headers.

Name	Description	Required
`Content-Length`	The size of the object, in bytes. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.13. Type: Number Default: None Constraints: None	Yes
`x-amz-archive-description`	The optional description of the archive you are uploading . It can be a plain language description or some identifier you choose to assign. The description need not be unique across archives. When you retrieve a vault inventory (see Initiate Job (POST jobs)), it includes this description for each of the archives it returns in response. Type: String Default: None Constraints: The description must be less than or equal to 1,024 characters. The allowable characters are 7-bit ASCII without control codes, specifically ASCII values 32—126 de cimal or 0x20—0x7E hexadecimal.	No
`x-amz-content-sha256`	The SHA256 checksum (a linear hash) of the payload. This is not the same value as you specify in the `x-amz-sha 256-tree-hash` header.	Yes

Name	Description	Required
	Type: String Default: None Constraints: None	
`x-amz-sha256-tree-hash`	The user-computed checksum, SHA256 tree hash, of the payload. For information on computing the SHA256 tree hash, see Computing Checksums. If Amazon Glacier c omputes a different checksum of the payload, it will reject the request. Type: String Default: None Constraints: None	Yes

Request Body

The request body contains the data to upload.

Responses

In response, Amazon Glacier durably stores the archive and returns a URI path to the archive ID.

Syntax

```
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
```

Response Headers

A successful response includes the following response headers, in addition to the response headers that are common to all operations. For more information about common response headers, see Common Response Headers.

Name	Description
`Location`	The relative URI path of the newly added archive resource. Type: String
`x-amz-archive-id`	The ID of the archive. This value is also included as part of the `Location` header. Type: String
`x-amz-sha256-tree-hash`	The checksum of the archive computed by Amazon Glacier. Type: String

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example Request

The following example shows a request to upload an archive.

```
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).>
```

Example Response

The successful response below has a `Location` header where you can get the ID that Amazon Glacier assigned to the archive.

```
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
```

Related Sections

- Working with Archives in Amazon Glacier
- Uploading Large Archives in Parts (Multipart Upload)
- Delete Archive (DELETE archive)
- Identity and Access Management for Amazon Glacier

Multipart Upload Operations

The following are the multipart upload operations available for use in Amazon Glacier.

Topics

- Abort Multipart Upload (DELETE uploadID)

- Complete Multipart Upload (POST uploadID)
- Initiate Multipart Upload (POST multipart-uploads)
- List Parts (GET uploadID)
- List Multipart Uploads (GET multipart-uploads)
- Upload Part (PUT uploadID)

Abort Multipart Upload (DELETE uploadID)

Description

This command for multipart upload operation stops a multipart upload identified by the upload ID.

After the Abort Multipart Upload request succeeds, you cannot use the upload ID to upload any more parts or perform any other operations. Stopping a completed multipart upload fails. However, stopping an already-stopped upload will succeed, for a short time.

This operation is idempotent.

For information about multipart upload, see Uploading Large Archives in Parts (Multipart Upload).

Requests

To stop a multipart upload, send an HTTP `DELETE` request to the URI of the `multipart-uploads` subresource of the vault and identify the specific multipart upload ID as part of the URI.

Syntax

```
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

The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon

Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Example

Example Request

In the following example, a DELETE request is sent to the URI of a multipart upload ID resource.

```
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
```

Example Response

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:00:00 GMT
```

Related Sections

- Initiate Multipart Upload (POST multipart-uploads)
- Upload Part (PUT uploadID)
- Complete Multipart Upload (POST uploadID)
- List Multipart Uploads (GET multipart-uploads)
- List Parts (GET uploadID)
- Uploading Large Archives in Parts (Multipart Upload)
- Identity and Access Management for Amazon Glacier

Complete Multipart Upload (POST uploadID)

Description

You call this multipart upload operation to inform Amazon Glacier (Amazon Glacier) that all the archive parts have been uploaded and Amazon Glacier can now assemble the archive from the uploaded parts.

For information about multipart upload, see Uploading Large Archives in Parts (Multipart Upload).

After assembling and saving the archive to the vault, Amazon Glacier returns the archive ID of the newly created archive resource. After you upload an archive, you should save the archive ID returned to retrieve the archive at a later point.

In the request, you must include the computed SHA256 tree hash of the entire archive you have uploaded. For information about computing a SHA256 tree hash, see Computing Checksums. On the server side, Amazon Glacier also constructs the SHA256 tree hash of the assembled archive. If the values match, Amazon Glacier saves the archive to the vault; otherwise, it returns an error, and the operation fails. The List Parts (GET uploadID) operation returns list of parts uploaded for a specific multipart upload. It includes checksum information for each uploaded part that can be used to debug a bad checksum issue.

Additionally, Amazon Glacier also checks for any missing content ranges. When uploading parts, you specify range values identifying where each part fits in the final assembly of the archive. When assembling the final archive Amazon Glacier checks for any missing content ranges and if there are any missing content ranges, Amazon Glacier returns an error and the Complete Multipart Upload operation fails.

Complete Multipart Upload is an idempotent operation. After your first successful complete multipart upload, if you call the operation again within a short period, the operation will succeed and return the same archive ID. This is useful in the event you experience a network issue or receive a 500 server error, in which case you can repeat your Complete Multipart Upload request and get the same archive ID without creating duplicate archives. Note, however, that after the multipart upload completes, you cannot call the List Parts operation and the multipart upload will not appear in List Multipart Uploads response, even if idempotent complete is possible.

Requests

To complete a multipart upload, you send an HTTP POST request to the URI of the upload ID that Amazon Glacier created in response to your Initiate Multipart Upload request. This is the same URI you used when uploading parts. In addition to the common required headers, you must include the result of the SHA256 tree hash of the entire archive and the total size of the archive in bytes.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses the following request headers, in addition to the request headers that are common to all operations. For more information about the common request headers, see Common Request Headers.

Name	Description	Required
`x-amz-archive-size`	The total size, in bytes, of the entire archive. This value should be the sum of all the sizes of the individual parts that you uploaded. Type: String Default: None Constraints: None	Yes
`x-amz-sha256-tree-hash`	The SHA256 tree hash of the entire archive. It is the tree hash of SHA256 tree hash of the individual parts. If the value you specify in the request does not match the SHA256 tree hash of the final assembled archive as computed by Amazon Glacier, Amazon Glacier returns an error and the request fails.	Yes

Name	Description	Required
	Type: String Default: None Constraints: None	

Request Elements

This operation does not use request elements.

Responses

Amazon Glacier (Amazon Glacier) creates a SHA256 tree hash of the entire archive. If the value matches the SHA256 tree hash of the entire archive you specified in the request, Amazon Glacier adds the archive to the vault. In response it returns the HTTP `Location` header with the URL path of the newly added archive resource. If the archive size or SHA256 that you sent in the request does not match, Amazon Glacier will return an error and the upload remains in the incomplete state. It is possible to retry the Complete Multipart Upload operation later with correct values, at which point you can successfully create an archive. If a multipart upload does not complete, then eventually Amazon Glacier will reclaim the upload ID.

Syntax

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
x-amz-archive-id: ArchiveId
```

Response Headers

A successful response includes the following response headers, in addition to the response headers that are common to all operations. For more information about common response headers, see Common Response Headers.

Name	Description
`Location`	

Name	Description
	The relative URI path of the newly created archive. This URL includes the archive ID that is generated by Amazon Glacier. Type: String
x-amz-archive-id	The ID of the archive. This value is also included as part of the Location header. Type: String

Response Fields

This operation does not return a response body.

Example

Example Request

In this example, an HTTP POST request is sent to the URI that was returned by an Initiate Multipart Upload request. The request specifies both the SHA256 tree hash of the entire archive and the total archive size.

```
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
```

Example Response

The following example response shows that Amazon Glacier successfully created an archive from the parts you uploaded. The response includes the archive ID with complete path.

```
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
```

You can now send HTTP requests to the URI of the newly added resource/archive. For example, you can send a GET request to retrieve the archive.

Related Sections

- Initiate Multipart Upload (POST multipart-uploads)
- Upload Part (PUT uploadID)
- Abort Multipart Upload (DELETE uploadID)
- List Multipart Uploads (GET multipart-uploads)
- List Parts (GET uploadID)
- Uploading Large Archives in Parts (Multipart Upload)
- Delete Archive (DELETE archive)
- Identity and Access Management for Amazon Glacier

Initiate Multipart Upload (POST multipart-uploads)

Description

This operation initiates a multipart upload (see Uploading Large Archives in Parts (Multipart Upload)). Amazon Glacier (Amazon Glacier) creates a multipart upload resource and returns its ID in the response. You use this Upload ID in subsequent multipart upload operations.

When you initiate a multipart upload, you specify the part size in number of bytes. The part size must be a mebibyte (MiB) (1024 kibibytes [KiB]) multiplied by a power of 2—for example, 1048576 (1 MiB), 2097152 (2 MiB), 4194304 (4 MiB), 8388608 (8 MiB), and so on. The minimum allowable part size is 1 MiB, and the maximum is 4 gibibytes (GiB).

Every part you upload using this upload ID, except the last one, must have the same size. The last one can be the same size or smaller. For example, suppose you want to upload a 16.2 MiB file. If you initiate the multipart upload with a part size of 4 MiB, you will upload four parts of 4 MiB each and one part of 0.2 MiB.

> **Note**
>
> You don't need to know the size of the archive when you start a multipart upload because Amazon Glacier does not require you to specify the overall archive size.

After you complete the multipart upload, Amazon Glacier removes the multipart upload resource referenced by the ID. Amazon Glacier will also remove the multipart upload resource if you cancel the multipart upload or it may be removed if there is no activity for a period of 24 hours. The ID may still be available after 24 hours, but applications should not expect this behavior.

Requests

To initiate a multipart upload, you send an HTTP `POST` request to the URI of the `multipart-uploads` subresource of the vault in which you want to save the archive. The request must include the part size and can optionally include a description of the archive.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses the following request headers, in addition to the request headers that are common to all operations. For more information about the common request headers, see Common Request Headers.

Name	Description	Required
`x-amz-part-size`	The size of each part except the last, in bytes. The last part can be smaller than this part size. Type: String Default: None Constraints: The part size must be a mebibyte (1024 KiB) multiplied by a power of 2—for example, 1048576 (1 MiB), 2097152 (2 MiB), 4194304 (4 MiB), 8388608 (8 MiB), and so on. The minimum allowable part size is 1 MiB, and the maximum is 4 GiB (4096 MiB).	Yes
`x-amz-archive-description`	Archive description you are uploading in parts. It can be a plain-language description or some unique identifier you choose to assign. When you retrieve a vault inventory (see Initiate Job (POST jobs)), the inventory includes this description for each of the archives it returns in response. Leading spaces in archive descriptions are removed. Type: String	No

Name	Description	Required
	Default: None Constraints: The description must be less than or equal to 1024 bytes. The allowable characters are 7 bit ASCII without control codes, specifically ASCII values 32-126 decimal or 0x20-0x7E hexadecimal.	

Request Body

This operation does not have a request body.

Responses

In the response, Amazon Glacier creates a multipart upload resource identified by an ID and returns the relative URI path of the multipart upload ID.

Syntax

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
Location: Location
x-amz-multipart-upload-id: multiPartUploadId
```

Response Headers

A successful response includes the following response headers, in addition to the response headers that are common to all operations. For more information about common response headers, see Common Response Headers.

Name	Description
`Location`	The relative URI path of the multipart upload ID Amazon Glacier created. You use this URI path to scope your requests to upload parts, and to complete the multipart upload.

Name	Description
	Type: String
`x-amz-multipart-upload-id`	The ID of the multipart upload. This value is also included as part of the `Location` header. Type: String

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Example

Example Request

The following example initiates a multipart upload by sending an HTTP POST request to the URI of the `multipart-uploads` subresource of a vault named `examplevault`. The request includes headers to specify the part size of 4 MiB (4194304 bytes) and the optional archive description.

```
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
```

Example Response

Amazon Glacier creates a multipart upload resource and adds it to the `multipart-uploads` subresource of the vault. The `Location` response header includes the relative URI path to the multipart upload 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/multipart-uploads/
OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE
x-amz-multipart-upload-id:
 OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-
khxOjyEXAMPLE
```

For information about uploading individual parts, see Upload Part (PUT uploadID).

Related Sections

- Upload Part (PUT uploadID)
- Complete Multipart Upload (POST uploadID)
- Abort Multipart Upload (DELETE uploadID)
- List Multipart Uploads (GET multipart-uploads)
- List Parts (GET uploadID)
- Delete Archive (DELETE archive)
- Uploading Large Archives in Parts (Multipart Upload)
- Identity and Access Management for Amazon Glacier

List Parts (GET uploadID)

Description

This multipart upload operation lists the parts of an archive that have been uploaded in a specific multipart upload identified by an upload ID. For information about multipart upload, see Uploading Large Archives in Parts (Multipart Upload).

You can make this request at any time during an in-progress multipart upload before you complete the multipart upload. Amazon Glacier returns the part list sorted by range you specified in each part upload. If you send a List Parts request after completing the multipart upload, Amazon Glacier (Amazon Glacier) returns an error.

The List Parts operation supports pagination. You should always check the `Marker` field in the response body for a marker at which to continue the list. if there are no more items the `marker` field is `null`. If the `marker` is not null, to fetch the next set of parts you sent another List Parts request with the `marker` request parameter set to the marker value Amazon Glacier returned in response to your previous List Parts request.

You can also limit the number of parts returned in the response by specifying the `limit` parameter in the request.

Requests

Syntax

To list the parts of an in-progress multipart upload, you send a `GET` request to the URI of the multipart upload ID resource. The multipart upload ID is returned when you initiate a multipart upload (Initiate Multipart Upload (POST multipart-uploads)). You may optionally specify `marker` and `limit` parameters.

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

Name	Description	Required
`limit`	The maximum number of parts to be returned. The default limit is 50. The number of parts returned might be fewer than	No

Name	Description	Required
	the specified limit, but the number of returned parts never exceeds the limit. Type: String Constraints: Minimum integer value of 1. Maximum integer value of 50.	
marker	An opaque string used for pagination. marker specifies the part at which the listing of parts should begin. Get the marker value from the response of a previous List Parts response. You need only include the marker if you are continuing the pagination of results started in a previous List Parts request. Type: String Constraints: None	No

Request Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
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
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

ArchiveDescription

The description of the archive that was specified in the Initiate Multipart Upload request. This field is `null` if no archive description was specified in the Initiate Multipart Upload operation.

Type: String

CreationDate

The UTC time that the multipart upload was initiated.

Type: String. A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

Marker

An opaque string that represents where to continue pagination of the results. You use the `marker` in a new List Parts request to obtain more jobs in the list. If there are no more parts, this value is `null`.

Type: String

MultipartUploadId

The ID of the upload to which the parts are associated.

Type: String

PartSizeInBytes

The part size in bytes. This is the same value that you specified in the Initiate Multipart Upload request.

Type: Number

Parts

A list of the part sizes of the multipart upload. Each object in the array contains a `RangeBytes` and `sha256-tree-hash` name/value pair.

Type: Array

RangeInBytes

The byte range of a part, inclusive of the upper value of the range.

Type: String

SHA256TreeHash

The SHA256 tree hash value that Amazon Glacier calculated for the part. This field is never `null`.

Type: String

VaultARN

The Amazon Resource Name (ARN) of the vault to which the multipart upload was initiated.

Type: String

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example: List Parts of a Multipart Upload

The following example lists all the parts of an upload. The example sends an HTTP GET request to the URI of the specific multipart upload ID of an in-progress multipart upload and returns up to 1,000 parts.

Example Request

```
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
```

Example Response

In the response, Amazon Glacier returns a list of uploaded parts associated with the specified multipart upload ID. In this example, there are only two parts. The returned `Marker` field is `null` indicating that there are no more parts of the multipart upload.

```
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"
}
```

Example: List Parts of a Multipart Upload (Specify the Marker and the Limit Request Parameters)

The following example demonstrates how to use pagination to get a limited number of results. The example sends an HTTP GET request to the URI of the specific multipart upload ID of an in-progress multipart upload to return one part. A starting `marker` parameter specifies at which part to start the part list. You can get the `marker` value from the response of a previous request for a part list. Furthermore, in this example, the `limit` parameter is set to 1 and returns one part. Note that the `Marker` field is not `null`, indicating that there is at least one more part to obtain.

Example Request

```
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
```

Example Response

In the response, Amazon Glacier returns a list of uploaded parts that are associated with the specified in-progress multipart upload ID.

```
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"
}
```

Related Sections

- Initiate Multipart Upload (POST multipart-uploads)
- Upload Part (PUT uploadID)
- Complete Multipart Upload (POST uploadID)
- Abort Multipart Upload (DELETE uploadID)
- List Multipart Uploads (GET multipart-uploads)
- Uploading Large Archives in Parts (Multipart Upload)
- Identity and Access Management for Amazon Glacier

List Multipart Uploads (GET multipart-uploads)

Description

This multipart upload operation lists in-progress multipart uploads for the specified vault. An in-progress multipart upload is a multipart upload that has been initiated by an Initiate Multipart Upload (POST multipart-uploads) request, but has not yet been completed or stopped. The list returned in the List Multipart Upload response has no guaranteed order.

The List Multipart Uploads operation supports pagination. By default, this operation returns up to 50 multipart uploads in the response. You should always check the `marker` field in the response

body for a marker at which to continue the list; if there are no more items the `marker` field is `null`.

If the `marker` is not null, to fetch the next set of multipart uploads you sent another List Multipart Uploads request with the `marker` request parameter set to the marker value Amazon Glacier (Amazon Glacier) returned in response to your previous List Multipart Uploads request.

Note the difference between this operation and the List Parts (GET uploadID)) operation. The List Multipart Uploads operation lists all multipart uploads for a vault. The List Parts operation returns parts of a specific multipart upload identified by an Upload ID.

For information about multipart upload, see Uploading Large Archives in Parts (Multipart Upload).

Requests

Syntax

To list multipart uploads, send a GET request to the URI of the `multipart-uploads` subresource of the vault. You may optionally specify `marker` and `limit` parameters.

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

Name	Description	Required
`limit`		No

Name	Description	Required
	Specifies the maximum number of uploads returned in the response body. If not specified, the List Uploads operation returns up to 50 uploads. Type: String Constraints: Minimum integer value of 1. Maximum integer value of 50.	
`marker`	An opaque string used for pagination. `marker` specifies the upload at which the listing of uploads should begin. Get the `marker` value from a previous List Uploads response. You need only include the `marker` if you are continuing the pagination of results started in a previous List Uploads request. Type: String Constraints: None	No

Request Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
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
    },
   ...
  ]
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

ArchiveDescription

The description of the archive that was specified in the Initiate Multipart Upload request. This field is `null` if no archive description was specified in the Initiate Multipart Upload operation.

Type: String

CreationDate

The UTC time that the multipart upload was initiated.

Type: String. A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

Marker

An opaque string that represents where to continue pagination of the results. You use the `marker` in a new List Multipart Uploads request to obtain more uploads in the list. If there are no more uploads, this value is `null`.

Type: String

PartSizeInBytes

The part size specified in the Initiate Multipart Upload (POST multipart-uploads) request. This is the size of all the parts in the upload except the last part, which may be smaller than this size.

Type: Number

MultipartUploadId

The ID of the multipart upload.

Type: String

UploadsList

A list of metadata about multipart upload objects. Each item in the list contains a set of name-value pairs for the corresponding upload, including `ArchiveDescription`, `CreationDate`, `MultipartUploadId`, `PartSizeInBytes`, and `VaultARN`.

Type: Array

VaultARN

The Amazon Resource Name (ARN) of the vault that contains the archive.

Type: String

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example: List All Multipart Uploads

The following example lists all the multipart uploads in progress for the vault. The example shows an HTTP `GET` request to the URI of the `multipart-uploads` subresource of a specified vault. Because the `marker` and `limit` parameters are not specified in the request, up to 1,000 in-progress multipart uploads are returned.

Example Request

```
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
```

Example Response

In the response Amazon Glacier returns a list of all in-progress multipart uploads for the specified vault. The `marker` field is `null`, which indicates that there are no more uploads to list.

```
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"
    }
  ]
}
```

Example: Partial List of Multipart Uploads

The following example demonstrates how to use pagination to get a limited number of results. The example shows an HTTP GET request to the URI of the `multipart-uploads` subresource for a specified vault. In this example, the `limit` parameter is set to 1, which means that only one upload is returned in the list, and the `marker` parameter indicates the multipart upload ID at which the returned list begins.

Example Request

```
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
```

Example Response

In the response, Amazon Glacier (Amazon Glacier) returns a list of no more than two in-progress multipart uploads for the specified vault, starting at the specified marker and returning two results.

```
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"
    }
  ]
}
```

Related Sections

- Initiate Multipart Upload (POST multipart-uploads)
- Upload Part (PUT uploadID)
- Complete Multipart Upload (POST uploadID)
- Abort Multipart Upload (DELETE uploadID)
- List Parts (GET uploadID)
- Uploading Large Archives in Parts (Multipart Upload)
- Identity and Access Management for Amazon Glacier

Upload Part (PUT uploadID)

Description

This multipart upload operation uploads a part of an archive. You can upload archive parts in any order because in your Upload Part request you specify the range of bytes in the assembled archive that will be uploaded in this part. You can also upload these parts in parallel. You can upload up to 10,000 parts for a multipart upload.

For information about multipart upload, see Uploading Large Archives in Parts (Multipart Upload).

Amazon Glacier (Amazon Glacier) rejects your upload part request if any of the following conditions is true:

- **SHA256 tree hash does not match—**To ensure that part data is not corrupted in transmission, you compute a SHA256 tree hash of the part and include it in your request. Upon receiving the part data, Amazon Glacier also computes a SHA256 tree hash. If the two hash values don't

match, the operation fails. For information about computing a SHA256 tree hash, see Computing Checksums.

- **SHA256 linear hash does not match—**Required for authorization, you compute a SHA256 linear hash of the entire uploaded payload and include it in your request. For information about computing a SHA256 linear hash, see Computing Checksums.
- **Part size does not match—**The size of each part except the last must match the size that is specified in the corresponding Initiate Multipart Upload (POST multipart-uploads) request. The size of the last part must be the same size as, or smaller than, the specified size.

  > **Note**
  >
  > If you upload a part whose size is smaller than the part size you specified in your initiate multipart upload request and that part is not the last part, then the upload part request will succeed. However, the subsequent Complete Multipart Upload request will fail.

- **Range does not align—**The byte range value in the request does not align with the part size specified in the corresponding initiate request. For example, if you specify a part size of 4194304 bytes (4 MB), then 0 to 4194303 bytes (4 MB —1) and 4194304 (4 MB) to 8388607 (8 MB —1) are valid part ranges. However, if you set a range value of 2 MB to 6 MB, the range does not align with the part size and the upload will fail.

This operation is idempotent. If you upload the same part multiple times, the data included in the most recent request overwrites the previously uploaded data.

Requests

You send this HTTP PUT request to the URI of the upload ID that was returned by your Initiate Multipart Upload request. Amazon Glacier uses the upload ID to associate part uploads with a specific multipart upload. The request must include a SHA256 tree hash of the part data (`x-amz-SHA256-tree-hash` header), a SHA256 linear hash of the entire payload (`x-amz-content-sha256` header), the byte range (`Content-Range` header), and the length of the part in bytes (`Content-Length` header).

Syntax

```
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

The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses the following request headers, in addition to the request headers that are common to all operations. For more information about the common request headers, see Common Request Headers.

Name	Description	Required
`Content-Length`	Identifies the length of the part in bytes. Type: String Default: None Constraints: None	No
`Content-Range`	Identifies the range of bytes in the assembled archive that will be uploaded in this part. Amazon	Yes

Name	Description	Required
	Glacier uses this information to assemble the archive in the proper sequence. The format of this header follows RFC 2616. An example header is `Content-Range:bytes 0-4194303/*` . Type: String Default: None Constraints: The range cannot be greater than the part size that you specified when you initiated the multipart upload.	
`x-amz-content-sha256`	The SHA256 checksum (a linear hash) of the uploaded payload. This is not the same value as you specify in the `x-amz-sha256-tree-hash` header. Type: String Default: None Constraints: None	Yes
`x-amz-sha256-tree-hash`	Specifies a SHA256 tree hash of the data being uploaded. For information about computing a SHA256 tree hash, see Computing Checksums. Type: String Default: None Constraints: None	Yes

Request Body

The request body contains the data to upload.

Responses

Upon a successful part upload, Amazon Glacier returns a `204 No Content` response.

Syntax

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-sha256-tree-hash: ChecksumComputedByAmazonGlacier
```

Response Headers

A successful response includes the following response headers, in addition to the response headers that are common to all operations. For more information about common response headers, see Common Response Headers.

Name	Description
`x-amz-sha256-tree-hash`	The SHA256 tree hash that Amazon Glacier computed for the uploaded part. Type: String

Response Body

This operation does not return a response body.

Example

The following request uploads a 4 MB part. The request sets the byte range to make this the first part in the archive.

Example Request

The example sends an HTTP PUT request to upload a 4 MB part. The request is sent to the URI of the Upload ID that was returned by the Initiate Multipart Upload request. The `Content-Range` header identifies the part as the first 4 MB data part of the archive.

```
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
```

To upload the next part, the procedure is the same; however, you must calculate a new SHA256 tree hash of the part you are uploading and also specify a new byte range to indicate where the part will go in the final assembly. The following request uploads another part using the same upload ID. The request specifies the next 4 MB of the archive after the previous request and a part size of 4 MB.

```
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
```

The parts can be uploaded in any order; Amazon Glacier uses the range specification for each part to determine the order in which to assemble them.

Example Response

```
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
```

Related Sections

- Initiate Multipart Upload (POST multipart-uploads)
- Upload Part (PUT uploadID)
- Complete Multipart Upload (POST uploadID)
- Abort Multipart Upload (DELETE uploadID)
- List Multipart Uploads (GET multipart-uploads)
- List Parts (GET uploadID)
- Uploading Large Archives in Parts (Multipart Upload)
- Identity and Access Management for Amazon Glacier

Job Operations

The following are the job operations available in Amazon Glacier.

Topics

- Describe Job (GET JobID)
- Get Job Output (GET output)
- Initiate Job (POST jobs)
- List Jobs (GET jobs)

Describe Job (GET JobID)

Description

This operation returns information about a job you previously initiated, including the job initiation date, the user who initiated the job, the job status code/message, and the Amazon Simple Notification Service (Amazon SNS) topic to notify after Amazon Glacier (Amazon Glacier) completes the job. For more information about initiating a job, see Initiate Job (POST jobs).

> **Note**
>
> This operation enables you to check the status of your job. However, we strongly recommend that you set up an Amazon SNS topic and specify it in your initiate job request so that Amazon Glacier can notify the topic after it completes the job.

A job ID will not expire for at least 24 hours after Amazon Glacier completes the job.

Requests

Syntax

To obtain information about a job, you use the HTTP `GET` method and scope the request to the specific job. Note that the relative URI path is the same one that Amazon Glacier returned to you when you initiated the job.

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

> **Note**
>
> In the request, if you omit the `JobID`, the response returns a list of all active jobs on the specified vault. For more information about listing jobs, see List Jobs (GET jobs).

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
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"
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

Action

The job type. It is either `ArchiveRetrieval`, `InventoryRetrieval`, or `Select`.

Type: String

ArchiveId

The archive ID requested for a select or archive retrieval job. Otherwise, this field is `null`.

Type: String

ArchiveSHA256TreeHash

The SHA256 tree hash of the entire archive for an archive retrieval job. For inventory retrieval jobs, this field is `null`.

Type: String

ArchiveSizeInBytes

For an `ArchiveRetrieval` job, this is the size in bytes of the archive being requested for download. For the `InventoryRetrieval` job, the value is `null`.

Type: Number

Completed

The job status. When an archive or inventory retrieval job is completed, you get the job's output using the Get Job Output (GET output).

Type: Boolean

CompletionDate

The Universal Coordinated Time (UTC) time that the job request completed. While the job is in progress, the value is null.

Type: String

CreationDate

The UTC time that the job was created.

Type: A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

InventoryRetrievalParameters

Input parameters used for a range inventory retrieval.

Type: InventoryRetrievalJobInput object

InventorySizeInBytes

For an `InventoryRetrieval` job, this is the size in bytes of the inventory requested for download. For the `ArchiveRetrieval` or `Select` job, the value is `null`.

Type: Number

JobDescription

The job description you provided when you initiated the job.

Type: String

JobId

The ID that identifies the job in Amazon Glacier.

Type: String

JobOutputPath

Contains the job output location.

Type: String

OutputLocation

An object that contains information about the location where the select job results and errors are stored.

Type: OutputLocation object

RetrievalByteRange

The retrieved byte range for archive retrieval jobs in the form "*StartByteValue*-*EndByteValue*." If you don't specify a range in the archive retrieval, then the whole archive is retrieved; also *StartByteValue* equals 0, and *EndByteValue* equals the size of the archive minus 1. For inventory retrieval or select jobs, this field is `null`.

Type: String

SelectParameters

An object that contains information about the parameters used for a select.

Type: SelectParameters object

SHA256TreeHash

The SHA256 tree hash value for the requested range of an archive. If the Initiate Job (POST jobs) request for an archive specified a tree-hash aligned range, then this field returns a value. For more information about tree-hash alignment for archive range retrievals, see Receiving Checksums When Downloading Data.

For the specific case when the whole archive is retrieved, this value is the same as the `ArchiveSHA256TreeHash` value.

This field is `null` in the following situations:

- Archive retrieval jobs that specify a range that is not tree-hash aligned.
- Archival jobs that specify a range that is equal to the whole archive and the job status is `InProgress`.
- Inventory jobs.
- Select jobs.

Type: String

SNSTopic

An Amazon SNS topic that receives notification.

Type: String

StatusCode

The code indicating the status of the job.

Valid Values: `InProgress` | `Succeeded` | `Failed`

Type: String

StatusMessage

A friendly message that describes the job status.

Type: String

Tier

The data access tier to use for the select or archive retrieval.

Valid Values: `Bulk` | `Expedited` | `Standard`

Type: String

VaultARN

The Amazon Resource Name (ARN) of the vault of which the job is a subresource.

Type: String

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example shows the request for a job that retrieves an archive.

Example Request: Get job description

```
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
```

Example Response

The response body includes JSON that describes the specified job. Note that for both the inventory retrieval and archive retrieval jobs, the JSON fields are the same. However, when a field doesn't apply to the type of job, its value is `null`. The following is an example response for an archive retrieval job. Note the following:

- The `Action` field value is `ArchiveRetrieval`.
- The `ArchiveSizeInBytes` field shows the size of the archive requested in the archive retrieval job.
- The `ArchiveSHA256TreeHash` field shows the SHA256 tree hash for the entire archive.
- The `RetrievalByteRange` field shows the range requested in the Initiate Job request. In this example, the whole archive is requested.
- The `SHA256TreeHash` field shows the SHA256 tree hash for the range requested in the Initiate Job request. In this example, it is the same as the `ArchiveSHA256TreeHash` field, which means that the whole archive was requested.
- The `InventorySizeInBytes` field value is `null` because it does not apply to an archive retrieval job.

```
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"
}
```

The following is an example response for an inventory retrieval job. Note the following:

- The `Action` field value is `InventoryRetrieval`.
- The `ArchiveSizeInBytes`, `ArchiveSHA256TreeHash`, and `RetrievalByteRange` field values are null because these fields do not apply to an inventory retrieval job.
- The `InventorySizeInBytes` field value is `null` because the job is still in progress, and has not fully prepared the inventory for download. If the job was completed before your describe job request, this field would give you the size of the output.

```
{
   "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"
}
```

The following is an example response for a completed inventory retrieval job that contains a marker used to continue pagination of the vault inventory retrieval.

```
{
    "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
    },
}
```

Related Sections

- Get Job Output (GET output)
- Identity and Access Management for Amazon Glacier

Get Job Output (GET output)

Description

This operation downloads the output of the job you initiated using Initiate Job (POST jobs). Depending on the job type you specified when you initiated the job, the output will be either the content of an archive or a vault inventory.

You can download all the job output or download a portion of the output by specifying a byte range. For both archive and inventory retrieval jobs, you should verify the downloaded size against the size returned in the headers from the **Get Job Output** response.

For archive retrieval jobs, you should also verify that the size is what you expected. If you download a portion of the output, the expected size is based on the range of bytes you specified. For example, if you specify a range of `bytes=0-1048575`, you should verify your download size is 1,048,576 bytes. If you download an entire archive, the expected size is the size of the archive when you uploaded it to Amazon Glacier (Amazon Glacier). The expected size is also returned in the headers from the **Get Job Output** response.

In the case of an archive retrieval job, depending on the byte range you specify, Amazon Glacier returns the checksum for the portion of the data. To ensure the portion you downloaded is the correct data, compute the checksum on the client, verify that the values match, and verify that the size is what you expected.

A job ID does not expire for at least 24 hours after Amazon Glacier completes the job. That is, you can download the job output within the 24-hour period after Amazon Glacier completes the job.

Requests

Syntax

To retrieve a job output, you send the HTTP GET request to the URI of the output of the specific job.

```
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**
>
> The AccountId value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses the following request headers, in addition to the request headers that are common to all operations. For more information about the common request headers, see Common Request Headers.

Name	Description	Required
Range	The range of bytes to retrieve from the output. For example, if you want to download the first 1,048,576 bytes, specify the range as bytes=0-1048575 . For more information, go to Range Header Field Definition. The range is relative to	No

Name	Description	Required
	any range specified in the Initiate Job request. By default, this operation downloads the entire output. If the job output is large, then you can use the Range request header to retrieve a portion of the output. This allows you to download the entire output in smaller chunks of bytes. For example, suppose you have 1 GB job output you want to download and you decide to download 128 MB chunks of data at a time, a total of eight Get Job Output requests. You will use the following process to download the job output: 1. Download a 128 MB chunk of output by specifying the appropriate byte range using the Range header. Verify that all 128 MB of data was received. 2. Along with the data, the response will include a checksum of the payload. You compute the checksum of the payload on the client and compare it with the checksum you received in the response to ensure you received all the expected data. 3. Repeat steps 1 and 2 for all the eight 128 MB chunks of output data, each time specifying the appropriate byte range. 4. After downloading all the parts of the job output, you have a list of eight checksum values. Compute the tree hash of these values to find the checksum of the entire output. Using the Describe Job (GET JobID) operation, obtain job information of the job that provided you the output. The response includes the checksum of the entire archive stored in Amazon Glacier. You compare this value with the checksum you computed to ensure you have downloaded the entire archive content with no errors. Type: String	

Name	Description	Required
	Default: None Constraints: None	

Request Body

This operation does not have a request body.

Responses

Syntax

For a retrieval request that returns all of the job data, the job output response returns a `200 OK` response code. When partial content is requested, for example, if you specified the `Range` header in the request, then the response code `206 Partial Content` is returned.

```
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.]
```

Response Headers

Header	Description
`Content-Range`	The range of bytes returned by Amazon Glacier. If only partial output is downloaded, the response provides the range of bytes Amazon Glacier returned. For example, `bytes 0-1048575/8388608` returns the first 1 MB from 8 MB. For more information about the `Content-Range` header, go to Content-Range Header Field Definition.

Header	Description
	Type: String
`Content-Type`	The Content-Type depends on whether the job output is an archive or a vault inventory. • For archive data, the Content-Type is `application/octet-stream` . • For vault inventory, if you requested CSV format when you initiated the job, the Content-Type is `text/csv`. Otherwise, by default, vault inventory is returned as JSON, and the Content-Type is `application/json` . Type: String

Header	Description
`x-amz-sha256-tree-hash`	The checksum of the data in the response. This header is returned only when retrieving the output for an archive retrieval job. Furthermore, this header appears when the retrieved data range requested in the Initiate Job request is tree hash aligned and the range to download in the Get Job Output is also tree hash aligned. For more informati on about tree hash aligned ranges, see Receiving Checksums When Downloading Data. For example, if in your Initiate Job request you specified a tree hash aligned range to retrieve (which includes the whole archive), then you will receive the checksum of the data you download under the following conditions: • You get the entire range of the retrieved data. • You request a byte range of the retrieved data that has a size of a megabyte (1024 KB) multiplied by a power of 2 and that starts and ends on a multiple of the size of the requested range. For example, if you have 3.1 MB of retrieved data and you specify a range to return that starts at 1 MB and ends at 2 MB, then the `x-amz-sha 256-tree-hash` is returned as a response header. • You request a range to return of the retrieved data that goes to the end of the data, and the start of the range is a multiple of the size of the range to retrieve rounded up to the next power of two but not smaller than one megabyte (1024 KB). For example, if you have 3.1 MB of retrieved data and you specify a range that starts at 2 MB and ends at 3.1 MB (the end of the data), then the `x-amz-sha 256-tree-hash` is returned as a response header. Type: String

Response Body

Amazon Glacier returns the job output in the response body. Depending on the job type, the output can be the archive contents or the vault inventory. In case of a vault inventory, by default the inventory list is returned as the following JSON body.

```
{
 "VaultARN": String,
 "InventoryDate": String,
 "ArchiveList": [
      {"ArchiveId": String,
       "ArchiveDescription": String,
       "CreationDate": String,
       "Size": Number,
       "SHA256TreeHash": String
      },
      ...
    ]
}
```

If you requested the comma-separated values (CSV) output format when you initiated the vault inventory job, then the vault inventory is returned in CSV format in the body. The CSV format has five columns "ArchiveId", "ArchiveDescription", "CreationDate", "Size", and "SHA256TreeHash" with the same definitions as the corresponding JSON fields.

> **Note**
>
> In the returned CSV format, fields may be returned with the whole field enclosed in double-quotes. Fields that contain a comma or double-quotes are always returned enclosed in double-quotes. For example, `my archive description,1` is returned as `"my archive description,1"`. Double-quote characters that are within returned double-quote enclosed fields are *escaped* by preceding them with a backslash character. For example, `my archive description,1"2` is returned as `"my archive description,1\"2"` and `my archive description,1\"2` is returned as `"my archive description,1\\"2"`. The backslash character is not escaped.

The JSON response body contains the following JSON fields.

ArchiveDescription

The description of an archive.

Type: String

ArchiveId

The ID of an archive.

Type: String

ArchiveList

An array of archive metadata. Each object in the array represents metadata for one archive contained in the vault.

Type: Array

CreationDate

The UTC date and time the archive was created.

Type: A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

InventoryDate

The UTC date and time of the last inventory for the vault that was completed after changes to the vault. Even though Amazon Glacier prepares a vault inventory once a day, the inventory date is only updated if there have been archive additions or deletions to the vault since the last inventory.

Type: A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

SHA256TreeHash

The tree hash of the archive.

Type: String

Size

The size in bytes of the archive.

Type: Number

VaultARN

The Amazon Resource Name (ARN) resource from which the archive retrieval was requested.

Type: String

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example shows the request for a job that retrieves an archive.

Example 1: Download output

This example retrieves data prepared by Amazon Glacier in response to your initiate archive retrieval job request.

Example Request

```
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
```

Example Response

The following is an example response of an archive retrieval job. Note that the `Content-Type` header is `application/octet-stream` and that `x-amz-sha256-tree-hash` header is included in the response, which means that all the job data is returned.

```
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.]
```

The following is an example response of an inventory retrieval job. Note that the `Content-Type` header is `application/json`. Also note that the response does not include the `x-amz-sha256-tree-hash` header.

```
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"
   }
  ]
```

```
}
```

Example 2: Download only partial output

This example retrieves only a portion of the archive prepared by Amazon Glacier in response to your initiate archive retrieval job request. The request uses the optional `Range` header to retrieve only the first 1,024 bytes.

Example Request

```
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
```

Example Response

The following successful response shows the `206 Partial Content` response. In this case, the response also includes a `Content-Range` header that specifies the range of bytes Amazon Glacier returns.

```
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.]
```

Related Sections

- Describe Job (GET JobID)
- Initiate Job (POST jobs)

- Identity and Access Management for Amazon Glacier

Initiate Job (POST jobs)

This operation initiates the following types of Amazon Glacier (Amazon Glacier) jobs:

- `archive-retrieval`— Retrieve an archive
- `inventory-retrieval`— Inventory a vault

Topics

- Initializing an Archive or Vault Inventory Retrieval Job
- Requests
- Responses
- Examples
- Related Sections

Initializing an Archive or Vault Inventory Retrieval Job

Retrieving an archive or a vault inventory are asynchronous operations that require you to initiate a job. Once started, job cannot be cancelled. Retrieval is a two-step process:

1. Initiate a retrieval job by using the Initiate Job (POST jobs) operation.

   > **Important**
   >
   > A data retrieval policy can cause your initiate retrieval job request to fail with a `PolicyEnforcedException`. For more information about data retrieval policies, see Amazon Glacier Data Retrieval Policies. For more information about the `PolicyEnforcedException` exception, see Error Responses.

2. After the job completes, download the bytes using the Get Job Output (GET output) operation.

The retrieval request is ran asynchronously. When you initiate a retrieval job, Amazon Glacier creates a job and returns a job ID in the response. When Amazon Glacier completes the job, you can

get the job output (archive or inventory data). For information about getting job output, see the Get Job Output (GET output) operation.

The job must complete before you can get its output. To determine when a job is complete, you have the following options:

- **Use an Amazon SNS notification—** You can specify an Amazon SNS topic to which Amazon Glacier can post a notification after the job is completed. You can specify an SNS topic per job request. The notification is sent only after Amazon Glacier completes the job. In addition to specifying an SNS topic per job request, you can configure vault notifications for a vault so that job notifications are sent for all retrievals. For more information, see Set Vault Notification Configuration (PUT notification-configuration).
- **Get job details—** You can make a Describe Job (GET JobID) request to obtain job status information while a job is in progress. However, it is more efficient to use an Amazon SNS notification to determine when a job is complete.

> **Note**
>
> The information you get via notification is same that you get by calling Describe Job (GET JobID).

If for a specific event, you add both the notification configuration on the vault and also specify an SNS topic in your initiate job request, Amazon Glacier sends both notifications. For more information, see Set Vault Notification Configuration (PUT notification-configuration).

The Vault Inventory

Amazon Glacier updates a vault inventory approximately once a day, starting on the day you first upload an archive to the vault. If there have been no archive additions or deletions to the vault since the last inventory, the inventory date is not updated. When you initiate a job for a vault inventory, Amazon Glacier returns the last inventory it generated, which is a point-in-time snapshot and not real-time data.

After Amazon Glacier creates the first inventory for the vault, it typically takes half a day and up to a day before that inventory is available for retrieval.

You might not find it useful to retrieve a vault inventory for each archive upload. However, suppose that you maintain a database on the client-side associating metadata about the archives you upload to Amazon Glacier. Then, you might find the vault inventory useful to reconcile information, as needed, in your database with the actual vault inventory. For more information about the data fields returned in an inventory job output, see Response Body.

Range Inventory Retrieval

You can limit the number of inventory items retrieved by filtering on the archive creation date or by setting a limit.

Filtering by Archive Creation Date

You can retrieve inventory items for archives created between `StartDate` and `EndDate` by specifying values for these parameters in the **Initiate Job** request. Archives created on or after the `StartDate` and before the `EndDate` are returned. If you provide only the `StartDate` without the `EndDate`, you retrieve the inventory for all archives created on or after the `StartDate`. If you provide only the `EndDate` without the `StartDate`, you get back the inventory for all archives created before the `EndDate`.

Limiting Inventory Items per Retrieval

You can limit the number of inventory items returned by setting the `Limit` parameter in the **Initiate Job** request. The inventory job output contains inventory items up to the specified `Limit`. If there are more inventory items available, the result is paginated. After a job is complete, you can use the Describe Job (GET JobID) operation to get a marker that you use in a subsequent **Initiate Job** request. The marker indicates the starting point to retrieve the next set of inventory items. You can page through your entire inventory by repeatedly making **Initiate Job** requests with the marker from the previous **Describe Job** output. You do so until you get a marker from **Describe Job** that returns null, indicating that there are no more inventory items available.

You can use the `Limit` parameter together with the date range parameters.

Ranged Archive Retrieval

You can initiate archive retrieval for the whole archive or a range of the archive. In the case of ranged archive retrieval, you specify a byte range to return or the whole archive. The range specified must be megabyte (MB) aligned. In other words, the range start value must be divisible by 1 MB and the range end value plus 1 must be divisible by 1 MB or equal the end of the archive.

If the ranged archive retrieval is not megabyte-aligned, this operation returns a `400` response. Furthermore, to ensure that you get checksum values for data you download using **Get Job Output** (Get Job Output (GET output)), the range must be tree-hash aligned. For more information about tree-hash aligned ranges, see Receiving Checksums When Downloading Data.

Expedited, Standard, and Bulk Tiers

When initiating an archive retrieval job, you can specify one of the following options in the `Tier` field of the request body:

- **`Expedited`** – Expedited allows you to quickly access your data when occasional urgent requests for restoring archives are required. For all but the largest archives (250 MB+), data accessed using the Expedited tier is typically made available within 1–5 minutes.
- **`Standard`** – Standard allows you to access any of your archives within several hours. Data accessed using the Standard tier typically made available within 3–5 hours. This option is the default one for job requests that don't specify the tier option.
- **`Bulk`** – Bulk is the lowest-cost tier for Amazon Glacier, enabling you to retrieve large amounts, even petabytes, of data inexpensively in a day. Data accessed using the Bulk tier is typically made available within 5–12 hours.

For more information about expedited and bulk retrievals, see Retrieving Amazon Glacier Archives.

Requests

To initiate a job, you use the HTTP `POST` method and scope the request to the vault's `jobs` subresource. You specify details of the job request in the JSON document of your request. The job type is specified with the `Type` field. Optionally, you can specify an `SNSTopic` field to indicate an Amazon SNS topic to which Amazon Glacier can post notification after it completes the job.

> **Note**
>
> To post a notification to Amazon SNS, you must create the topic yourself if it doesn't already exist. Amazon Glacier doesn't create the topic for you. The topic must have permissions to receive publications from a Amazon Glacier vault. Amazon Glacier doesn't verify if the vault has permission to publish to the topic. If the permissions are not configured appropriately, you might not receive notification even after the job completes.

Syntax

The following is the request syntax for initiating a job.

```
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

The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon

Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Body

The request accepts the following data in JSON format in the body of the request.

jobParameters

Provides options for specifying job information.

Type: jobParameters object

Required: Yes

Responses

Amazon Glacier creates the job. In the response, it returns the URI of the job.

Syntax

```
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
```

Response Headers

Header	Description
Location	The relative URI path of the job. You can use this URI path to find the job status. For more information, see Describe Job (GET JobID). Type: String Default: None

Header	Description
`x-amz-job-id`	The ID of the job. This value is also included as part of the `Location` header. Type: String Default: None
`x-amz-job-output-path`	The path to the location of where the select results are stored. Type: String Default: None

Response Body

This operation does not return a response body.

Errors

This operation includes the following error or errors, in addition to the possible errors common to all Amazon Glacier operations. For information about Amazon Glacier errors and a list of error codes, see Error Responses.

Code	Description	HTTP Status Code	Type
`InsufficientCapacityException`	Returned if there is insufficient capacity to process this expedited request. This error only applies to expedited retrievals and not to standard or bulk retrievals.	`503 Service Unavailable`	Server

Examples

Example Request: Initiate an archive retrieval job

```
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"
}
```

The following is an example of the body of a request that specifies a range of the archive to retrieve using the `RetrievalByteRange` field.

```
{
  "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"
}
```

Example Response

```
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
```

Example Request: Initiate an inventory retrieval job

The following request initiates an inventory retrieval job to get a list of archives from the `examplevault` vault. The `Format` set to `CSV` in the body of the request indicates that the inventory is returned in CSV format.

```
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"
}
```

Example Response

```
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
```

Example Requests: Initiate an inventory retrieval job by using date filtering with a set limit, and a subsequent request to retrieve the next page of inventory items.

The following request initiates a vault inventory retrieval job by using date filtering and setting a limit.

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

The following request is an example of a subsequent request to retrieve the next page of inventory items using a marker obtained from Describe Job (GET JobID).

```
{
    "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
    },
}
```

Example Response

```
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/
```

Related Sections

- Describe Job (GET JobID)
- Get Job Output (GET output)
- Identity and Access Management for Amazon Glacier

List Jobs (GET jobs)

Description

This operation lists jobs for a vault, including jobs that are in-progress and jobs that have recently finished.

> **Note**
>
> Amazon Glacier (Amazon Glacier) retains recently completed jobs for a period before deleting them; however, it eventually removes completed jobs. The output of completed jobs can be retrieved. Retaining completed jobs for a period of time after they have completed enables you to get a job output in the event you miss the job completion notification, or your first attempt to download it fails. For example, suppose that you start an archive retrieval job to download an archive. After the job completes, you start to download the archive but encounter a network error. In this scenario, you can retry and download the archive while the job exists.

The `List Jobs` operation supports pagination. You should always check the response `Marker` field. If there are no more jobs to list, the `Marker` field is set to `null`. If there are more jobs to list, the `Marker` field is set to a non-null value, which you can use to continue the pagination of the list. To return a list of jobs that begins at a specific job, set the `marker` request parameter to the `Marker` value for that job that you obtained from a previous `List Jobs` request.

You can set a maximum limit for the number of jobs returned in the response by specifying the `limit` parameter in the request. The default limit is 50. The number of jobs returned might be fewer than the limit, but the number of returned jobs never exceeds the limit.

Additionally, you can filter the jobs list returned by specifying the optional `statuscode` parameter or `completed` parameter, or both. Using the `statuscode` parameter, you can specify to return only jobs that match either the `InProgress`, `Succeeded`, or `Failed` status. Using the `completed` parameter, you can specify to return only jobs that were completed (`true`) or jobs that were not completed (`false`).

Requests

Syntax

To return a list of jobs of all types, send a `GET` request to the URI of the vault's `jobs` subresource.

```
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**
>
> The `AccountId` value is the AWS account ID of the account that owns the vault. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you use an account ID, do not include any hyphens ('-') in the ID.

Request Parameters

Name	Description	Required
`completed`	The state of the jobs to return. You can specify `true` or `false`. Type: Boolean Constraints: None	No
`limit`	The maximum number of jobs to be returned. The default limit is 50. The number of jobs returned might be fewer than the specified limit, but the number of returned jobs never exceeds the limit. Type: String Constraints: Minimum integer value of 1. Maximum integer value of 50.	No
`marker`	An opaque string used for pagination that specifies the job at which the listing of jobs should begin. You get the `marker` value from a previous `List Jobs` response. You only need to include the `marker` if you are continuing the pagination of the results started in a previous `List Jobs` request. Type: String Constraints: None	No
`statuscode`	The type of job status to return. Type: String Constraints: One of the values `InProgress` , `Succeeded` , or `Failed`.	No

Request Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
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"
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

JobList

A list of job objects. Each job object contains metadata describing the job.

Type: Array of GlacierJobDescription objects

Marker

An opaque string that represents where to continue pagination of the results. You use the `marker` value in a new `List Jobs` request to obtain more jobs in the list. If there are no more jobs to list, this value is `null`.

Type: String

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following examples demonstrate how to return information about vault jobs. The first example returns a list of two jobs, and the second example returns a subset of jobs.

Example: Return All Jobs

Example Request

The following GET request returns the jobs for a vault.

```
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
```

Example Response

The following response includes an archive retrieval job and an inventory retrieval job that contains a marker used to continue pagination of the vault inventory retrieval. The response also shows the `Marker` field set to `null`, which indicates there are no more jobs to list.

```
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
}
```

Example: Return a Partial List of Jobs

Example Request

The following GET request returns the job specified by the `marker` parameter. Setting the `limit` parameter as 2 specifies that up to two jobs are returned.

```
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
```

Example Response

The following response shows two jobs returned and the `Marker` field set to a non-null value that can be used to continue pagination of the job list.

```
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"
}
```

Related Sections

- Describe Job (GET JobID)
- Identity and Access Management for Amazon Glacier

Data Types Used in Job Operations

The following are data types used with the job operations in Amazon Glacier.

Topics

CSVInput

Contains information about the comma-separated values (CSV) file.

Contents

Comments

A single character used to indicate that a row should be ignored when the character is present at the start of that row.

Type: String

Required: no

FieldDelimiter

A single character used to separate individual fields from each other within a record. The character must be a \n, \r, or an ASCII character in the range 32–126. The default is a comma (,).

Type: String

Default: ,

Required: no

FileHeaderInfo

A value that describes what to do with the first line of the input.

Type: String

Valid Values: `Use` | `Ignore` | `None`

Required: no

QuoteCharacter

A single character used as an escape character where the field delimiter is part of the value.

Type: String

Required: no

QuoteEscapeCharacter

A single character used for escaping the quotation-mark character inside an already escaped value.

Type: String

Required: no

RecordDelimiter

A single character used to separate individual records from each other.

Type: String

Required: no

More Info

- Initiate Job (POST jobs)

CSVOutput

Contains information about the comma-separated values (CSV) format that the job results are stored in.

Contents

FieldDelimiter

A single character used to separate individual fields from each other within a record.

Type: String

Required: no

QuoteCharacter

A single character used as an escape character where the field delimiter is part of the value.

Type: String

Required: no

QuoteEscapeCharacter

A single character used for escaping the quotation-mark character inside an already escaped value.

Type: String

Required: no

QuoteFields

A value that indicates whether all output fields should be contained within quotation marks.

Valid Values: ALWAYS | ASNEEDED

Type: String

Required: no

RecordDelimiter

A single character used to separate individual records from each other.

Type: String

Required: no

More Info

- Initiate Job (POST jobs)

Encryption

Contains information about the encryption used to store the job results in Amazon S3.

Contents

Encryption

The server-side encryption algorithm used when storing job results in Amazon S3. The default is no encryption.

Type: String

Valid Values: aws:kms | AES256

Required: no

KMSContext

Optional. If the encryption type is aws:kms, you can use this value to specify the encryption context for the job results.

Type: String

Required: no

KMSKeyId

The AWS Key Management Service (AWS KMS) key ID to use for object encryption.

Type: String

Required: no

More Info

- Initiate Job (POST jobs)

GlacierJobDescription

Contains the description of an Amazon Glacier (Amazon Glacier) job.

Contents

Action

The job type. It is either `ArchiveRetrieval`, `InventoryRetrieval`, or `Select`.

Type: String

ArchiveId

The archive ID requested for a select or archive retrieval job. Otherwise, this field is `null`.

Type: String

ArchiveSHA256TreeHash

The SHA256 tree hash of the entire archive for an archive retrieval. For inventory retrieval jobs, this field is `null`.

Type: String

ArchiveSizeInBytes

For an `ArchiveRetrieval` job, this is the size in bytes of the archive being requested for download. For the `InventoryRetrieval` job, the value is `null`.

Type: Number

Completed

`true` if the job is completed; `false` otherwise.

Type: Boolean

CompletionDate

The date when the job completed.

The Universal Coordinated Time (UTC) time that the job request completed. While the job is in progress, the value is null.

Type: A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

CreationDate

The Universal Coordinated Time (UTC) date the job started.

Type: A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

InventoryRetrievalParameters

Input parameters used for a range inventory retrieval.

Type: InventoryRetrievalJobInput object

InventorySizeInBytes

For an `InventoryRetrieval` job, this is the size in bytes of the inventory requested for download. For the `ArchiveRetrieval` or `Select` job, the value is `null`.

Type: Number

JobDescription

The job description that you provided when you initiated the job.

Type: String

JobId

The ID that identifies the job in Amazon Glacier.

Type: String

JobOutputPath

Contains the job output location.

Type: String

OutputLocation

An object that contains information about the location where the select job results and errors are stored.

Type: OutputLocation object

RetrievalByteRange

The retrieved byte range for archive retrieval jobs in the form "*StartByteValue*-*EndByteValue*." If no range was specified in the archive retrieval, then the whole archive is retrieved and *StartByteValue* equals 0 and *EndByteValue* equals the size of the archive minus 1. For inventory retrieval jobs, this field is `null`.

Type: String

SelectParameters

An object that contains information about the parameters used for a select.

Type: SelectParameters object

SHA256TreeHash

The SHA256 tree hash value for the requested range of an archive. If the Initiate Job (POST jobs) request for an archive specified a tree-hash aligned range, then this field returns a value. For more information about tree-hash alignment for archive range retrievals, see Receiving Checksums When Downloading Data.

For the specific case in which the whole archive is retrieved, this value is the same as the `ArchiveSHA256TreeHash` value.

This field is `null` in the following situations:

- Archive retrieval jobs that specify a range that is not tree-hash aligned.
- Archival jobs that specify a range that is equal to the whole archive and the job status is `InProgress`.

- Inventory jobs.
- Select jobs.

Type: String

SNSTopic

The Amazon Resource Name (ARN) that represents an Amazon SNS topic where notification of job completion or failure is sent, if notification was configured in the job initiation (Initiate Job (POST jobs)).

Type: String

StatusCode

The code indicating the status of the job.

Valid Values: `InProgress` | `Succeeded` | `Failed`

Type: String

StatusMessage

The job status message.

Type: String

Tier

The data access tier to use for the select or archive retrieval.

Valid Values: `Expedited` | `Standard` | `Bulk`

Type: String

VaultARN

The ARN of the vault of which the job is a subresource.

Type: String

More Info

- Initiate Job (POST jobs)

Grant

Contains information about a grant.

Contents

Grantee

The grantee.

Type: Grantee object

Required: no

Permission

The permission given to the grantee.

Type: String

Valid Values: `FULL_CONTROL` | `WRITE` | `WRITE_ACP` | `READ` | `READ_ACP`

Required: no

More Info

- Initiate Job (POST jobs)

Grantee

Contains information about a grantee.

Contents

DisplayName

The screen name of the grantee.

Type: String

Required: no

EmailAddress

The email address of the grantee.

Type: String

Required: no

ID

The canonical user ID of the grantee.

Type: String

Required: no

Type

The type of the grantee.

Type: String

Valid Values: `AmazonCustomerByEmail` | `CanonicalUser` | `Group`

Required: no

URI

The URI of the grantee group.

Type: String

Required: no

More Info

- Initiate Job (POST jobs)

InputSerialization

Describes how the archive is serialized.

Contents

CSV

An object that describes the serialization of a CSV-encoded object.

Type: CSVInput object

Required: no

More Info

- Initiate Job (POST jobs)

InventoryRetrievalJobInput

Provides options for specifying a range inventory retrieval job.

Contents

EndDate

The end of the date range, in UTC time, for a vault inventory retrieval that includes archives created before this date.

Valid Values: A string representation in the ISO 8601 date format (`YYYY-MM-DDThh:mm:ssTZD`) in seconds, for example `2013-03-20T17:03:43Z`.

Type: String. A string representation in the ISO 8601 date format (`YYYY-MM-DDThh:mm:ssTZD`) in seconds, for example `2013-03-20T17:03:43Z`.

Required: no

Format

The output format for the vault inventory list, which is set by the Initiate Job (POST jobs) request when initiating a job to retrieve a vault inventory.

Valid Values: `CSV` | `JSON`

Required: no

Type: String

Limit

The maximum number of inventory items that can be returned for each vault inventory retrieval request.

Valid Values: An integer value greater than or equal to 1.

Type: String

Required: no

Marker

An opaque string that represents where to continue pagination of the vault inventory retrieval results. You use this marker in a new `Initiate Job` request to obtain additional inventory items. If there are no more inventory items, this value is null.

Type: String

Required: no

StartDate

The start of the date range, in UTC time, for a vault inventory retrieval that includes archives created on or after this date.

Valid Values: A string representation in the ISO 8601 date format (`YYYY-MM-DDThh:mm:ssTZD`) in seconds, for example `2013-03-20T17:03:43Z`.

Type: String. A string representation in the ISO 8601 date format (`YYYY-MM-DDThh:mm:ssTZD`) in seconds, for example `2013-03-20T17:03:43Z`.

Required: no

More Info

- Initiate Job (POST jobs)

jobParameters

Provides options for defining a job.

Contents

ArchiveId

The ID of the archive that you want. This field is required if the `Type` field is set to `select` or `archive-retrieval`. An error occurs if you specify this field for an inventory retrieval job request.

Valid Values: Must be a valid archive ID that you obtained from a previous request to Amazon Glacier (Amazon Glacier).

Type: String

Required: Yes when `Type` is set to `select` or `archive-retrieval`.

Description

The optional description for the job.

Valid Values: The description must be less than or equal to 1,024 bytes. The allowable characters are 7-bit ASCII without control codes—specifically, ASCII values 32–126 decimal or 0x20–0x7E hexadecimal.

Type: String

Required: no

Format

(Optional) The output format, when initiating a job to retrieve a vault inventory. If you are initiating an inventory job and don't specify a `Format` field, JSON is the default format.

Valid Values: `CSV` | `JSON`

Type: String

Required: no

InventoryRetrievalParameters

Input parameters used for a range inventory retrieval.

Type: InventoryRetrievalJobInput object

Required: no

OutputLocation

An object that contains information about the location where the select job results are stored.

Type: OutputLocation object

Required: Yes, for `select` jobs.

RetrievalByteRange

The byte range to retrieve for an `archive-retrieval`, in the form "*`StartByteValue`*-*`EndByteValue`*". If this field isn't specified, the whole archive is retrieved. If this field is specified, the byte range must be megabyte (1024*1024) aligned. Megabyte-aligned means that *StartByteValue* must be divisible by 1 MB, and *EndByteValue* plus 1 must be divisible by 1 MB or be the end of the archive specified as the archive byte size value minus 1. If **RetrievalByteRange** is not megabyte-aligned, this operation returns a `400` response.

An error occurs if you specify this field for an `inventory-retrieval` or `select` job request.

Type: String

Required: no

SelectParameters

An object that contains information about the parameters used for a select.

Type: SelectParameters object

Required: no

SNSTopic

The Amazon Resource Name (ARN) of the Amazon SNS topic where Amazon Glacier sends a notification when the job is completed and output is ready for you to download. The specified topic publishes the notification to its subscribers.

The SNS topic must exist. If it doesn't, Amazon Glacier doesn't create it for you. Additionally, the SNS topic must have a policy that allows the account that created the job to publish messages to the topic. For information about SNS topic names, see CreateTopic in the *Amazon Simple Notification Service API Reference*.

Type: String

Required: no

Tier

The tier to use for a select or an archive retrieval job. `Standard` is the default value used.

Valid Values: `Expedited` | `Standard` | `Bulk`

Type: String

Required: no

Type

The job type. You can initiate a job to perform a select query on an archive, retrieve an archive, or get an inventory of a vault.

Valid Values: `select` | `archive-retrieval` | `inventory-retrieval`

Type: String

Required: yes

More Info

- Initiate Job (POST jobs)

OutputLocation

Contains information about the location where the job results and errors are stored.

Contents

S3

An object that describes an Amazon S3 location to receive the results of the restore request.

Type: S3Location

Required: yes

More Info

- Initiate Job (POST jobs)

OutputSerialization

Describes how the output is serialized.

Contents

CSV

An object that describes the serialization of the comma-separated values (CSV)-encoded query results.

Type: CSVOutput object

Required: no

More Info

- Initiate Job (POST jobs)

S3Location

Contains information about the location in Amazon S3 where the job results are stored.

Contents

AccessControlList

A list of grants that control access to the stored results.

Type: Array of Grant objects

Required: no

BucketName

The name of the Amazon S3 bucket where the job results are stored. The bucket must be in the same AWS Region as the vault that contains the input archive object.

Type: String

Required: yes

CannedACL

The canned access control list (ACL) to apply to the job results.

Type: String

Valid Values: `private` | `public-read` | `public-read-write` | `aws-exec-read` | `authenticated-read` | `bucket-owner-read` | `bucket-owner-full-control`

Required: no

Encryption

An object that contains information about the encryption used to store the job results in Amazon S3.

Type: Encryption object

Required: no

Prefix

The prefix that is prepended to the results for this request. The maximum length for the prefix is 512 bytes.

Type: String

Required: yes

StorageClass

The class of storage used to store the job results.

Type: String

Valid Values: `STANDARD` | `REDUCED_REDUNDANCY` | `STANDARD_IA`

Required: no

Tagging

The tag set that is applied to the job results.

Type: String to string map

Required: no

UserMetadata

A map of metadata to store with the job results in Amazon S3.

Type: String to string map

Required: no

More Info

- Initiate Job (POST jobs)

SelectParameters

Contains information about the parameters used for the select.

Contents

Expression

The expression that is used to select the object. The expression must not exceed the quota of 128,000 characters.

Type: String

Required: yes

ExpressionType

The type of the provided expression, for example SQL.

Valid Values: SQL

Type: String

Required: yes

InputSerialization

Describes the serialization format of the object in the select.

Type: InputSerialization object

Required: no

OutputSerialization

Describes how the results of the select job are serialized.

Required: no

Type: OutputSerialization object

More Info

- Initiate Job (POST jobs)

Data Retrieval Operations

The following are the data retrieval–related operations available in Amazon Glacier.

Topics

- Get Data Retrieval Policy (GET policy)
- List Provisioned Capacity (GET provisioned-capacity)
- Purchase Provisioned Capacity (POST provisioned-capacity)
- Set Data Retrieval Policy (PUT policy)

Get Data Retrieval Policy (GET policy)

Description

This operation returns the current data retrieval policy for the AWS account and AWS Region specified in the GET request. For more information about data retrieval policies, see Amazon Glacier Data Retrieval Policies.

Requests

To return the current data retrieval policy, send an HTTP GET request to the data retrieval policy URI as shown in the following syntax example.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

Syntax

```
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
         }
       ]
    }
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

BytesPerHour

The maximum number of bytes that can be retrieved in an hour.

This field will be present only if the value of the **Strategy** field is `BytesPerHour`.

Type: Number

Rules

The policy rule. Although this is a list type, currently there will be only one rule, which contains a Strategy field and optionally a BytesPerHour field.

Type: Array

Strategy

The type of data retrieval policy.

Type: String

Valid values: `BytesPerHour|FreeTier|None`. `BytesPerHour` is equivalent to selecting **Max Retrieval Rate** in the console. `FreeTier` is equivalent to selecting **Free Tier Only** in the console. None is equivalent to selecting **No Retrieval Policy** in the console. For more information about selecting data retrieval policies in the console, see Amazon Glacier Data Retrieval Policies.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example demonstrates how to get a data retrieval policy.

Example Request

In this example, a GET request is sent to the URI of a policy's location.

```
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
```

Example Response

A successful response shows the data retrieval policy in the body of the response in JSON format.

```
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"
          }
       ]
    }
}
```

Related Sections

- Set Data Retrieval Policy (PUT policy)
- Initiate Job (POST jobs)

List Provisioned Capacity (GET provisioned-capacity)

This operation lists the provisioned capacity units for the specified AWS account. For more information about provisioned capacity, see Archive Retrieval Options.

A provisioned capacity unit lasts for one month starting at the date and time of purchase, which is the start date. The unit expires on the expiration date, which is exactly one month after the start date to the nearest second.

If the start date is on the 31st day of a month, the expiration date is the last day of the next month. For example, if the start date is August 31, the expiration date is September 30. If the start date is January 31, the expiration date is February 28. You can see this functionality in the Example Response.

Request Syntax

To list the provisioned retrieval capacity for an account, send an HTTP GET request to the provisioned-capacity URI as shown in the following syntax example.

```
GET /AccountId/provisioned-capacity HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
```

```
x-amz-glacier-version: 2012-06-01
```

> **Note**
>
> The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

If the operation is successful, the service sends back an HTTP `200 OK` response.

Response Syntax

```
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"
      }
}
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

The response body contains the following JSON fields.

CapacityId

The ID that identifies the provisioned capacity unit.

Type: String.

StartDate

The date that the provisioned capacity unit was purchased, in Coordinated Universal Time (UTC).

Type: String. A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

ExpirationDate

The date that the provisioned capacity unit expires, in Coordinated Universal Time (UTC).

Type: String. A string representation in the ISO 8601 date format, for example `2013-03-20T17:03:43.221Z`.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

The following example lists the provisioned capacity units for an account.

Example Request

In this example, a GET request is sent to retrieve a list of the provisioned capacity units for the specified account.

```
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
```

Example Response

If the request was successful, Amazon Glacier (Amazon Glacier) returns a `HTTP 200 OK` with a list of provisioned capacity units for the account as shown in the following example.

The provisioned capacity unit listed first is an example of a unit with a start date of January 31, 2017 and expiration date of February 28, 2017. As stated earlier, if the start date is on the 31st day of a month, the expiration date is the last day of the next month.

```
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" ",
      },
      ...
}
```

Related Sections

- Purchase Provisioned Capacity (POST provisioned-capacity)

Purchase Provisioned Capacity (POST provisioned-capacity)

This operation purchases a provisioned capacity unit for an AWS account.

A provisioned capacity unit lasts for one month starting at the date and time of purchase, which is the start date. The unit expires on the expiration date, which is exactly one month after the start date to the nearest second.

If the start date is on the 31st day of a month, the expiration date is the last day of the next month. For example, if the start date is August 31, the expiration date is September 30. If the start date is January 31, the expiration date is February 28.

Provisioned capacity helps ensure that your retrieval capacity for expedited retrievals is available when you need it. Each unit of capacity ensures that at least three expedited retrievals can be performed every five minutes and provides up to 150 MB/s of retrieval throughput. For more information about provisioned capacity, see Archive Retrieval Options.

> **Note**
>
> There is a limit of two provisioned capacity units per AWS account.

Requests

To purchase provisioned capacity unit for an AWS account send an HTTP POST request to the provisioned-capacity URI.

Syntax

```
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**
>
> The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

Request Parameters

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

This operation does not have a request body.

Responses

If the operation request is successful, the service returns an HTTP `201 Created` response.

Syntax

```
HTTP/1.1 201 Created
x-amzn-RequestId: x-amzn-RequestId
Date: Date
x-amz-capacity-id: CapacityId
```

Response Headers

A successful response includes the following response headers, in addition to the response headers that are common to all operations. For more information about common response headers, see Common Response Headers.

Name	Description
x-amz-capacity-id	The ID that identifies the provisioned capacity unit. Type: String

Response Body

This operation does not return a response body.

Errors

This operation includes the following error or errors, in addition to the possible errors common to all Amazon Glacier operations. For information about Amazon Glacier errors and a list of error codes, see Error Responses.

Code	Description	HTTP Status Code	Type
LimitExceededException	Returned if the given request would exceed the account's limit of provisioned capacity units.	400 Bad Request	Client

Examples

The following example purchases provisioned capacity for an account.

Example Request

The following example sends an HTTP POST request to purchase a provisioned capacity unit.

```
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
```

Example Response

If the request was successful, Amazon Glacier (Amazon Glacier) returns an `HTTP 201 Created` response, as shown in the following example.

```
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
```

Related Sections

- List Provisioned Capacity (GET provisioned-capacity)

Set Data Retrieval Policy (PUT policy)

Description

This operation sets and then enacts a data retrieval policy in the AWS Region specified in the PUT request. You can set one policy per AWS Region for an AWS account. The policy is enacted within a few minutes of a successful PUT operation.

The set policy operation does not affect retrieval jobs that were in progress before the policy was enacted. For more information about data retrieval policies, see Amazon Glacier Data Retrieval Policies.

Requests

Syntax

To set a data retrieval policy, send an HTTP PUT request to the data retrieval policy URI as shown in the following syntax example.

```
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**
>
> The `AccountId` value is the AWS account ID. This value must match the AWS account ID associated with the credentials used to sign the request. You can either specify an AWS account ID or optionally a single '-' (hyphen), in which case Amazon Glacier uses the AWS account ID associated with the credentials used to sign the request. If you specify your account ID, do not include any hyphens ('-') in the ID.

Request Parameters

This operation does not use request parameters.

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers.

Request Body

The request body contains the following JSON fields.

BytesPerHour

The maximum number of bytes that can be retrieved in an hour.

This field is required only if the value of the Strategy field is `BytesPerHour`. Your PUT operation will be rejected if the Strategy field is not set to `BytesPerHour` and you set this field.

Type: Number

Required: Yes, if the Strategy field is set to `BytesPerHour`. Otherwise, no.

Valid Values: Minimum integer value of 1. Maximum integer value of 2^63 - 1 inclusive.

Rules

The policy rule. Although this is a list type, currently there must be only one rule, which contains a Strategy field and optionally a BytesPerHour field.

Type: Array

Required: Yes

Strategy

The type of data retrieval policy to set.

Type: String

Required: Yes

Valid values: `BytesPerHour`|`FreeTier`|None. `BytesPerHour` is equivalent to selecting **Max Retrieval Rate** in the console. `FreeTier` is equivalent to selecting **Free Tier Only** in the console. None is equivalent to selecting **No Retrieval Policy** in the console. For more information about selecting data retrieval policies in the console, see Amazon Glacier Data Retrieval Policies.

Responses

Syntax

```
HTTP/1.1 204 No Content
x-amzn-RequestId: x-amzn-RequestId
Date: Date
```

Response Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers.

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses.

Examples

Example Request

The following example sends an HTTP PUT request with the Strategy field set to `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
          }
       ]
    }
}
```

The following example sends an HTTP PUT request with the Strategy field set to `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"
          }
       ]
    }
}
```

The following example sends an HTTP PUT request with the Strategy field set to 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"
          }
       ]
    }
}
```

Example Response

If the request was successful Amazon Glacier (Amazon Glacier) sets the policy and returns a HTTP `204 No Content` as shown in the following example.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Wed, 10 Feb 2017 12:02:00 GMT
```

Related Sections

- Get Data Retrieval Policy (GET policy)
- Initiate Job (POST jobs)

Document History

- **Current product version:** 2012-06-01

The following table describes the important changes in each release of the *Amazon Glacier Developer Guide* from July 5, 2018, onward. For notification about updates to this documentation, you can subscribe to an RSS feed.

Change	Description	Date
Improved start times for Standard restore requests made through S3 Batch Operations	Standard retrievals for restore requests that are made through S3 Batch Operations now can start within minutes. For more information, see Archive Retrieval Options.	August 9, 2023
Amazon S3 supports higher restore request rates for S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive	Amazon S3 supports restore requests at a rate of up to 1,000 transactions per second, per AWS account for the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes.	November 15, 2022
Amazon Glacier name change	Amazon Glacier is now Amazon Glacier to better reflect Glacier's integration with Amazon S3.	November 20, 2018
Updates now available over RSS	You can now subscribe to an RSS feed to receive notificat ions about updates to the Amazon Glacier Developer Guide guide.	July 5, 2018

Earlier Updates

The following table describes the important changes in each release of the *Amazon Glacier Developer Guide* before July 5, 2018.

Change	Description	Release Date
Expedited and Bulk Data Retrievals	Amazon Glacier now supports Expedited and Bulk data retrievals in addition to Standard retrievals. For more information, see Archive Retrieval Options.	November 21, 2016
Vault Lock	Amazon Glacier now supports Vault Lock, which allows you to easily deploy and enforce compliance controls on individual Amazon Glacier vaults with a Vault Lock policy. For more information, see Amazon Glacier Vault Lock and Vault Lock Policies.	July 8, 2015
Vault tagging	Amazon Glacier now allows you to tag your Amazon Glacier vaults for easier resource and cost management. Tags are labels that you can define and associate with your vaults, and using tags adds filtering capabilities to operations such as AWS cost reports. For more information, see Tagging Amazon Glacier Resources and Tagging Your Amazon Glacier Vaults.	June 22, 2015
Vault access policies	Amazon Glacier now supports managing access to your individual Amazon Glacier vaults by using vault access policies. You can now define an access policy directly on a vault, making it easier to grant vault access to users and business groups internal to your organization, as well as to your external business partners. For more information, see Vault Access Policies.	April 27, 2015
Data retrieval policies and audit logging	Amazon Glacier now supports data retrieval policies and audit logging. Data retrieval policies allow you to easily set data retrieval limits and simplify data	December 11, 2014

Change	Description	Release Date
	retrieval cost management. You can define your own data retrieval limits with a few clicks in the AWS Management Console or by using the Amazon Glacier API. For more information, see Amazon Glacier Data Retrieval Policies. In addition, Amazon Glacier now supports audit logging with AWS CloudTrail, which records Amazon Glacier API calls for your account and delivers the log files to an Amazon S3 bucket that you specify. For more information, see Logging Amazon Glacier API Calls with AWS CloudTrail.	
Updates to Java samples	Updated the Java code examples in this guide that use the AWS SDK for Java.	June 27, 2014
Limiting vault inventory retrieval	You can now limit the number of vault inventory items retrieved by filtering on the archive creation date or by setting a limit. For more information about limiting inventory retrieval, see Range Inventory Retrieval in the Initiate Job (POST jobs) topic.	December 31, 2013
Removed outdated URLs	Removed the URLs that pointed to the old security credentials page from code examples.	July 26, 2013
Support for range retrievals	Amazon Glacier now supports retrieval of specific ranges of your archives. You can initiate a job requesting Amazon Glacier to prepare an entire archive or a portion of the archive for subsequent download. When an archive is very large, you may find it cost effective to initiate several sequential jobs to prepare your archive. For more information, see Downloading an Archive in Amazon Glacier.	November 13, 2012

Change	Description	Release Date
New Guide	This is the first release of the *Amazon Glacier Developer Guide*.	August 20, 2012

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.