
Amazon Glacier

Developer Guide

API Version 2012-06-01



Amazon Glacier: Developer Guide

Copyright © 2016 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is Amazon Glacier?	1
Are You a First-Time Amazon Glacier User?	1
Data Model	2
Vault	3
Archive	3
Job	4
Notification Configuration	4
Supported Operations	5
Vault Operations	5
Archive Operations	5
Jobs	5
Accessing Amazon Glacier	5
Regions and Endpoints	6
Getting Started	7
Step 1: Before You Begin	8
Step 1 of 6: Sign Up	8
Download the Appropriate AWS SDK	8
Step 2: Create a Vault	9
Step 3: Upload an Archive to a Vault	10
Upload an Archive Using Java	11
Upload an Archive Using .NET	12
Step 4: Download an Archive from a Vault	13
Download an Archive Using Java	14
Download an Archive Using .NET	15
Step 5: Delete an Archive from a Vault	18
Related Sections	18
Delete an Archive Using Java	18
Delete an Archive Using .NET	19
Step 6: Delete a Vault	20
Where Do I Go From Here?	21
Working with Vaults	22
Vault Operations in Amazon Glacier	22
Creating and Deleting Vaults	22
Retrieving Vault Metadata	23
Downloading a Vault Inventory	23
Configuring Vault Notifications	23
Creating a Vault	24
Creating a Vault Using Java	24
Creating a Vault Using .NET	27
Creating a Vault Using REST	30
Creating a Vault Using the Console	30
Retrieving Vault Metadata	30
Retrieving Vault Metadata Using Java	31
Retrieving Vault Metadata Using .NET	32
Retrieving Vault Metadata Using REST	34
Downloading a Vault Inventory	34
About the Inventory	35
Downloading a Vault Inventory Using Java	36
Downloading a Vault Inventory Using .NET	41
Downloading a Vault Inventory Using REST	47
Configuring Vault Notifications	47
General Concepts	47
Configuring Vault Notifications Using Java	48
Configuring Vault Notifications Using .NET	51
Configuring Vault Notifications Using the REST API	53

Configuring Vault Notifications Using the Console	53
Deleting a Vault	56
Deleting a Vault Using Java	56
Deleting a Vault Using .NET	57
Deleting a Vault Using REST	58
Deleting a Vault Using the Console	58
Tagging Vaults	58
Tagging Vaults Using the Amazon Glacier Console	59
Tagging Vaults Using the Amazon Glacier API	59
Related Sections	59
Vault Lock	60
Vault Locking Overview	60
Vault Locking Using the API	60
Working with Archives	62
Archive Operations	62
Uploading an Archive	63
Downloading an Archive	63
Deleting an Archive	63
Updating an Archive	63
Maintaining Client-Side Archive Metadata	63
Uploading an Archive	64
Options for Uploading an Archive	64
Uploading an Archive in a Single Operation	64
Uploading Large Archives in Parts	70
Downloading an Archive	79
Retrieving Archives	79
Downloading an Archive Using Java	82
Downloading an Archive Using .NET	94
Downloading an Archive Using REST	107
Deleting an Archive	107
Deleting an Archive Using Java	108
Deleting an Archive Using .NET	109
Deleting an Archive Using REST	111
Using the AWS SDKs	112
AWS SDKs that Support Amazon Glacier	112
AWS SDK Libraries for Java and .NET	113
What Is the Low-Level API?	113
What Is the High-Level API?	113
When to Use the High-Level and Low-Level API	113
Using the AWS SDK for Java	113
Using the Low-Level API	114
Using the High-Level API	114
Running Java Examples Using Eclipse	115
Setting the Endpoint	115
Using the AWS SDK for .NET	116
Using the Low-Level API	116
Using the High-Level API	117
Running .NET Examples	117
Setting the Endpoint	118
Authentication and Access Control	119
Authentication	119
Access Control	120
Overview of Managing Access	121
Amazon Glacier Resources and Operations	121
Understanding Resource Ownership	121
Managing Access to Resources	122
Specifying Policy Elements: Actions, Effects, Resources, and Principals	124
Specifying Conditions in a Policy	125

Using Identity-Based Policies (IAM Policies)	125
Permissions Required to Use the Amazon Glacier Console	126
AWS Managed Policies (Predefined Policies) for Amazon Glacier	127
Customer Managed Policy Examples	127
Using Resource-Based Policies (Vault Policies)	129
Vault Access Policies	130
Vault Lock Policies	132
Amazon Glacier API Permissions Reference	134
Data Retrieval Policies	140
Choosing an Amazon Glacier Data Retrieval Policy	140
Free Tier Only Policy	141
Max Retrieval Rate Policy	141
No Retrieval Limit Policy	141
Using the Amazon Glacier Console to Set Up a Data Retrieval Policy	141
Using the Amazon Glacier API to Set Up a Data Retrieval Policy	142
Using the Amazon Glacier REST API to Set Up a Data Retrieval Policy	142
Using the AWS SDKs to Set Up a Data Retrieval Policy	143
Tagging Resources	144
Tagging Basics	144
Tag Restrictions	145
Tracking Costs Using Tagging	145
Managing Access Control with Tagging	145
Related Sections	145
Audit Logging with AWS CloudTrail	146
Amazon Glacier Information in CloudTrail	146
Understanding Amazon Glacier Log File Entries	147
API Reference	150
Common Request Headers	150
Common Response Headers	153
Signing Requests	153
Example Signature Calculation	154
Calculating Signatures for the Streaming Operations	155
Computing Checksums	157
Tree Hash Example 1: Uploading an archive in a single request	158
Tree Hash Example 2: Uploading an archive using a multipart upload	158
Computing the Tree Hash of a File	159
Receiving Checksums When Downloading Data	162
Error Responses	163
Example 1: Describe Job request with a job ID that does not exist	165
Example 2: List Jobs request with an invalid value for the request parameter	166
Vault Operations	166
Abort Vault Lock	167
Add Tags To Vault	169
Create Vault	171
Complete Vault Lock	173
Delete Vault	175
Delete Vault Access Policy	177
Delete Vault Notifications	179
Describe Vault	181
Get Vault Access Policy	183
Get Vault Lock	186
Get Vault Notifications	189
Initiate Vault Lock	191
List Tags For Vault	194
List Vaults	196
Remove Tags From Vault	201
Set Vault Access Policy	203
Set Vault Notification Configuration	205

Archive Operations	208
Delete Archive	208
Upload Archive	210
Multipart Upload Operations	214
Abort Multipart Upload	214
Complete Multipart Upload	216
Initiate Multipart Upload	219
List Parts	222
List Multipart Uploads	227
Upload Part	232
Job Operations	236
Describe Job	236
Get Job Output	243
Initiate Job	249
List Jobs	257
Data Retrieval Operations	266
Get Data Retrieval Policy	266
List Provision Capacity	269
Purchase Provisioned Capacity	271
Set Data Retrieval Policy	273
Document History	277
AWS Glossary	279

What Is Amazon Glacier?

Welcome to the *Amazon Glacier Developer Guide*. Amazon Glacier is a storage service optimized for infrequently used data, or "cold data."

Amazon Glacier is an extremely low-cost storage service that provides durable storage with security features for data archiving and backup. With Amazon Glacier, customers can store their data cost effectively for months, years, or even decades. Amazon Glacier enables customers to offload the administrative burdens of operating and scaling storage to AWS, so they don't have to worry about capacity planning, hardware provisioning, data replication, hardware failure detection and recovery, or time-consuming hardware migrations. For more service highlights and pricing information, go to the [Amazon Glacier detail page](#).

Amazon Glacier is a great storage choice when low storage cost is paramount, your data is rarely retrieved, and retrieval latency of several hours is acceptable. If your application requires fast or frequent access to your data, consider using Amazon S3. For more information, go to [Amazon Simple Storage Service \(Amazon S3\)](#).

Topics

- [Are You a First-Time Amazon Glacier User? \(p. 1\)](#)
- [Amazon Glacier Data Model \(p. 2\)](#)
- [Supported Operations in Amazon Glacier \(p. 5\)](#)
- [Accessing Amazon Glacier \(p. 5\)](#)

Are You a First-Time Amazon Glacier User?

If you are a first-time user of Amazon Glacier, 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 \(p. 7\)](#) 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 provides a management console, which you can use 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, by using either the REST API directly or by using the AWS 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](#).

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 using the REST API and the AWS Software Development Kits (SDKs) for Java and Microsoft .NET:

- [Using the AWS SDKs with Amazon Glacier \(p. 112\)](#)

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 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 \(p. 22\)](#)

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 Amazon Glacier console, you can use AWS SDKs for various vault operations. This section describes the API and provides working samples using the AWS SDK for Java and .NET.

- [Working with Archives in Amazon Glacier \(p. 62\)](#)

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 creating jobs to download archives asynchronously. The section provides examples using the AWS SDK for Java and .NET.

- [API Reference for Amazon Glacier \(p. 150\)](#)

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

Amazon Simple Storage Service (Amazon S3) supports lifecycle configuration on a bucket that enables you to optionally transition objects to Amazon Glacier for archival purposes. When you transition Amazon S3 objects to the Glacier storage class, Amazon S3 internally uses Amazon Glacier for durable storage at lower cost. For more information about lifecycle configuration and transitioning objects to Amazon Glacier, go to [Object Lifecycle Management](#) and [Object Archival](#) in the *Amazon Simple Storage Service Developer Guide*.

Amazon Glacier Data Model

The Amazon Glacier data model core concepts 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 \(p. 3\)](#)
- [Archive \(p. 3\)](#)
- [Job \(p. 4\)](#)
- [Notification Configuration \(p. 4\)](#)

Vault

In Amazon Glacier, a vault is a container for storing archives. When you create a vault, you specify a name and select 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/<vaultname>
```

For example, suppose you create a vault (`examplevault`) in the US West (Oregon) Region. This vault can then be addressed by the following URI:

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

In the URI,

- `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 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 \(p. 5\)](#). 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. Note that 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 only includes vaults created in that specific region.

Archive

An archive can be any data such as a photo, video, or document and is a base unit of storage in Amazon Glacier. Each archive has a unique ID and an optional description. Note that you can only specify the optional description during the upload of an archive. Amazon Glacier assigns the archive an ID, which is unique in the AWS region in which it is stored.

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

```
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:

```
https://glacier.us-west-2.amazonaws.com/111122223333/vaults/  
examplevault/archives/NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-  
TjhqG6eGoOY9Z8i1_AUyUshPAdTqLHy8pT15nfCFJmD12yEZONi5L26Omw12vcs01MNGntHEQL8MBfG1qrEXAMPLEA
```

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

In addition, the Amazon Glacier data model includes job and notification-configuration resources. These resources complement the core vault and archive resources.

Job

Retrieving an archive and vault inventory (list 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. With Amazon Glacier, your data retrieval requests are queued and most jobs take about four hours to complete.

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. The archive retrieval job requires both the vault name where the archive resides and the archive ID you wish to download. You can also provide an optional job description when you initiate these jobs. These descriptions can help you in identifying jobs.

Both the 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.

```
https://glacier.us-west-2.amazonaws.com/111122223333/vaults/examplevault/  
jobs/HkF9p6o7yjhFx-  
K3CGl6fuSm6VzW9T7esGQfco8nUXVYws0j1b5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

For each job, Amazon Glacier maintains information such as 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.

After Amazon Glacier completes a job, you can download the job output. You can download all the job output or optionally download only a portion of the output by specifying a byte range.

Notification Configuration

Because jobs take time to complete, Amazon Glacier supports a notification mechanism to notify you when a job is complete. You can configure a vault to send notification to an Amazon Simple Notification Service (Amazon SNS) topic when jobs complete. You can specify one 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"]  
}
```

Note that 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 complete.

Supported Operations in Amazon Glacier

To work with vaults and archives (see [Amazon Glacier Data Model \(p. 2\)](#)), 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 a region. The vault description provides information such as creation date, number of archives in the vault, total size in bytes used by all the archives in the vault, and the date 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 \(p. 22\)](#).

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. Note that each time you upload an archive, Amazon Glacier generates a new archive ID. For more information, see [Working with Archives in Amazon Glacier \(p. 62\)](#).

Jobs

Retrieving an archive or vault inventory from Amazon Glacier is an asynchronous operation. It requires you to first initiate a job, wait for the job to complete and then download the job output. Amazon Glacier provides operations for you to initiate a job, get job description, and retrieve a list of jobs associated with a vault. Note that most jobs take about four hours to complete. Amazon Glacier can post a message to an Amazon Simple Notification Service (Amazon SNS) topic upon job completion. For more information about retrieving an archive or a vault inventory, see [Downloading an Archive in Amazon Glacier \(p. 79\)](#) and [Downloading a Vault Inventory in Amazon Glacier \(p. 34\)](#).

Accessing Amazon Glacier

Amazon Glacier is a RESTful web service that uses HTTP and HTTPS as a transport 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 \(p. 150\)](#).

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 \(p. 112\)](#).

Amazon Glacier also provides a management console. You can use the console to create and delete vaults. However, all the archive and job operations require you to write code and make requests 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 regions. You always send your Amazon Glacier requests to a region-specific endpoint. For a list of the AWS regions supported by Amazon Glacier, go to [Regions and Endpoints](#) in the *AWS General Reference*.

Getting Started with Amazon Glacier

In Amazon Glacier, 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 the vaults and archives resources described in the [Amazon Glacier Data Model \(p. 2\)](#) section.

In the getting started exercise, you will create a vault, upload and download an archive, and finally 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 AWS Software Development Kits (SDKs) for Java and .NET high-level API. The high-level API provides a simplified programming experience when working with Amazon Glacier. For more information about these APIs, see [Using the AWS SDKs with Amazon Glacier \(p. 112\)](#).

Important

Amazon Glacier provides a management console. You can use the console to create and delete vaults as shown in this getting started exercise. 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 AWS 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](#).

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 provides steps where you can learn more about the developer experience with Amazon Glacier.

Topics

- [Step 1: Before You Begin with Amazon Glacier \(p. 8\)](#)
- [Step 2: Create a Vault in Amazon Glacier \(p. 9\)](#)
- [Step 3: Upload an Archive to a Vault in Amazon Glacier \(p. 10\)](#)
- [Step 4: Download an Archive from a Vault in Amazon Glacier \(p. 13\)](#)
- [Step 5: Delete an Archive from a Vault in Amazon Glacier \(p. 18\)](#)
- [Step 6: Delete a Vault in Amazon Glacier \(p. 20\)](#)
- [Where Do I Go From Here? \(p. 21\)](#)

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 Software Development Kits (SDKs). The following sections provide instructions.

Important

Amazon Glacier provides a management console, which you can use 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 AWS 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](#).

Sign Up

If you already have an AWS account, go ahead and skip to the next section: [Download the Appropriate AWS SDK \(p. 8\)](#). Otherwise, follow these steps.

To sign up for an AWS account

1. Open <https://aws.amazon.com/>, and then choose **Create an AWS Account**.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Download the Appropriate AWS SDK

To try the getting started exercise, you must decide which programming language you want to use and 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 using the update site <http://aws.amazon.com/eclipse/>. For more information, go to [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. Go to <http://aws.amazon.com/sdkfornet> and click **Download AWS SDK for .NET**. By default, the installation script installs both the AWS SDK and the AWS Toolkit for Visual Studio. To learn more about the toolkit, go to [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. In this getting started exercise, you create a vault in the US West (Oregon) region. For a list of the AWS regions supported by Amazon Glacier, go to [Regions and Endpoints](#) in the *AWS General Reference*.

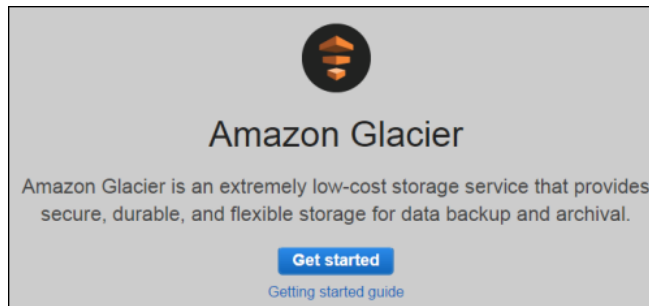
You can create vaults programmatically or by using the Amazon Glacier console. This section uses the console to create a vault. In a later step, you will upload an archive to the vault.

To create a vault

1. Sign into the AWS Management Console and open the Amazon Glacier console at <https://console.aws.amazon.com/glacier/>.
2. Select a region from the region selector.

In this getting started exercise, we use the US West (Oregon) region.

3. If you are using Amazon Glacier for the first time, click **Get started**. (Otherwise, you would click **Create Vault**.)



4. Enter `examplevault` as the vault name in the **Vault Name** field and then click **Next Step**.

There are guidelines for naming a vault. For more information, see [Creating a Vault in Amazon Glacier](#) (p. 24).

A screenshot of the 'Welcome to Amazon Glacier' dialog box. The dialog box has a title bar that says 'Welcome to Amazon Glacier'. Below the title bar, there is a paragraph of text: 'Data is stored in Amazon Glacier in "archives." An archive can be any data such as a photo, video, or document. You can upload a single file as an archive or aggregate multiple files into a TAR or ZIP file and upload as one archive.' Below that, another paragraph: 'A single archive can be as large as 40 terabytes. You can store an unlimited number of archives and an unlimited amount of data in Amazon Glacier. Each archive is assigned a unique archive ID at the time of creation, and the content of the archive is immutable, meaning that after an archive is created it cannot be updated.' Below that, a third paragraph: 'Vaults allow you to organize your archives and set access policies and notification policies. Get started by giving your vault a name. You can then create your vault now or click **Next Step** to set up your vault's properties.' At the bottom of the dialog box, there are two input fields. The first is labeled 'Region*' and contains the text 'us-west-2'. The second is labeled 'Vault Name*' and contains the text 'examplevault'. To the right of the 'Region*' field is an information icon. At the bottom right of the dialog box, there are two buttons: 'Cancel' and 'Next Step'.

5. Select **Do not enable notifications**. For this getting started exercise, you will not configure notifications for the vault.

If you wanted to have notifications sent to you or your application whenever certain Amazon Glacier jobs complete, you would select **Enable notifications and create a new SNS topic** or **Enable notifications and use an existing SNS topic** to set up Amazon Simple Notification Service (Amazon SNS) notifications. In subsequent steps, you upload an archive and then download it using the high-level API of the AWS SDK. Using the high-level API does not require that you configure vault notification to retrieve your data.

Set Event Notifications

You can choose to have notifications sent to you or your application whenever certain Amazon Glacier jobs complete. Notifications are sent using the Amazon Simple Notifications Service (SNS). To use Amazon SNS, you first need to specify a topic that applications or people can subscribe to. You can then select specific jobs that, on completion, will trigger the notifications. Notifications can be delivered over the protocol of your choice (HTTP, email, etc.).

- Do not enable notifications**
You can enable, set up, and change your notification settings later.
- Enable notifications and create a new SNS topic**
Enable notifications and create a new Amazon SNS topic to send the notifications.
- Enable notifications and use an existing SNS topic**
Enable notifications and enter an existing SNS topic to send the notifications.

Cancel Previous **Next Step**

6. If the region and vault name are correct, then click **Submit**.

Review

Make sure the following information is correct before you choose **Submit**. To go back and make changes, choose **Previous**.

Region us-west-2

Vault Name examplevault

Cancel Previous **Submit**

7. Your new vault is listed on the **Amazon Glacier Vaults** page.

Name	Inventory Last Updated	Size (as of last inventory)	# of Archives (as of last inventory)
examplevault	Not updated yet	--	--

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

In this step, you upload a sample archive to the vault you created in the preceding step (see [Step 2: Create a Vault in Amazon Glacier \(p. 9\)](#)). Depending on the development platform you are using, click one of the links at the end of this section.

Important

Any archive operation, such as upload, download, or deletion, requires that you 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, using either the REST API directly or by using the AWS 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](#).

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. You can upload an archive in a single request. For large archives, Amazon Glacier provides a multipart upload API 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 \(Multipart Upload\)](#) (p. 70).

Topics

- [Upload an Archive to a Vault in Amazon Glacier Using the AWS SDK for Java](#) (p. 11)
- [Upload an Archive to a Vault in Amazon Glacier Using the AWS SDK for .NET](#) (p. 12)

Upload an Archive to a Vault in Amazon Glacier 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` method 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`) to match the location where you created the vault previously in [Step 2: Create a Vault in Amazon Glacier](#) (p. 9).

For step-by-step instructions on how to run this example, see [Running Java Examples for Amazon Glacier Using Eclipse](#) (p. 115). You need to update the code as shown with the name of the archive file 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.

Example — Uploading an Archive Using the AWS SDK for Java

```
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 AmazonGlacierUploadArchive_GettingStarted {

    public static String vaultName = "examplevault2";
    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);
        }
    }
}
```

Upload an Archive to a Vault in Amazon Glacier 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`) to match the location where you created the vault previously in [Step 2: Create a Vault in Amazon Glacier \(p. 9\)](#).
- The example uses the `Upload` method of the `ArchiveTransferManager` class to upload your archive. For small archives, this method uploads the archive directly to Amazon Glacier. For larger archives, this method uses Amazon Glacier's multipart upload API 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](#) (p. 117). You need to 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, 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.

Example — Uploading an Archive 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 download the sample archive you uploaded previously in [Step 3: Upload an Archive to a Vault in Amazon Glacier](#) (p. 10).

Important

Any archive operation, such as upload, download, or deletion, requires that you 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, using either the REST API directly or the AWS SDKs. For more information about using Amazon Glacier with the AWS CLI, see [AWS CLI Reference for Amazon Glacier](#). To install the AWS CLI, see [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 completes, download the bytes of data.

To retrieve an archive from Amazon Glacier, you first initiate a job. After the job completes, you download the data. For more information about archive retrievals, see [Retrieving Amazon Glacier Archives](#) (p. 79).

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

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

Topics

- [Download an Archive from a Vault in Amazon Glacier Using the AWS SDK for Java](#) (p. 14)
- [Download an Archive from a Vault in Amazon Glacier Using the AWS SDK for .NET](#) (p. 15)

Download an Archive from a Vault in Amazon Glacier 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 you uploaded in the previous step. In the code example, note the following:

- The example creates an instance of the `AmazonGlacierClient` class.
- The example uses the `download` method 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`) to match the location where you created the vault in [Step 2: Create a Vault in Amazon Glacier](#) (p. 9).

For step-by-step instructions on how to run this example, see [Running Java Examples for Amazon Glacier Using Eclipse](#) (p. 115). You need to update the code as shown with the archive ID of the file you uploaded in [Step 3: Upload an Archive to a Vault in Amazon Glacier](#) (p. 10).

Example — Downloading an Archive 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 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 you uploaded previously in [Upload an Archive to a Vault in Amazon Glacier Using the AWS SDK for .NET](#) (p. 12). 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 \(p. 9\)](#).
- The example uses the `Download` method of the `ArchiveTransferManager` class to upload your archive. The `Download` method creates an Amazon SNS topic, and an 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. This polling takes about 4 hours. Once the archive is available, download will begin.

For step-by-step instructions on how to run this example, see [Running Code Examples \(p. 117\)](#). You need to update the code as shown with the archive ID of the file you uploaded in [Step 3: Upload an Archive to a Vault in Amazon Glacier \(p. 10\)](#).

Example — Download an Archive 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("This polling takes about 4 hours. 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 delete the sample archive you uploaded in [Step 3: Upload an Archive to a Vault in Amazon Glacier](#) (p. 10).

Important

You cannot delete an archive using the Amazon Glacier console. Any archive operation, such as upload, download, or deletion, requires that you use the AWS Config (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 AWS 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](#).

Depending on which SDK you are using, delete the sample archive by following one of these steps:

- [Delete an Archive from a Vault in Amazon Glacier Using the AWS SDK for Java](#) (p. 18)
- [Delete an Archive from a Vault in Amazon Glacier Using the AWS SDK for .NET](#) (p. 19)

Related Sections

- [Step 3: Upload an Archive to a Vault in Amazon Glacier](#) (p. 10)
- [Deleting an Archive in Amazon Glacier](#) (p. 107)

Delete an Archive from a Vault in Amazon Glacier 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` method sends the request to Amazon Glacier to delete the archive.
- The example 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](#) (p. 9).

For step-by-step instructions on how to run this example, see [Running Java Examples for Amazon Glacier Using Eclipse](#) (p. 115). You need to update the code as shown with the archive ID of the file you uploaded in [Step 3: Upload an Archive to a Vault in Amazon Glacier](#) (p. 10).

Example — Deleting an Archive 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 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 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`) to match the location where you created the vault previously in [Step 2: Create a Vault in Amazon Glacier \(p. 9\)](#).
- The example uses the `Delete` method of the `ArchiveTransferManager` class 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 \(p. 117\)](#). You need to update the code as shown with the archive ID of the file you uploaded in [Step 3: Upload an Archive to a Vault in Amazon Glacier \(p. 10\)](#).

Example — Deleting an Archive 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();
        }
    }
}
```

Step 6: Delete a Vault in Amazon Glacier

A vault is a container for storing archives. You can delete an Amazon Glacier vault only if there are no archives in the vault as of the last inventory that Amazon Glacier computed and 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 the vault is indeed empty by checking if there were any write operations since the last vault inventory.

You can delete a vault programmatically or by using the Amazon Glacier console. This section uses the console to delete a vault. For information about deleting a vault programmatically, see [Deleting a Vault in Amazon Glacier \(p. 56\)](#).

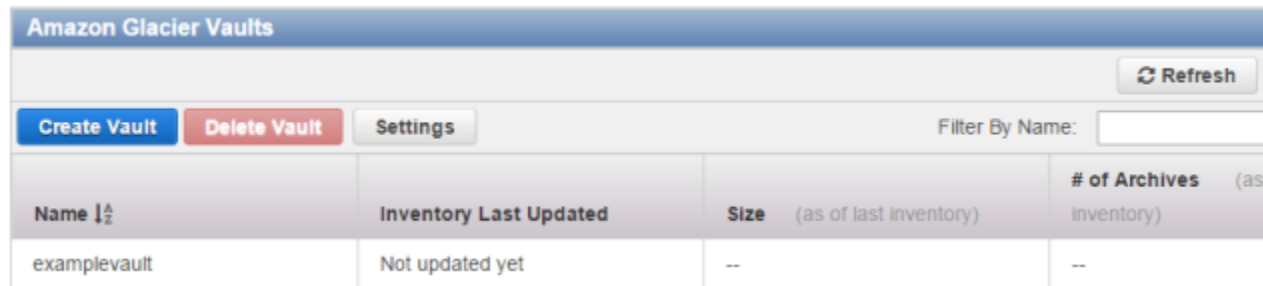
To delete a vault

1. Sign into the AWS Management Console and open the Amazon Glacier console at <https://console.aws.amazon.com/glacier>.
2. From the region selector, select the AWS region where the vault exists that you want to delete.

In this getting started exercise, we use the US West (Oregon) region.

3. Select the vault that you want to delete.

In this getting started exercise, we've been using a vault named `examplevault`.



Amazon Glacier Vaults			
Create Vault		Delete Vault	Settings
Name	Inventory Last Updated	Size (as of last inventory)	# of Archives (as of last inventory)
examplevault	Not updated yet	--	--

4. Click **Delete Vault**.

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 \(p. 22\)](#)
- [Working with Archives in Amazon Glacier \(p. 62\)](#)

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 a region in which you want to create the vault. For a list of supported regions, see [Accessing Amazon Glacier \(p. 5\)](#).

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

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 AWS 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

- [Vault Operations in Amazon Glacier \(p. 22\)](#)
- [Creating a Vault in Amazon Glacier \(p. 24\)](#)
- [Retrieving Vault Metadata in Amazon Glacier \(p. 30\)](#)
- [Downloading a Vault Inventory in Amazon Glacier \(p. 34\)](#)
- [Configuring Vault Notifications in Amazon Glacier \(p. 47\)](#)
- [Deleting a Vault in Amazon Glacier \(p. 56\)](#)
- [Tagging Your Amazon Glacier Vaults \(p. 58\)](#)
- [Amazon Glacier Vault Lock \(p. 60\)](#)

Vault Operations in Amazon Glacier

Amazon Glacier supports various vault operations. Note that vault operations are region specific. For example, when you create a vault, you create it in a specific region. When you list vaults, Amazon Glacier returns the vault list from the region you specified in the request.

Creating and Deleting Vaults

An AWS account can create up to 1,000 vaults per region. For a list of the AWS regions supported by Amazon Glacier, see [Regions and Endpoints](#) 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 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 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 \(p. 24\)](#) and [Deleting a Vault in Amazon Glacier \(p. 56\)](#).

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 region in your account. For more information, see [Retrieving Vault Metadata in Amazon Glacier \(p. 30\)](#).

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 archive ID, creation date, and size. Amazon Glacier updates the vault inventory approximately once a day, starting on the day 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 completes, 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 completes. 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 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 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 in your database with the actual vault inventory.

For more information about retrieving a vault inventory, see [Downloading a Vault Inventory in Amazon Glacier \(p. 34\)](#).

Configuring Vault Notifications

Retrieving anything from Amazon Glacier, such as an archive from a vault or a vault inventory, is a two-step process in which you first initiate a job. After the job completes, you can download the output. You can use Amazon Glacier notifications support to know when your job is complete. 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 \(p. 47\)](#).

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 region. For a list of the AWS regions supported by Amazon Glacier, go to [Regions and Endpoints](#) in the *AWS General Reference*. For information on creating more vaults, go to the [Amazon Glacier product detail page](#).

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 region in which the vault is being created. That is, an account can create vaults with the same name in different regions but not in the same region.

Topics

- [Creating a Vault in Amazon Glacier Using the AWS SDK for Java](#) (p. 24)
- [Creating a Vault in Amazon Glacier Using the AWS SDK for .NET](#) (p. 27)
- [Creating a Vault in Amazon Glacier Using the REST API](#) (p. 30)
- [Creating a Vault Using the Amazon Glacier Console](#) (p. 30)

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 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 region.

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

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](#) (p. 113).

3. Execute 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 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 \(PUT vault\) \(p. 171\)](#).

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 regions, see [Accessing Amazon Glacier \(p. 5\)](#)). In addition, the code example retrieves the vault information, lists all vaults in the same 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 \(p. 115\)](#).

```
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 = "examplevaultfordelate";

        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 \(p. 112\)](#) provided by the AWS 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 \(p. 27\)](#)
- [Creating a Vault Using the Low-Level API of the AWS SDK for .NET \(p. 28\)](#)

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 \(p. 5\)](#).

For step-by-step instructions on how to run the following example, see [Running Code Examples \(p. 117\)](#). You need to update the code as shown with a vault name.

```
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 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 region.

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

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 \(p. 116\)](#).

3. Execute 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 region, and then deletes the vault created. The `Location` printed is the relative URI of the vault that includes your account ID, the region, and the vault name.

Note

For information about the underlying REST API, see [Create Vault \(PUT vault\) \(p. 171\)](#).

For step-by-step instructions on how to run the following example, see [Running Code Examples \(p. 117\)](#). You need to update the code as shown with a vault name.

```
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.");  
                }  
            }  
            catch { }  
        }  
    }  
}
```

```
        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 \(PUT vault\) \(p. 171\)](#).

Creating a Vault Using the Amazon Glacier Console

To create a vault using the Amazon Glacier console, see [Step 2: Create a Vault in Amazon Glacier \(p. 9\)](#) in the *Getting Started* tutorial.

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 provides API calls for you to retrieve this information for a specific vault or all the vaults in a specific 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 \(p. 31\)](#)
- [Retrieving Vault Metadata in Amazon Glacier Using the AWS SDK for .NET \(p. 32\)](#)
- [Retrieving Vault Metadata Using the REST API \(p. 34\)](#)

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

Topics

- [Retrieve Vault Metadata for a Vault \(p. 31\)](#)
- [Retrieve Vault Metadata for All Vaults in a Region \(p. 31\)](#)
- [Example: Retrieving Vault Metadata Using the AWS SDK for Java \(p. 32\)](#)

Retrieve Vault Metadata for a Vault

You can retrieve metadata for a specific vault or all the vaults in a specific region. The following are the steps to retrieve vault metadata for a specific 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 region.

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

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 \(p. 113\)](#).

3. Execute 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 \(GET vault\) \(p. 181\)](#).

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 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 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 1,000 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 \(GET vaults\)](#) (p. 196)).

Example: Retrieving Vault Metadata Using the AWS SDK for Java

For a working code example, see [Example: Creating a Vault Using the AWS SDK for Java](#) (p. 25). 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](#) (p. 32)
- [Retrieve Vault Metadata for All Vaults in a Region](#) (p. 33)
- [Example: Retrieving Vault Metadata Using the Low-Level API of the AWS SDK for .NET](#) (p. 34)

Retrieve Vault Metadata for a Vault

You can retrieve metadata for a specific vault or all the vaults in a specific 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 region.

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

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 \(p. 116\)](#).

3. Execute 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 \(GET vault\) \(p. 181\)](#).

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 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 region.

```
AmazonGlacierClient client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USWest2);
string lastMarker = null;
Console.WriteLine("\n List of vaults in your account in the specific
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 1,000 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 \(GET vaults\)](#) (p. 196)).

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](#) (p. 28). 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 \(GET vaults\)](#) (p. 196). To describe one vault, see [Describe Vault \(GET vault\)](#) (p. 181).

Downloading a Vault Inventory in Amazon Glacier

After you upload your first archive to your vault, 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 a retrieval job.

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](#) (p. 140). For more information about the `PolicyEnforcedException` exception, see [Error Responses](#) (p. 163).

2. After the job completes, download the bytes.

For example, retrieving an archive or a vault inventory requires you to first initiate a retrieval job. The job request is executed 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 \(p. 47\)](#)). 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 \(GET JobID\) \(p. 236\)](#)) 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 \(p. 35\)](#)
- [Downloading a Vault Inventory in Amazon Glacier Using the AWS SDK for Java \(p. 36\)](#)
- [Downloading a Vault Inventory in Amazon Glacier Using the AWS SDK for .NET \(p. 41\)](#)
- [Downloading a Vault Inventory Using the REST API \(p. 47\)](#)

About the 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. 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 limit. For more information about limiting inventory retrieval, see [Range Inventory Retrieval \(p. 250\)](#).

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 \(p. 246\)](#) 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 region.

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

Execute `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 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.

Most Amazon Glacier jobs take about four hours 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 \(p. 113\)](#).

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 \(p. 236\)](#).

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

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

Note

It takes about four hours for most jobs to complete.

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.File;
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 region.

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

You provide job information in an `InitiateJobRequest` object. 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.

Most Amazon Glacier jobs take about four hours 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 \(p. 116\)](#).

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 \(p. 236\)](#).

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.

Caution

Note that it takes about four hours for most jobs to complete.

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.

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

namespace glacier.amazon.com.docsamples
{
    class VaultInventoryJobLowLevelUsingSNSSQS
    {
        static string topicArn;
        static string queueUrl;
        static string queueArn;
        static string vaultName = "**** Provide vault name ****";
        static string fileName = "**** Provide file name and path where to store
inventory ****";
        static AmazonSimpleNotificationServiceClient snsClient;
        static AmazonSQSClient sqsClient;
        const string SQS_POLICY =
            "{" +
            "    \"Version\" : \"2012-10-17\", " +
            "    \"Statement\" : [ " +
            "        { " +
            "            \"Sid\" : \"sns-rule\", " +
            "            \"Effect\" : \"Allow\", " +
            "            \"Principal\" : \"*\", " +
            "            \"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.WriteLine("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 \(POST jobs\)](#) (p. 249).
2. After the job completes, download the inventory data. For more information, see [Get Job Output \(GET output\)](#) (p. 243).

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 completes, download the job output.

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

Topics

- [Configuring Vault Notifications in Amazon Glacier: General Concepts](#) (p. 47)
- [Configuring Vault Notifications in Amazon Glacier Using the AWS SDK for Java](#) (p. 48)
- [Configuring Vault Notifications in Amazon Glacier Using the AWS SDK for .NET](#) (p. 51)
- [Configuring Vault Notifications in Amazon Glacier Using the REST API](#) (p. 53)
- [Configuring Vault Notifications Using the Amazon Glacier Console](#) (p. 53)

Configuring Vault Notifications in Amazon Glacier: General Concepts

An Amazon Glacier retrieval job request is executed 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 completes, it can post a message to an Amazon Simple Notification Service (Amazon SNS) topic. This requires you to set 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 completes, Amazon Glacier publishes a notification message to the SNS topic.

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

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

Note that 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 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 that the Amazon SNS topic must allow the vault to publish a notification. By default, only the SNS topic owner can publish a message to the topic. However, if the SNS topic and the vault are owned by different AWS accounts, then you must configure the SNS topic to accept publications from the vault. You can configure the 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 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 \(PUT notification-configuration\)](#) (p. 205).

3. Execute 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 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](#) (p. 166).

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](#) (p. 113).

```
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 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 \(p. 116\)](#).

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 \(PUT notification-configuration\) \(p. 205\)](#).

3. Execute 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 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 \(p. 166\)](#).

For step-by-step instructions to run the following example, see [Running Code Examples \(p. 117\)](#). You need to update the code as shown and provide an existing vault name and an Amazon SNS topic.

```
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

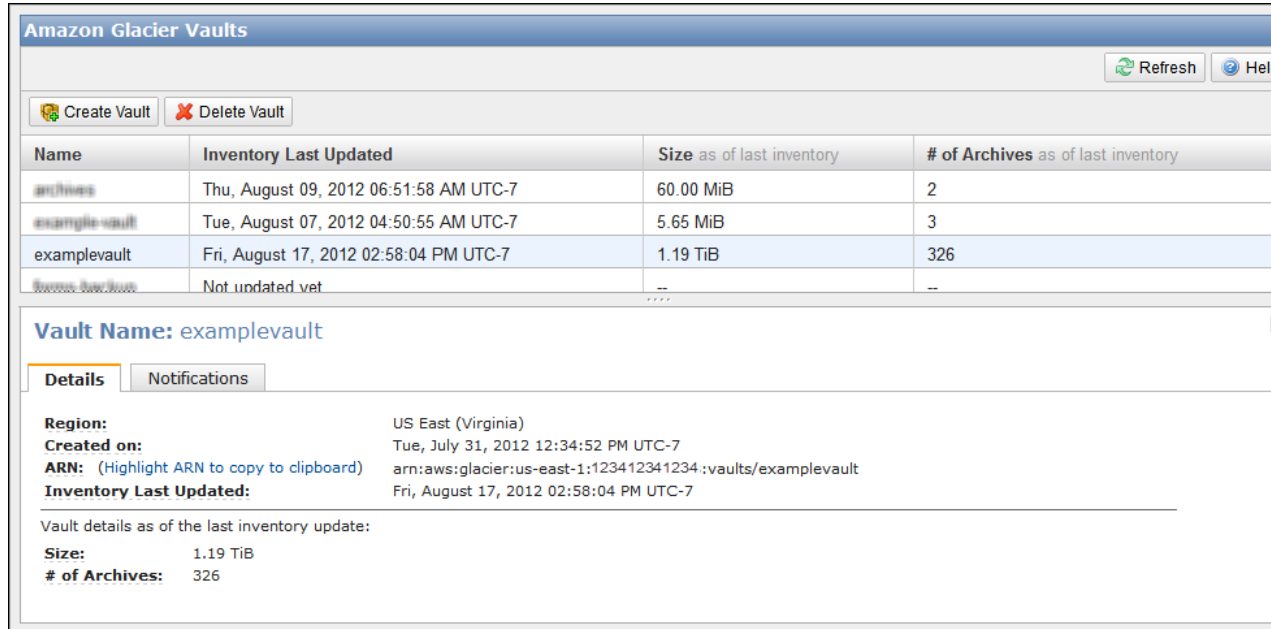
To configure vault notifications using the REST API, see [Set Vault Notification Configuration \(PUT notification-configuration\)](#) (p. 205). Additionally, you can also get vault notifications ([Get Vault Notifications \(GET notification-configuration\)](#) (p. 189)) and delete vault notifications ([Delete Vault Notifications \(DELETE notification-configuration\)](#) (p. 179)).

Configuring Vault Notifications Using the Amazon Glacier Console

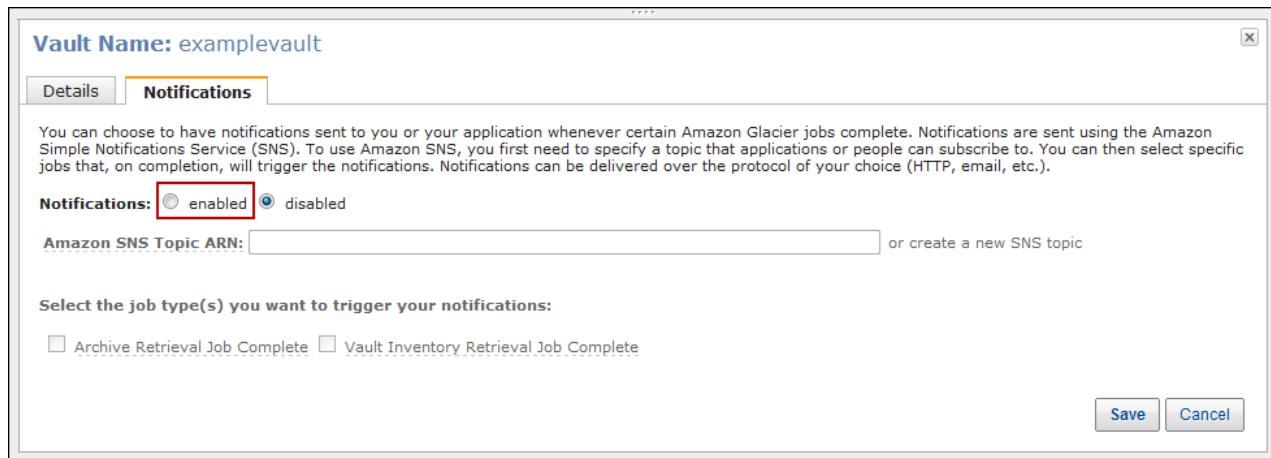
This section describes how to configure vault notifications using the Amazon Glacier console. When you configure notifications, you specify job completion events that trigger notification to an Amazon Simple Notification Service (Amazon SNS) topic. In addition to configuring notifications for the vault, 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.

To configure a vault notification

1. Sign into the AWS Management Console and open the Amazon Glacier console at <https://console.aws.amazon.com/glacier>.
2. Select a vault in the vault list.

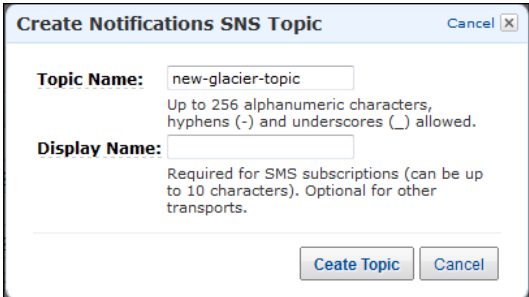


3. Select the **Notifications** tab.
4. Select the **enabled** in the **Notifications** field.



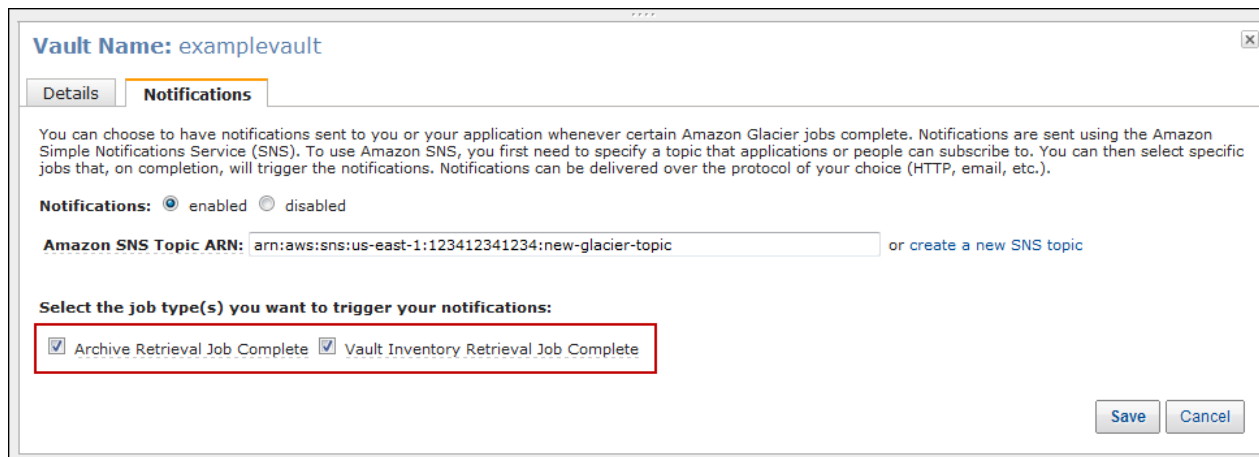
5. On the **Notifications** tab, do one of the following:

To...	Do this...
Specify an existing Amazon SNS topic	<p>Enter the Amazon SNS topic in the Amazon SNS Topic ARN text box.</p> <p>The topic is an Amazon Resource Name (ARN) that has the form shown below.</p> <pre>arn:aws:sns:region:accountId:topicname</pre>

To...	Do this...
	You can find the an Amazon SNS topic ARN from the Amazon Simple Notification Service (Amazon SNS) console.
Create a new Amazon SNS topic	<p>a. Click create a new SNS topic.</p> <p>A Create Notifications SNS Topic dialog box appears.</p> <p>b. In the Topic Name field, specify the name of the new topic.</p> <p>If you will subscribe to the topic using SMS subscriptions, put a name in the Display Name field.</p>  <p>c. Click Create Topic.</p> <p>The Amazon SNS Topic ARN text box is populated with the ARN of the new topic.</p>

6. Select the events that trigger notification.

For example, to trigger notification when only archive retrieval jobs are complete, check only **Get Archive Job Complete**.



7. Click **Save**.

Important

By default, a new topic does not have any subscriptions associated with it. To receive notifications published to this topic, you must subscribe to the topic. Follow the steps in

[Subscribe to a Topic](#) in the *Amazon Simple Notification Service Getting Started Guide* to subscribe to a new topic.

Deleting a Vault in 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 \(p. 107\)](#). For information about downloading a vault inventory, [Downloading a Vault Inventory in Amazon Glacier \(p. 34\)](#).

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.

Topics

- [Deleting a Vault in Amazon Glacier Using the AWS SDK for Java \(p. 56\)](#)
- [Deleting a Vault in Amazon Glacier Using the AWS SDK for .NET \(p. 57\)](#)
- [Deleting a Vault in Amazon Glacier Using the REST API \(p. 58\)](#)
- [Deleting a Vault Using the Amazon Glacier Console \(p. 58\)](#)

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 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 \(p. 113\)](#).

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

Amazon Glacier deletes the vault only if it is empty. For more information, see [Delete Vault \(DELETE vault\) \(p. 175\)](#).

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 \(DELETE vault\) \(p. 175\)](#).

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 \(p. 25\)](#). 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 \(p. 112\)](#) provided by the AWS 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 \(p. 57\)](#)
- [Deleting a Vault Using the Low-Level API of the AWS SDK for .NET \(p. 57\)](#)

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 \(p. 27\)](#). 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 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 \(p. 116\)](#).

3. Execute the `DeleteVault` method by providing the request object as a parameter.

Amazon Glacier deletes the vault only if it is empty. For more information, see [Delete Vault \(DELETE vault\) \(p. 175\)](#).

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 \(DELETE vault\) \(p. 175\)](#).

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 \(p. 28\)](#). 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 \(DELETE vault\) \(p. 175\)](#).

Deleting a Vault Using the Amazon Glacier Console

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 \(p. 107\)](#). For information about downloading a vault inventory, see [Downloading a Vault Inventory in Amazon Glacier \(p. 34\)](#).

The following are the steps to delete a vault using the Amazon Glacier console.

1. Sign into the AWS Management Console and open the Amazon Glacier console at <https://console.aws.amazon.com/glacier>.
2. From the region selector, select the AWS region where the vault exists.
3. Select the vault.
4. Click **Delete Vault**.

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 \(p. 144\)](#).

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

Topics

- [Tagging Vaults Using the Amazon Glacier Console \(p. 59\)](#)
- [Tagging Vaults Using the Amazon Glacier API \(p. 59\)](#)

- [Related Sections \(p. 59\)](#)

Tagging Vaults 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>.
2. From the region selector, choose a region.
3. On the **Amazon Glacier Vaults** page, choose a vault.
4. Choose the **Tags** tab. The tags for that vault will appear.

To add a tag to a vault

1. Open the Amazon Glacier console, and then choose a region from the region selector.
2. On the **Amazon Glacier Vaults** page, choose a vault.
3. Choose the **Tags** tab.
4. Specify the tag key in the **Key** field, optionally specify a tag value in the **Value** field, and then choose **Save**.

If the **Save** button is not enabled, either the tag key or the tag value that you specified does not meet the tag restrictions. For more about tag restrictions, see [Tag Restrictions \(p. 145\)](#).

To remove a tag from a vault

1. Open the Amazon Glacier console, and then choose a region from the region selector.
2. On the **Amazon Glacier Vaults** page, choose a vault.
3. Choose the **Tags** tab, and then choose the **x** at the end of the row that describes the tag you want to delete.
4. Choose **Delete**.

Tagging Vaults Using the Amazon Glacier API

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

[Add Tags To Vault \(POST tags add\) \(p. 169\)](#)

Adds or updates tags for the specified vault.

[List Tags For Vault \(GET tags\) \(p. 194\)](#)

Lists the tags for the specified vault.

[Remove Tags From Vault \(POST tags remove\) \(p. 201\)](#)

Removes tags from the specified vault.

Related Sections

- [Tagging Amazon Glacier Resources \(p. 144\)](#)

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 \(p. 60\)](#)
- [Locking a Vault by Using the Amazon Glacier API \(p. 60\)](#)

Vault Locking Overview

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. Once locked, the policy can no longer be changed.

Amazon Glacier enforces the controls set in the vault lock policy to help achieve your compliance objectives, for example, for data retention. You can deploy a variety of compliance controls in a vault lock policy using the AWS Identity and Access Management (IAM) policy language. For more information about vault lock policies, see [Amazon Glacier Access Control with Vault Lock Policies \(p. 132\)](#).

A vault lock policy is different than a vault access policy. Both policies govern access controls to your vault. However, a vault lock policy can be locked to prevent future changes, providing 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. In contrast, you use a vault access policy to implement access controls that are not compliance related, temporary, and subject to frequent modification. Vault lock and vault access policies can be used 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).

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 in the in-progress state, you have 24 hours to validate your vault lock policy before the lock ID expires.
2. Use the lock ID to complete the lock process. If the vault lock policy doesn't work as expected, you can abort the lock and restart from the beginning. For information on how to use the Amazon Glacier API to lock a vault, see [Locking a Vault by Using the Amazon Glacier API \(p. 60\)](#).

Locking a Vault by Using the Amazon Glacier API

To lock your vault with the Amazon Glacier API, you first call [Initiate Vault Lock \(POST lock-policy\) \(p. 191\)](#) with a vault lock policy that specifies the controls you want to deploy. [Initiate Vault Lock \(POST lock-policy\) \(p. 191\)](#) 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 \(POST lockid\) \(p. 173\)](#) with the lock ID returned from [Initiate Vault Lock \(POST lock-policy\) \(p. 191\)](#). After the vault is locked it cannot be unlocked.

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 \(POST lock-policy\) \(p. 191\)](#) 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 \(DELETE lock-policy\) \(p. 167\)](#). To fine-tune your policy, you can repeat the [Abort Vault Lock \(DELETE lock-policy\) \(p. 167\)](#)/[Initiate Vault Lock \(POST lock-policy\) \(p. 191\)](#) 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 \(POST lockId\) \(p. 173\)](#) 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 \(DELETE lock-policy\) \(p. 167\)](#).

Related Sections

- [Amazon Glacier Access Control with Vault Lock Policies \(p. 132\)](#)
- [Abort Vault Lock \(DELETE lock-policy\) \(p. 167\)](#)
- [Complete Vault Lock \(POST lockId\) \(p. 173\)](#)
- [Get Vault Lock \(GET lock-policy\) \(p. 186\)](#)
- [Initiate Vault Lock \(POST lock-policy\) \(p. 191\)](#)

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. 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 region in which the archive is stored. The following is an example archive ID.

```
TJgHcrOSfAkV6hdPqOATYfp_0ZaxL1pIBOc02iz0gDPMr2ig-  
nhwd_PafstsdIf6HSrjHnP-3p6LCJC1YytFT_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 AWS 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](#) (p. 62)
- [Maintaining Client-Side Archive Metadata](#) (p. 63)
- [Uploading an Archive in Amazon Glacier](#) (p. 64)
- [Downloading an Archive in Amazon Glacier](#) (p. 79)
- [Deleting an Archive in Amazon Glacier](#) (p. 107)

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](#) (p. 64).

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 AWS 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](#).

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](#) (p. 79).

Deleting an Archive in Amazon Glacier

Amazon Glacier provides API call you can use to delete one archive at a time. For more information, see [Deleting an Archive in Amazon Glacier](#) (p. 107).

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 key name 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 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 AWS 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 AWS SDK for Java, the AWS SDK for .NET, and the REST API.

Topics

- [Options for Uploading an Archive to Amazon Glacier \(p. 64\)](#)
- [Uploading an Archive in a Single Operation \(p. 64\)](#)
- [Uploading Large Archives in Parts \(Multipart Upload\) \(p. 70\)](#)

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 \(p. 64\)](#).
- **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 \(Multipart Upload\) \(p. 70\)](#).

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 Import/Export Service

To upload existing data to Amazon Glacier, you might consider using the AWS Import/Export service. AWS Import/Export accelerates moving large amounts of data into and out of AWS using portable storage devices for transport. AWS transfers your data directly onto and off of storage devices using Amazon's high-speed internal network, bypassing the Internet. For more information, go to the [AWS Import/Export](#) detail page.

Uploading an Archive in a Single Operation

As described in [Uploading an Archive in Amazon Glacier \(p. 64\)](#), you can upload smaller archives in a single operation. However, we encourage 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 SDK for Java \(p. 65\)](#)
- [Uploading an Archive in a Single Operation Using the AWS SDK for .NET in Amazon Glacier \(p. 68\)](#)
- [Uploading an Archive in a Single Operation Using the REST API \(p. 70\)](#)

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

Both the [high-level and low-level APIs \(p. 112\)](#) provided by the AWS 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 \(p. 65\)](#)
- [Uploading an Archive in a Single Operation Using the Low-Level API of the AWS SDK for Java \(p. 66\)](#)

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 regions and endpoints, see [Accessing Amazon Glacier \(p. 5\)](#).

For step-by-step instructions on how to run this example, see [Running Java Examples for Amazon Glacier Using Eclipse \(p. 115\)](#). 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.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 .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 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 \(p. 113\)](#).

3. Execute the `uploadArchive` method by providing the request object as a parameter.

In response, 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 \(p. 115\)](#). 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 \(p. 112\)](#) provided by the AWS 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 \(p. 68\)](#)
- [Uploading an Archive in a Single Operation Using the Low-Level API of the AWS SDK for .NET \(p. 69\)](#)

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 \(p. 117\)](#). You need to update the code as shown with the name of the file you want to upload.

```
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;  
            }  
            catch { }  
        }  
    }  
}
```

```
        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 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 \(p. 116\)](#).

3. Execute 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 \(POST archive\) \(p. 210\)](#).

For step-by-step instructions on how to run this example, see [Running Code Examples \(p. 117\)](#). You need to update the code as shown with the name of the file you want to upload.

```
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 Amazon Glacier Upload Archive API call to upload an archive in a single operation. For more information, see [Upload Archive \(POST archive\)](#) (p. 210).

Uploading Large Archives in Parts (Multipart Upload)

Topics

- [Multipart Upload Process](#) (p. 71)
- [Quick Facts](#) (p. 72)
- [Uploading Large Archives in Parts Using the AWS SDK for Java](#) (p. 72)
- [Uploading Large Archives Using the AWS SDK for .NET](#) (p. 76)
- [Uploading Large Archives in Parts Using the REST API](#) (p. 79)

Multipart Upload Process

As described in [Uploading an Archive in Amazon Glacier](#) (p. 64), we encourage Amazon Glacier customers to use Multipart Upload to upload archives greater than 100 MB.

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. The ID is valid for at least 24 hours.

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

Note

You don't need to know the overall archive size when using the Multipart Upload. This allows for use cases where the archive size is not known when you start uploading the archive. You only need to decide part size at the time you initiate a 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 Abort) 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 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 abort a multipart upload, you cannot upload any more parts using that multipart upload ID. All storage consumed by any parts associated with the aborted multipart upload is freed. If any part uploads were in-progress, they can still succeed or fail even after you abort.

Additional Multipart Upload Operations

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 aborted. 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 GB
Maximum number of parts per upload	10,000
Part size	1 MB to 4 GB, last part can be < 1 MB. You specify the size value in bytes. The part size must be a megabyte (1024 KB) multiplied by a power of 2. For example, 1048576 (1 MB), 2097152 (2 MB), 4194304 (4 MB), 8388608 (8 MB).
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 in Parts Using the AWS SDK for Java

Both the [high-level and low-level APIs \(p. 112\)](#) provided by the AWS SDK for Java provide a method to upload a large archive (see [Uploading an Archive in Amazon Glacier \(p. 64\)](#)).

- 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 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 \(p. 113\)](#).

Topics

- [Uploading Large Archives in Parts Using the High-Level API of the AWS SDK for Java \(p. 73\)](#)
- [Upload Large Archives in Parts Using the Low-Level API of the AWS SDK for Java \(p. 73\)](#)

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 \(p. 65\)](#).

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 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 \(p. 115\)](#). 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.

```
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 \(p. 112\)](#) provided by the AWS SDK for .NET provide a method to upload large archives in parts (see [Uploading an Archive in Amazon Glacier \(p. 64\)](#)).

- 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 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 \(p. 116\)](#).

Topics

- [Uploading Large Archives in Parts Using the High-Level API of the AWS SDK for .NET \(p. 76\)](#)
- [Uploading Large Archives in Parts Using the Low-Level API of the AWS SDK for .NET \(p. 76\)](#)

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 \(p. 68\)](#).

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 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 AWS 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 \(p. 117\)](#). You need to update the code as shown with the name of a file you want to upload.

```
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 \(Multipart Upload\)](#) (p. 70), multipart upload refers to a set of Amazon Glacier 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 \(POST multipart-uploads\)](#) (p. 219)
- [Upload Part \(PUT uploadID\)](#) (p. 232)
- [Complete Multipart Upload \(POST uploadID\)](#) (p. 216)
- [Abort Multipart Upload \(DELETE uploadID\)](#) (p. 214)
- [List Parts \(GET uploadID\)](#) (p. 222)
- [List Multipart Uploads \(GET multipart-uploads\)](#) (p. 227)

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 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 [AWS CLI Reference for Amazon Glacier](#). To install the AWS CLI, see [AWS Command Line Interface](#). The following **Downloading an Archive** topics describe how to upload archives to Amazon Glacier by using the AWS SDK for Java, the AWS SDK for .NET, and the REST API.

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 completes. To initiate an archive retrieval job you use the [Initiate Job \(POST jobs\)](#) (p. 249) REST API or the equivalent in the AWS CLI, or AWS SDKs.

Topics

- [Archive Retrieval Options](#) (p. 81)
- [Ranged Archive Retrievals](#) (p. 81)

Retrieving an archive from Amazon Glacier is a two-step 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. For more information, see [Downloading a Vault Inventory in Amazon Glacier \(p. 34\)](#).
 - b. 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 might find it cost effective to initiate several sequential jobs to prepare your archive. For example, to retrieve a 1 GB archive, you might choose to send a series of four initiate archive-retrieval job requests, each time requesting Amazon Glacier to prepare only a 256 MB portion of the archive. You can send the series of initiate requests anytime. However, it is more cost effective if you wait for a previous initiate request to complete before sending the next request. For more information about the benefits of range retrievals, see [Ranged Archive Retrievals \(p. 81\)](#).

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

Important

For Standard retrievals only, 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 \(p. 140\)](#).

For more information about the `PolicyEnforcedException` exception, see [Error Responses \(p. 163\)](#).

2. **After the job completes, download the bytes.**

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 will not expire for at least 24 hours after completion, which means you can download the output within the 24-hour period after the job is completed. 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 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 configuration 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 \(p. 47\)](#).

- **Request job information explicitly** — You can also use the Amazon Glacier describe job operation ([Describe Job \(GET JobID\) \(p. 236\)](#)) to periodically poll for job information. However, we recommend using Amazon SNS notifications.

Note

The information you get by using SNS notification is the same as what you get by calling Describe Job.

Archive Retrieval Options

You can specify one of the following when initiating a job to retrieve an archive based on your access time and cost requirements. For information about retrieval pricing, see the [Amazon Glacier Pricing](#).

- **Expedited** — Expedited retrievals allow you to quickly access your data when occasional urgent requests for a subset of archives are required. For all but the largest archives (250 MB+), data accessed using Expedited retrievals are typically made available within 1–5 minutes. There are two types of Expedited retrievals: On-Demand and Provisioned. On-Demand requests are similar to EC2 On-Demand instances and are available most of the time. Provisioned requests are guaranteed to be available when you need them. For more information, see [Provisioned Capacity \(p. 81\)](#).
- **Standard** — Standard retrievals allow you to access any of your archives within several hours. Standard retrievals typically complete within 3–5 hours. This is the default option for retrieval requests that do not specify the retrieval option.
- **Bulk** — Bulk retrievals are Amazon Glacier's lowest-cost retrieval option, which you can use to retrieve large amounts, even petabytes, of data inexpensively in a day. Bulk retrievals typically complete within 5–12 hours.

To make an Expedited, Standard, or Bulk retrieval, set the `Tier` parameter in the [Initiate Job \(POST jobs\) \(p. 249\)](#) REST API request to the option you want, or the equivalent in the AWS CLI or AWS SDKs. You don't need to designate whether an expedited retrieval is On-Demand or Provisioned. If you have purchased provisioned capacity, then all expedited retrievals are automatically served through your provisioned capacity.

Provisioned Capacity

Provisioned capacity guarantees 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.

You should purchase provisioned retrieval capacity if your workload requires highly reliable and predictable access to a subset of your data in minutes. Without provisioned capacity Expedited retrievals are 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. You can purchase provisioned capacity by using the Amazon Glacier console, the [Purchase Provisioned Capacity \(POST provisioned-capacity\) \(p. 271\)](#) REST API, the AWS SDKs, or the AWS CLI. For provisioned capacity pricing information, see [Amazon Glacier Pricing](#).

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 completes. 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 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 completes), the range you request in the job initiation must also be tree-hash aligned. Checksums are a way you can 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](#) (p. 162).

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

Both the [high-level and low-level APIs](#) (p. 112) provided by the AWS 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](#) (p. 82)
- [Downloading an Archive Using the Low-Level API of the AWS SDK for Java](#) (p. 83)

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. This polling takes about 4 hours. Once the archive is available, download will begin.

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](#) (p. 115). 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.

```
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 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 `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.

Most Amazon Glacier jobs take about four hours 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](#) (p. 162).

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](#) (p. 90).

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](#) (p. 90).

Caution

Note that it takes about four hours for most jobs to complete.

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](#) (p. 112) provided by the AWS 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](#) (p. 95)
- [Downloading an Archive Using the Low-Level API of the AWS SDK for .NET](#) (p. 96)

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. This polling takes about 4 hours. Once the archive is available, download will begin.

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 \(p. 117\)](#). 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("This polling takes about 4 hours. 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 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 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.

Most Amazon Glacier jobs take about four hours 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 \(p. 162\)](#).

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 \(p. 103\)](#).

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 \(p. 103\)](#).

Caution

Note that it takes about four hours for most jobs to complete.

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\", " +
            "  \"Statement\" : [ " +
            "    { " +
            "      \"Sid\" : \"sns-rule\", " +
            "      \"Effect\" : \"Allow\", " +
            "      \"Principal\" : \"*\", " +
            "      \"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("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("{QuernArn}", 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\", " +
            "  \"Statement\" : [ " +
            "    { " +
            "      \"Sid\" : \"sns-rule\", " +
            "      \"Effect\" : \"Allow\", " +
            "      \"Principal\" : \"*\", " +
            "      \"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.USSouth2))
    {
        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.USSouth2);
    sqsClient = new AmazonSQSClient(Amazon.RegionEndpoint.USSouth2);

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

    CreateQueueRequest createQueueRequest = new CreateQueueRequest();
    createQueueRequest.QueueName = "GlacierDownload-" + ticks;
    CreateQueueResponse createQueueResponse =
sqsClient.CreateQueue(createQueueRequest);
    queueUrl = createQueueResponse.QueueUrl;
    Console.WriteLine("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.WriteLine("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 an Archive Using the REST API

To download an archive 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 \(POST jobs\)](#) (p. 249).
2. After the job completes, download the archive data. For more information, see [Get Job Output \(GET output\)](#) (p. 243).

Deleting an Archive in Amazon Glacier

You cannot delete an archive using the 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. For information on using the CLI with Amazon Glacier, see [AWS CLI Reference for Amazon Glacier](#). The following topics explain how to use the AWS SDK for Java and .NET wrapper libraries, and the REST API.

Topics

- [Deleting an Archive in Amazon Glacier Using the AWS SDK for Java](#) (p. 108)
- [Deleting an Archive in Amazon Glacier Using the AWS SDK for .NET](#) (p. 109)
- [Deleting an Archive Using the REST API](#) (p. 111)

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](#) (p. 34).

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 \(p. 79\)](#).

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.

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 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 \(p. 113\)](#).

3. Execute 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 \(DELETE archive\) \(p. 208\)](#).

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 \(p. 115\)](#). You need to update the code as shown with a vault name and the archive ID of the archive you want to delete.

```
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](#) (p. 112) provided by the AWS 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](#) (p. 109)
- [Deleting an Archive Using the Low-Level API AWS SDK for .NET](#) (p. 110)

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](#) (p. 117). You need to update the code as shown with the archive ID of the archive you want to delete.

```
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 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 \(p. 112\)](#).

3. Execute 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 \(DELETE archive\) \(p. 208\)](#).

For step-by-step instructions on how to run this example, see [Running Code Examples \(p. 117\)](#). You need to update the code as shown with the archive ID of the archive you want to delete.

```
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 Archive Using the REST API

You can use the Amazon Glacier Delete Archive API to delete an archive.

- For information about the Delete Archive API, see [Delete Archive \(DELETE archive\) \(p. 208\)](#).
- For information about using the Amazon Glacier REST API, see [API Reference for Amazon Glacier \(p. 150\)](#).

Using the AWS SDKs with Amazon Glacier

Amazon Web Services 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

- [AWS SDKs that Support Amazon Glacier](#) (p. 112)
- [AWS SDK Libraries for Java and .NET](#) (p. 113)
- [Using the AWS SDK for Java with Amazon Glacier](#) (p. 113)
- [Using the AWS SDK for .NET with Amazon Glacier](#) (p. 116)

AWS SDKs that Support Amazon Glacier

Amazon Glacier is supported by the following AWS SDKs:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for JavaScript in Node.js](#)
- [AWS SDK for .NET](#)
- [AWS SDK for PHP](#)
- [AWS SDK for Python \(Boto\)](#)
- [AWS SDK for Ruby](#)

You can find examples of working with Amazon Glacier using the Java and .NET SDKs throughout this developer guide. For libraries and sample code in all languages, see [Sample Code & Libraries](#).

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 [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.

What Is the Low-Level API?

The low-level wrapper libraries map closely the underlying REST API ([API Reference for Amazon Glacier \(p. 150\)](#)) 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 \(p. 113\)](#) and [Using the AWS SDK for .NET with Amazon Glacier \(p. 116\)](#).

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 \(p. 113\)](#) and [Using the AWS SDK for .NET with Amazon Glacier \(p. 116\)](#).

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 the AWS SDK for Java with Amazon Glacier

The AWS SDK for Java provides a Java API for Amazon Glacier. For more information about downloading the AWS SDK for Java, go to [AWS SDK for Java](#). As described in [Using the AWS SDKs with Amazon Glacier \(p. 112\)](#), AWS SDK for Java provides both the high-level and low-level APIs.

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 \(p. 114\)](#)
- [Using the High-Level API \(p. 114\)](#)
- [Running Java Examples for Amazon Glacier Using Eclipse \(p. 115\)](#)
- [Setting the Endpoint \(p. 115\)](#)

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 \(p. 150\)](#)). 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 \(PUT vault\) \(p. 171\)](#)). 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 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 region you specified when creating the `ArchiveTransferManager` object. If you don't specify any region, the AWS SDK for Java sets `us-east-1` as the default 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 AWS 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.

```
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 regions and endpoints, see [Accessing Amazon Glacier \(p. 5\)](#).

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 \(p. 112\)](#), 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 \(p. 116\)](#)
- [Using the High-Level API \(p. 117\)](#)
- [Running Code Examples \(p. 117\)](#)
- [Setting the Endpoint \(p. 118\)](#)

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 \(p. 150\)](#)). When calling any of these methods, you must create a corresponding request object and provide a response object in which the method can return an 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 \(PUT vault\) \(p. 171\)](#)). To use this method, you must create instances of the `CreateVaultRequest` and `CreateVaultResponse` classes to provide request information and receive an 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 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 [AWS 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 AWS SDK for .NET topic Configuring AWS Credentials .
2	Create a new Visual Studio project using the <i>AWS Empty Project</i> template.
3	Replace the code in the project file, <code>Program.cs</code> , 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 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.

```
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 regions and endpoints, see [Accessing Amazon Glacier \(p. 5\)](#).

Authentication and Access Control for Amazon Glacier

Access to Amazon Glacier requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as an Amazon Glacier vault or an Amazon S3 bucket. The following sections provide details on how you can use [AWS Identity and Access Management \(IAM\)](#) and Amazon Glacier to help secure your resources by controlling who can access them:

- [Authentication](#) (p. 119)
- [Access Control](#) (p. 120)

Authentication

You can access AWS as any of the following types of identities:

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your AWS account. These are your *root credentials* and they provide complete access to all of your AWS resources.

Important

For security reasons, we recommend that you use the root credentials only to create an *administrator user*, which is an *IAM user* with full permissions to your AWS account. Then, you can use this administrator user to create other IAM users and roles with limited permissions. For more information, see [IAM Best Practices](#) and [Creating an Admin User and Group](#) in the *IAM User Guide*.

- **IAM user** – An *IAM user* is simply an identity within your AWS account that has specific custom permissions (for example, permissions to create a vault in Amazon Glacier). You can use an IAM user name and password to sign in to secure AWS webpages like the [AWS Management Console](#), [AWS Discussion Forums](#), or the [AWS Support Center](#).

In addition to a user name and password, you can also generate [access keys](#) for each user. You can use these keys when you access AWS services programmatically, either through [one of the several SDKs](#) or by using the [AWS Command Line Interface \(CLI\)](#). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use the AWS tools, you must sign the request yourself. Amazon Glacier supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 Signing Process](#) in the *AWS General Reference*.

- **IAM role** – An [IAM role](#) is another IAM identity you can create in your account that has specific permissions. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:
 - **Federated user access** – Instead of creating an IAM user, you can use preexisting user identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an [identity provider](#). For more information about federated users, see [Federated Users and Roles](#) in the *IAM User Guide*.
 - **Cross-account access** – You can use an IAM role in your account to grant another AWS account permissions to access your account's resources. For an example, see [Tutorial: Delegate Access Across AWS Accounts Using IAM Roles](#) in the *IAM User Guide*.
 - **AWS service access** – You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data stored in the bucket into an Amazon Redshift cluster. For more information, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *IAM User Guide*.
 - **Applications running on Amazon EC2** – Instead of storing access keys within the EC2 instance for use by applications running on the instance and making AWS API requests, you can use an IAM role to manage temporary credentials for these applications. To assign an AWS role to an EC2 instance and make it available to all of its applications, you can create an instance profile that is attached to the instance. An instance profile contains the role and enables programs running on the EC2 instance to get temporary credentials. For more information, see [Using Roles for Applications on Amazon EC2](#) in the *IAM User Guide*.

Access Control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access Amazon Glacier resources. For example, you must have permissions to create an Amazon Glacier vault.

The following sections describe how to manage permissions. We recommend that you read the overview first.

- [Overview of Managing Access Permissions to Your Amazon Glacier Resources \(p. 121\)](#)

- [Using Identity-Based Policies for Amazon Glacier \(IAM Policies\)](#) (p. 125)
- [Using Resource-Based Policies for Amazon Glacier \(Vault Policies\)](#) (p. 129)

Overview of Managing Access Permissions to Your Amazon Glacier Resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles) and some services (such as Amazon Glacier) also support attaching permissions policies to resources.

Note

An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see [IAM Best Practices](#) in the *IAM User Guide*.

When granting permissions, you decide who is getting the permissions, the resources they get permissions for, and the specific actions that you want to allow on those resources.

Topics

- [Amazon Glacier Resources and Operations](#) (p. 121)
- [Understanding Resource Ownership](#) (p. 121)
- [Managing Access to Resources](#) (p. 122)
- [Specifying Policy Elements: Actions, Effects, Resources, and Principals](#) (p. 124)
- [Specifying Conditions in a Policy](#) (p. 125)

Amazon Glacier Resources and Operations

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 any vault name.

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](#) (p. 150).

Understanding Resource Ownership

A *resource owner* is the AWS account that created the resource. That is, the resource owner is the AWS account of the *principal entity* (the root account, an IAM user, or an IAM role) that authenticates the request that creates the resource. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create an Amazon Glacier vault, your AWS account is the owner of the resource (in Amazon Glacier, the resource is the Amazon Glacier vault).
- If you create an IAM user in your AWS account and grant permissions to create an Amazon Glacier vault to that user, the user can create an Amazon Glacier vault. However, your AWS account, to which the user belongs, owns the Amazon Glacier vault resource.

- If you create an IAM role in your AWS account with permissions to create an Amazon Glacier vault, anyone who can assume the role can create an Amazon Glacier vault. Your AWS account, to which the role belongs, owns the Amazon Glacier vault resource.

Managing Access to Resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

Note

This section discusses using IAM in the context of Amazon Glacier. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see [What Is IAM?](#) in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM policies) and policies attached to a resource are referred to as *resource-based* policies. Amazon Glacier supports both identity-based (IAM policies) and resource-based policies.

Topics

- [Identity-Based Policies \(IAM policies\)](#) (p. 122)
- [Resource-Based Policies \(Amazon Glacier Vault Policies\)](#) (p. 123)

Identity-Based Policies (IAM policies)

You can attach policies to IAM identities. For example you can do the following:

- **Attach a permissions policy to a user or a group in your account** – An account administrator can use a permissions policy that is associated with a particular user to grant permissions for that user to create an Amazon Glacier vault.
- **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:
 1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in Account A.
 2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.
 3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows users in Account B to create or access resources in Account A. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

For more information about using IAM to delegate permissions, see [Access Management](#) in the *IAM User Guide*.

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 [Amazon Glacier Resources and Operations](#) (p. 121).

```
{
```



```
"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.

For more information about using identity-based policies with Amazon Glacier, see [Using Identity-Based Policies for Amazon Glacier \(IAM Policies\)](#) (p. 125). For more information about users, groups, roles, and permissions, see [Identities \(Users, Groups, and Roles\)](#) in the *IAM User Guide*.

Resource-Based Policies (Amazon Glacier Vault Policies)

Each Amazon Glacier vault can have resource-based permissions policies associated with it. For Amazon Glacier, a Amazon Glacier vault is the primary resource and resource-based policies are referred to as *vault policies*.

You use Amazon Glacier vault policies to manage permissions in the following ways:

- Manage user permissions in your account in a single vault policy, instead of individual user policies.
- Manage cross-account permissions as an alternative to using IAM roles.

An Amazon Glacier vault can have one vault access policy and one Vault Lock policy associated with it. An 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 a vault access policy that can be locked. After you lock a Vault Lock policy, the policy cannot be changed. You can use a Vault Lock policy to enforce compliance controls.

You can use vault policies to grant permissions to all users, or you can limit access to a vault to a few AWS accounts by attaching a policy directly to a vault resource. For example, you can use an Amazon Glacier vault policy to grant read-only permissions to all AWS accounts or to grant permissions to upload archives to a few AWS accounts.

Vault policies make it easy to grant cross-account access when you need to share your vault with other AWS accounts. For example, you can grant read-only access on a vault to a business partner with a different AWS account by simply including that account and allowed actions in the vault policy. You can grant cross-account access to multiple users in this fashion and have a single location to view all users with cross-account access in the vault access policy. For an example of a vault policy for cross-account access, see [Example 1: Grant Cross-Account Permissions for Specific Amazon Glacier Actions](#) (p. 130).

The following is an example of a Amazon Glacier vault policy (a resource-based policy). The example policy grants all AWS accounts permissions to perform the `glacier:InitiateJob` and `glacier:GetJobOutput` actions. This policy allows any AWS account to retrieve data from the specified vault.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "add-read-only-perm",
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "glacier:InitiateJob",
        "glacier:GetJobOutput"
      ],
      "Resource": [
        "arn:aws:glacier:us-west-2:123456789012:vaults/examplevault"
      ]
    }
  ]
}
```

For more information about using vault policies with Amazon Glacier, see [Using Resource-Based Policies for Amazon Glacier \(Vault Policies\)](#) (p. 129). For additional information about IAM roles (identity-based policies) as opposed to resource-based policies, see [How IAM Roles Differ from Resource-based Policies](#) in the *IAM User Guide*.

Specifying Policy Elements: Actions, Effects, Resources, and Principals

For each type of Amazon Glacier resource, the service defines a set of API operations (see [API Reference for Amazon Glacier](#) (p. 150)). To grant permissions for these API operations Amazon Glacier defines a set of actions that you can specify in a policy. Note that, performing an API operation can require permissions for more than one action. When granting permissions for specific actions, you also identify the resource on which the actions are allowed or denied.

The following are the most basic policy elements:

- **Resource** – In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies. For more information, see [Amazon Glacier Resources and Operations](#) (p. 121).
- **Actions** – You use action keywords to identify resource operations that you want to allow or deny.

For example, the `glacier:CreateVault` permission allows the user permissions to perform the Amazon Glacier `Create Vault` operation.

- **Effect** – You specify the effect when the user requests the specific action—this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource, which you might do to make sure that a user cannot access it, even if a different policy grants access.
- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only).

To learn more about the IAM policy syntax, and descriptions, see [AWS IAM Policy Reference](#) in the *IAM User Guide*.

For a table showing all of the Amazon Glacier API actions and the resources that they apply to, see [Amazon Glacier API Permissions: Actions, Resources, and Conditions Reference \(p. 134\)](#).

Specifying Conditions in a Policy

When you grant permissions, you can use the IAM policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see [Condition](#) in the *IAM User Guide*.

AWS provides a set of predefined condition keys, called *AWS-wide condition keys*, for all AWS services that support IAM for access control. AWS-wide condition keys use the prefix `aws`. Amazon Glacier supports all AWS-wide condition keys in vault access and Vault Lock policies. For example, you can use the `aws:MultiFactorAuthPresent` condition key to require multi-factor authentication (MFA) when requesting an action. For more information and a list of the AWS-wide condition keys, see [Available Keys for Conditions](#) in the *IAM User Guide*.

Note

Condition keys are case-sensitive.

In addition, Amazon Glacier also provides its own condition keys that you can include in `Condition` elements in an IAM permissions policy. Amazon Glacier-specific condition keys are applicable only when granting Amazon Glacier-specific permissions. Amazon Glacier condition key names have the prefix `glacier:`. The following table shows the Amazon Glacier condition keys that apply to Amazon Glacier resources.

Amazon Glacier Condition Key	Description	Value Type
<code>glacier:ArchiveAgeInDays</code>	Used to evaluate how long an archive has been stored in the vault, in days.	Integer
<code>glacier:ResourceTagKey</code>	Allows you to use a tag in your policy. For information about resource tagging, see Managing Access Control with Tagging (p. 145) .	String

For examples of using the Amazon Glacier-specific condition keys, see [Amazon Glacier Access Control with Vault Lock Policies \(p. 132\)](#).

Related Topics

- [Using Identity-Based Policies for Amazon Glacier \(IAM Policies\) \(p. 125\)](#)
- [Using Resource-Based Policies for Amazon Glacier \(Vault Policies\) \(p. 129\)](#)
- [Amazon Glacier API Permissions: Actions, Resources, and Conditions Reference \(p. 134\)](#)

Using Identity-Based Policies for Amazon Glacier (IAM Policies)

This topic provides examples of identity-based policies in which an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles).

Important

We recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your Amazon Glacier resources. For

more information, see [Overview of Managing Access Permissions to Your Amazon Glacier Resources](#) (p. 121).

The sections in this topic cover the following:

- [Permissions Required to Use the Amazon Glacier Console](#) (p. 126)
- [AWS Managed Policies \(Predefined Policies\) for Amazon Glacier](#) (p. 127)
- [Customer Managed Policy Examples](#) (p. 127)

The following shows an example of a permissions policy.

```
{
  "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 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.

The wildcard character (*) at the end of the ARN means that this statement can match any vault name. The statement allows the `glacier:DescribeVault` action on any vault in the specified region, `us-west-2`. If you want to limit permissions for this action to a specific vault only, you replace the wildcard character (*) with a vault name.

Permissions Required to Use the Amazon Glacier Console

The Amazon Glacier console provides an integrated environment for you to create and manage Amazon Glacier vaults. At a minimum IAM users that you create must be granted permissions for the `glacier:ListVaults` action to view the Amazon Glacier console as shown in the following example.

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

```
}
```

Both of the Amazon Glacier AWS Managed policies discussed in the next section grant permissions for `glacier:ListVaults`.

AWS Managed Policies (Predefined Policies) for Amazon Glacier

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.

Note

You can review these permissions policies by signing in to the IAM console and searching for specific policies there.

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 IAM users or groups that require those permissions or to custom execution roles (IAM roles) that you create for your Amazon Glacier vaults.

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 AWS 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 \(p. 127\)](#)
- [Example 2: Allow a User to Create a Vault and Configure Notifications \(p. 128\)](#)
- [Example 3: Allow a User to Upload Archives to a Specific Vault \(p. 128\)](#)
- [Example 4: Allow a User Full Permissions on a Specific Vault \(p. 129\)](#)

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 \(POST jobs\)](#) (p. 249).

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 \(GET vault\)](#) (p. 181) or the [List Vaults \(GET vaults\)](#) (p. 196) API actions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/
examplevault",
      "Action": [ "glacier:InitiateJob",
                  "glacier:GetJobOutput",
                  "glacier:DescribeJob" ]
    }
  ]
}
```

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](#) (p. 47). The policy also grants permissions to list vaults in the 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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/*",
      "Action": [ "glacier:CreateVault",
                  "glacier:SetVaultNotifications",
                  "glacier:GetVaultNotifications",
                  "glacier>DeleteVaultNotifications",
                  "glacier:DescribeVault",
                  "glacier:ListVaults" ]
    }
  ]
}
```

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 \(POST archive\)](#) (p. 210) API operation or in parts using the [Initiate Multipart Upload \(POST multipart-uploads\)](#) (p. 219) API operation.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/
examplevault",
      "Action": [ "glacier:UploadArchive",
                  "glacier:InitiateMultipartUpload",
                  "glacier:UploadMultipartPart",
                  "glacier:ListParts",
                  "glacier:ListMultipartUploads",
                  "glacier:CompleteMultipartUpload" ]
    }
  ]
}
```

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`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": "arn:aws:glacier:us-west-2:123456789012:vaults/
examplevault",
      "Action": [ "glacier:*" ]
    }
  ]
}
```

Using Resource-Based Policies for Amazon Glacier (Vault Policies)

An Amazon Glacier vault is the primary resource in Amazon Glacier. You can add permissions to the policy associated with a Amazon Glacier vault. Permissions policies attached to Amazon Glacier vaults are referred to as *resource-based policies* (or *vault policies* in Amazon Glacier). Each Amazon Glacier vault can have resource-based permissions policies associated with it. For information about available permissions policy options, see [Managing Access to Resources](#) (p. 122).

Important

Before you create resource-based policies, we recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your Amazon Glacier resources. For more information, see [Overview of Managing Access Permissions to Your Amazon Glacier Resources](#) (p. 121).

An Amazon Glacier vault can have one vault access policy and one Vault Lock policy associated with it. An 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.

For more information, see the following topics.

Topics

- [Amazon Glacier Access Control with Vault Access Policies \(p. 130\)](#)
- [Amazon Glacier Access Control with Vault Lock Policies \(p. 132\)](#)

Amazon Glacier Access Control with Vault Access Policies

An Amazon Glacier *vault access policy* is a resource-based policy that you can use to manage permissions to your vault. For information about the different permissions policy options available, see [Managing Access to Resources \(p. 122\)](#).

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 [Amazon Glacier Access Control with Vault Lock Policies \(p. 132\)](#).

You can use the Amazon Glacier API, AWS SDKs, AWS CLI, or the Amazon Glacier console to create and manage vault access policies. For a list of Amazon Glacier operations allowed for vault access resource-based policies, see [Amazon Glacier API Permissions: Actions, Resources, and Conditions Reference \(p. 134\)](#).

Examples

- [Example 1: Grant Cross-Account Permissions for Specific Amazon Glacier Actions \(p. 130\)](#)
- [Example 2: Grant Read-Only Permissions to All AWS Accounts \(p. 131\)](#)
- [Example 3: Grant Cross-Account Permissions for MFA Delete Operations \(p. 131\)](#)

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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "cross-account-upload",
      "Principal": {
        "AWS": [
          "arn:aws:iam::123456789012:root",
          "arn:aws:iam::444455556666:root"
        ]
      }
    }
  ],
}
```



```

    "Effect": "Allow",
    "Action": [
      "glacier:UploadArchive",
      "glacier:InitiateMultipartUpload",
      "glacier:AbortMultipartUpload",
      "glacier:CompleteMultipartUpload"
    ],
    "Resource": [
      "arn:aws:glacier:us-west-2:999999999999:vaults/examplevault"
    ]
  }
]
}

```

Example 2: Grant Read-Only Permissions to All AWS Accounts

The following example policy grants permissions that allow all AWS accounts to perform Amazon Glacier operations to retrieve any archive in a vault named `examplevault`. The retrieved archives will be read-only for these accounts.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "add-read-only-perm",
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "glacier:InitiateJob",
        "glacier:GetJobOutput"
      ],
      "Resource": [
        "arn:aws:glacier:us-west-2:999999999999:vaults/examplevault"
      ]
    }
  ]
}

```

Example 3: 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*.

```

{
  "Version": "2012-10-17",

```

```
"Statement": [
  {
    "Sid": "add-mfa-delete-requirement",
    "Principal": {
      "AWS": [
        "arn:aws:iam::123456789012:root"
      ]
    },
    "Effect": "Allow",
    "Action": [
      "glacier:Delete*"
    ],
    "Resource": [
      "arn:aws:glacier:us-west-2:999999999999:vaults/examplevault"
    ],
    "Condition": {
      "Bool": {
        "aws:MultiFactorAuthPresent": true
      }
    }
  }
]
```

Related Sections

- [Delete Vault Access Policy \(DELETE access-policy\) \(p. 177\)](#)
- [Get Vault Access Policy \(GET access-policy\) \(p. 183\)](#)
- [Set Vault Access Policy \(PUT access-policy\) \(p. 203\)](#)

Amazon Glacier Access Control with Vault Lock Policies

An 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 \(p. 60\)](#).

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 [Amazon Glacier Vault Lock \(p. 60\)](#). If you want to manage other user permissions that can be changed, you can use the vault access policy (see [Amazon Glacier Access Control with Vault Access Policies \(p. 130\)](#)).

You can use the Amazon Glacier API, AWS 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 [Amazon Glacier API Permissions: Actions, Resources, and Conditions Reference \(p. 134\)](#).

Examples

- [Example 1: Deny Deletion Permissions for Archives Less Than 365 Days Old \(p. 133\)](#)
- [Example 2: Deny Deletion Permissions Based on a Tag \(p. 133\)](#)

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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "deny-based-on-archive-age",
      "Principal": "*",
      "Effect": "Deny",
      "Action": "glacier:DeleteArchive",
      "Resource": [
        "arn:aws:glacier:us-west-2:123456789012:vaults/examplevault"
      ],
      "Condition": {
        "NumericLessThan": {
          "glacier:ArchiveAgeInDays": "365"
        }
      }
    }
  ]
}
```

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 to 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 because the vault has been locked by the first statement.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "no-one-can-delete-any-archive-from-vault",
      "Principal": "*",
      "Effect": "Deny",
      "Action": [
        "glacier:DeleteArchive"
      ],
      "Resource": [
```

```
    "arn:aws:glacier:us-west-2:123456789012:vaults/examplevault"
  ],
  "Condition": {
    "StringLike": {
      "glacier:ResourceTag/LegalHold": [
        "true",
        ""
      ]
    }
  }
},
{
  "Sid": "you-can-delete-archive-less-than-1-year-old",
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
    "glacier:DeleteArchive"
  ],
  "Resource": [
    "arn:aws:glacier:us-west-2:123456789012:vaults/examplevault"
  ],
  "Condition": {
    "NumericLessThan": {
      "glacier:ArchiveAgeInDays": "365"
    }
  }
}
]
}
```

Related Sections

- [Amazon Glacier Vault Lock \(p. 60\)](#)
- [Abort Vault Lock \(DELETE lock-policy\) \(p. 167\)](#)
- [Complete Vault Lock \(POST lockId\) \(p. 173\)](#)
- [Get Vault Lock \(GET lock-policy\) \(p. 186\)](#)
- [Initiate Vault Lock \(POST lock-policy\) \(p. 191\)](#)

Amazon Glacier API Permissions: Actions, Resources, and Conditions Reference

When you are setting up [Access Control \(p. 120\)](#) 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 [Overview of Managing Access Permissions to Your Amazon Glacier Resources](#) (p. 121).

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 a red asterisk (*) after the name of the API operation in the first column.

Amazon Glacier API and Required Permissions for Actions

[Abort Multipart Upload \(DELETE uploadID\)](#) (p. 214)

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 \(DELETE lock-policy\)](#) (p. 167)

Required Permissions (API Actions): glacier:AbortVaultLock

Resources:

Amazon Glacier Condition Keys:

[Add Tags To Vault \(POST tags add\)](#) (p. 169)

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 \(POST uploadID\)](#) (p. 216)

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 \(POST lockId\)](#) (p. 173)

Required Permissions (API Actions): glacier:CompleteVaultLock

Resources:

Amazon Glacier Condition Keys: glacier:ResourceTag/*TagKey*

[Create Vault \(PUT vault\)](#) (p. 171) *

Required Permissions (API Actions): glacier:CreateVault

Resources:

Amazon Glacier Condition Keys:

[Delete Archive \(DELETE archive\)](#) (p. 208)

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 \(DELETE vault\) \(p. 175\)](#)

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 \(DELETE access-policy\) \(p. 177\)](#)

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 \(DELETE notification-configuration\) \(p. 179\)](#)

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 \(GET JobID\) \(p. 236\)](#)

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 \(GET vault\) \(p. 181\)](#)

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 \(GET policy\) \(p. 266\) *](#)

Required Permissions (API Actions):glacier:GetDataRetrievalPolicy

Resources: arn:aws:glacier:*region*:*account-id*:policies/retrieval-limit-policy

Amazon Glacier Condition Keys:

[Get Job Output \(GET output\) \(p. 243\)](#)

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 \(GET access-policy\) \(p. 183\)](#)

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 \(GET lock-policy\) \(p. 186\)](#)

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 \(GET notification-configuration\) \(p. 189\)](#)

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 \(POST jobs\) \(p. 249\)](#)

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 \(POST multipart-uploads\) \(p. 219\)](#)

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 \(POST lock-policy\) \(p. 191\)](#)

Required Permissions (API Actions):glacier:InitiateVaultLock

Resources:

Amazon Glacier Condition Keys: glacier:ResourceTag/*TagKey*

[List Jobs \(GET jobs\) \(p. 257\)](#)

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 \(GET multipart-uploads\) \(p. 227\)](#)

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 \(GET uploadID\) \(p. 222\)](#)

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 \(GET tags\) \(p. 194\)](#)

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 \(GET vaults\) \(p. 196\)](#)

Required Permissions (API Actions):glacier:ListVaults

Resources:

Amazon Glacier Condition Keys:

[Remove Tags From Vault \(POST tags remove\) \(p. 201\)](#)

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 \(PUT policy\) \(p. 273\) *](#)

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 \(PUT access-policy\) \(p. 203\)](#)

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 \(PUT notification-configuration\) \(p. 205\)](#)

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 \(POST archive\) \(p. 210\)](#)

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 \(PUT uploadID\) \(p. 232\)](#)

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*

Amazon Glacier Data Retrieval Policies

With Amazon Glacier data retrieval policies, you can easily set data retrieval limits and manage the data retrieval activities across your AWS account in each region. For more information about Amazon Glacier data retrieval charges, see [Amazon Glacier pricing](#).

Important

A data retrieval policy applies to standard retrievals only and manages retrieval requests made directly to Amazon Glacier. It does not manage data restore requests for Amazon S3's `GLACIER` storage class. For more information, see [GLACIER Storage Class: Additional Lifecycle Configuration Considerations](#).

Topics

- [Choosing an Amazon Glacier Data Retrieval Policy \(p. 140\)](#)
- [Using the Amazon Glacier Console to Set Up a Data Retrieval Policy \(p. 141\)](#)
- [Using the Amazon Glacier API to Set Up a Data Retrieval Policy \(p. 142\)](#)

Choosing an Amazon Glacier Data Retrieval Policy

You can choose from three types of Amazon Glacier data retrieval policies: *Free Tier Only*, *Max Retrieval Rate*, and *No Retrieval Limit*. By using a Free Tier Only policy, you can keep your retrievals within your daily free tier allowance and not incur any data retrieval cost. If you want to retrieve more data than the free tier, you can use a Max Retrieval Rate policy to set a bytes-per-hour retrieval rate limit. The Max Retrieval Rate policy ensures that the peak retrieval rate from all retrieval jobs across your account in a region does not exceed the bytes-per-hour limit you set. If you don't want to set a retrieval limit, you can use a No Retrieval Limit policy where all valid data retrieval requests will be accepted.

With both Free Tier Only and Max Retrieval Rate policies, data retrieval requests that would exceed the retrieval limits you specified will not be accepted. If you use a Free Tier Only policy, Amazon Glacier will synchronously reject retrieval requests that would exceed your free tier allowance. If you use a Max Retrieval Rate policy, Amazon Glacier will reject retrieval requests that would cause the peak retrieval rate of the in progress jobs to exceed the bytes-per-hour limit 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.
- 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 will not be 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 region under your account. A data retrieval policy is region-specific 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 will always stay within your free tier allowance, so you don't incur data retrieval charges. If a retrieval request is rejected, you will receive an error message stating that the request has been denied by the current data retrieval policy.

You set the data retrieval policy to Free Tier Only for a particular AWS region. Once the policy is set, you cannot retrieve more data in a day than your prorated daily free retrieval allowance for that region and you will not incur data retrieval fees.

You can switch to a Free Tier Only policy after you have incurred data retrieval charges within a month. The Free Tier Only policy will take effect for new retrieval requests, but will 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 limit that has a bytes-per-hour maximum. When you set the data retrieval policy to Max Retrieval Rate, a new retrieval request will be rejected if it would cause the peak retrieval rate of the in progress jobs to exceed the bytes-per-hour limit specified by the policy. If a retrieval job request is rejected, you will 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 free tier you can use in a day. For example, suppose you set Max Retrieval Rate to 1 MB per hour. This is less than the free tier policy rate of 14 MB per hour. To ensure you make good use of the daily 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 on 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 will be accepted and your data retrieval costs will vary based on your usage.

Using the Amazon Glacier Console to Set Up a Data Retrieval Policy

You can view and update the data retrieval policies in the Amazon Glacier console or by using the Amazon Glacier API. To setup a data retrieval policy in the console, choose an AWS region and then click **Settings**.

Data retrieval settings

Retrieval policies | Provisioned capacity

Amazon Glacier data retrieval policies allow you to manage retrieval costs by setting limits on retrieval activities across your AWS account in each region. Retrieval policies apply to standard retrievals.

Free tier only **Max Retrieval Rate** **No Retrieval Limit**

Only retrieve data within the free tier. Data retrieval requests that exceed the free tier will not be accepted. Max Retrieval Rate: GB/hour All valid data retrieval requests will be accepted. Data retrieval cost will vary based on your usage.

Retrieval cost: *Free* Retrieval cost: \$7.20 / month or less Visit the [pricing page](#) for data retrieval pricing information

Note: Data retrieval policies govern all retrieval activities in a region. The retrieval cost estimates may not reflect previously incurred usage or charges in the month. [Learn more](#)

Cancel | Save

You can select one of the three data retrieval policies: **Free Tier Only**, **Max Retrieval Rate**, or **No Retrieval Limit**. If you click **Max Retrieval Rate**, you'll need to specify a value in the **GB/Hour** box. When you type a value in **GB/Hour**, the console will calculate an estimated cost for you. Click **No Retrieval Limit** if you don't want any restrictions placed on the rate of your data retrievals.

You can configure a data retrieval policy for each region. Each policy will take effect within a few minutes after you click **Save**.

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 \(GET policy\)](#) (p. 266) operation. You set a data retrieval policy using the [Set Data Retrieval Policy \(PUT policy\)](#) (p. 273) 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 selecting **Max Retrieval Rate** in the console, `FreeTier` to selecting **Free Tier Only**, and `None` to selecting **No Retrieval Policy**.

When you use the [Initiate Job \(POST jobs\)](#) (p. 249) operation to initiate a data retrieval job that will exceed the maximum retrieval rate set in your data retrieval policy, the Initiate Job operation will abort and throw 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 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 \(p. 112\)](#).

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 vault resources. Using tags is a simple yet powerful way to manage AWS resources and organize data, including billing data.

Topics

- [Tagging Basics \(p. 144\)](#)
- [Tag Restrictions \(p. 145\)](#)
- [Tracking Costs Using Tagging \(p. 145\)](#)
- [Managing Access Control with Tagging \(p. 145\)](#)
- [Related Sections \(p. 145\)](#)

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 \(p. 58\)](#).

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, white space, 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, white space, 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 and Cost Management 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 an example, see [Example 2: Deny Deletion Permissions Based on a Tag](#) (p. 133).

Related Sections

- [Tagging Your Amazon Glacier Vaults](#) (p. 58)

Logging Amazon Glacier API Calls by Using AWS CloudTrail

Amazon Glacier is integrated with CloudTrail, a service that captures API calls made by or on behalf of Amazon Glacier in your AWS account and delivers the log files to an Amazon S3 bucket that you specify. CloudTrail captures API calls from the Amazon Glacier console or from the Amazon Glacier API. Using the information collected by CloudTrail, you can determine what request was made to Amazon Glacier, the source IP address from which the request was made, who made the request, when it was made, and so on. To learn more about CloudTrail, including how to configure and enable it, see the [AWS CloudTrail User Guide](#).

Amazon Glacier Information in CloudTrail

When CloudTrail logging is enabled in your AWS account, API calls made to Amazon Glacier actions are tracked in log files. Amazon Glacier records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

All of the Amazon Glacier actions are logged and are documented in the [API Reference for Amazon Glacier \(p. 150\)](#). For example, calls to the **CreateVault**, **ListVaults**, **DescribeVault**, and **DeleteVault** actions generate entries in the CloudTrail log files.

Every log entry contains information about who generated the request. The user identity information in the log helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information, see the **userIdentity** field in the [CloudTrail Event Reference](#).

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

You can choose to have CloudTrail publish Amazon SNS notifications when new log files are delivered if you want to take quick action upon log file delivery. For more information, see [Configuring Amazon SNS Notifications](#).

You can also aggregate Amazon Glacier log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket. For more information, see [Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket](#).

Understanding Amazon Glacier Log File Entries

CloudTrail log files can contain one or more log entries where each entry is made up of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any parameters, the date and time of the action, and so on. The log entries are not guaranteed to be in any particular order. That is, they are not an ordered stack trace of the public API calls.

The following example shows a CloudTrail log entry that demonstrates the logging of Amazon Glacier 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": "HJiLgvfXCY88QJAC6rRoexS9ThvI2lQ1Nqukfily02hcUPPo",
      "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](#) (p. 153).

In the event of an error, 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 AWS SDKs that provide wrapper libraries to simplify your coding task. These libraries sign each request you send and compute the checksum of the payload in your request. Therefore, using the AWS 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](#) (p. 112).

Topics

- [Common Request Headers](#) (p. 150)
- [Common Response Headers](#) (p. 153)
- [Signing Requests](#) (p. 153)
- [Computing Checksums](#) (p. 157)
- [Error Responses](#) (p. 163)
- [Vault Operations](#) (p. 166)
- [Archive Operations](#) (p. 208)
- [Multipart Upload Operations](#) (p. 214)
- [Job Operations](#) (p. 236)
- [Data Retrieval Operations](#) (p. 266)

Common Request Headers

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	<p>The header that is required to sign requests. Amazon Glacier requires Signature Version 4. For more information, see Signing Requests (p. 153).</p> <p>Type: String</p>	Yes
Content-Length	<p>The length of the request body (without the headers).</p> <p>Type: String</p> <p>Condition: Required only for the Upload Archive (POST archive) (p. 210) API.</p>	Conditional
Date	<p>The date that can be used to create the signature contained in the <code>Authorization</code> header. If the <code>Date</code> header is to be used for signing it must be specified in the ISO 8601 basic format. In this case, the <code>x-amz-date</code> header is not needed. Note that when <code>x-amz-date</code> is present, it always overrides the value of the <code>Date</code> header.</p> <p>If the <code>Date</code> 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 <code>Sun, 23 Nov 2014 12:00:00 GMT</code> is a valid date/time header for use with Amazon Glacier.</p> <p>If you are using the <code>Date</code> header for signing, then it must be in the ISO 8601 basic <code>YYYYMMDD'T'HHMMSS'Z'</code> format.</p> <p>Type: String</p> <p>Condition: If <code>Date</code> is specified but is not in ISO 8601 basic format, then you must also include the <code>x-amz-date</code> header. If <code>Date</code> is specified in ISO 8601 basic format, then this is sufficient for signing requests and you do not need the <code>x-amz-date</code> header. For more information, see Handling Dates in Signature Version 4 in the <i>Amazon Web Services Glossary</i>.</p>	Conditional
Host	<p>This header specifies the service endpoint to which you send your requests. The value must be of the form <code>glacier.<i>region</i>.amazonaws.com</code>, where <i>region</i> is replaced with a region designation such as <code>us-west-2</code>.</p> <p>Type: String</p>	Yes

Header Name	Description	Required
x-amz-content-sha256	<p>The computed SHA256 checksum of an entire payload that is uploaded with either Upload Archive (POST archive) (p. 210) or Upload Part (PUT uploadID) (p. 232). 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.</p> <p>Type: String</p> <p>Condition: Required for streaming API, Upload Archive (POST archive) (p. 210) and Upload Part (PUT uploadID) (p. 232).</p>	Conditional
x-amz-date	<p>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 20141123T120000Z is a valid x-amz-date for use with Amazon Glacier.</p> <p>Type: String</p> <p>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 <i>Amazon Web Services Glossary</i>.</p>	Conditional
x-amz-glacier-version	<p>The Amazon Glacier API version to use. The current version is 2012-06-01.</p> <p>Type: String</p>	Yes
x-amz-sha256-tree-hash	<p>The computed SHA256 tree-hash checksum for an uploaded archive (Upload Archive (POST archive) (p. 210)) or archive part (Upload Part (PUT uploadID) (p. 232)). For more information about calculating this checksum, see Computing Checksums (p. 157).</p> <p>Type: String</p> <p>Default: None</p> <p>Condition: Required for Upload Archive (POST archive) (p. 210) and Upload Part (PUT uploadID) (p. 232).</p>	Conditional

Common Response Headers

The following table describes response headers that are common to most Amazon Glacier responses.

Name	Description
Content-Length	The length in bytes of the response body. Type: String
Date	The date and time Amazon Glacier responded, for example, <code>Sun, 23 Nov 2014 12:00:00 GMT</code> . The format of the date must be one of the full date formats specified by RFC 2616 , section 3.3. Note that <code>Date</code> returned may drift slightly from other dates, so for example, the date returned from an Upload Archive (POST archive) (p. 210) request may not match the date shown for the archive in an inventory list for the vault. 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 (p. 157) . 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, 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 \(p. 154\)](#)
- [Calculating Signatures for the Streaming Operations \(p. 155\)](#)

Example Signature Calculation

The following example walks you through the details of creating a signature for [Create Vault \(PUT vault\) \(p. 171\)](#). The example could be used as a reference to check your signature calculation method. Other reference calculations are included in the [Signature Version 4 Test Suite](#) of the Amazon Web Services Glossary.

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 \(p. 153\)](#) 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
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

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 \(p. 153\)](#) is:

```
AWS4-HMAC-SHA256
20120525T002453Z
```



```
20120525/us-east-1/glacier/aws4_request  
5f1dala2d0feb614dd03d71e87928b8e449ac87614479332aced3a701f916743
```

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](#) (p. 153). The service name to use in the credential scope is `glacier`.

For [Task 3: Create a Signature](#) (p. 154), 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 \(POST archive\)](#) (p. 210) and [Upload Part \(PUT uploadID\)](#) (p. 232) 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](#) (p. 157)). 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 \(POST archive\)](#) (p. 210), 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](#) (p. 153) 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](#) (p. 154). 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 \(p. 156\)](#).

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:

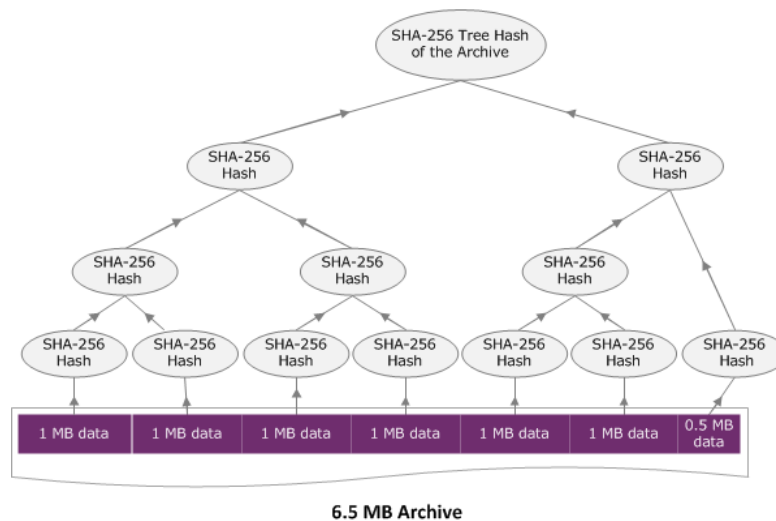
1. For each 1 MB chunk of payload data, compute the SHA-256 hash. The last chunk of data can be less than 1 MB. For example, if you are uploading a 3.2 MB archive, you compute the SHA-256 hash values for each of the first three 1 MB chunks of data, and then compute the SHA-256 hash of the remaining 0.2 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. When only one child node remains, you 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 \(p. 158\)](#)
- [Tree Hash Example 2: Uploading an archive using a multipart upload \(p. 158\)](#)
- [Computing the Tree Hash of a File \(p. 159\)](#)
- [Receiving Checksums When Downloading Data \(p. 162\)](#)

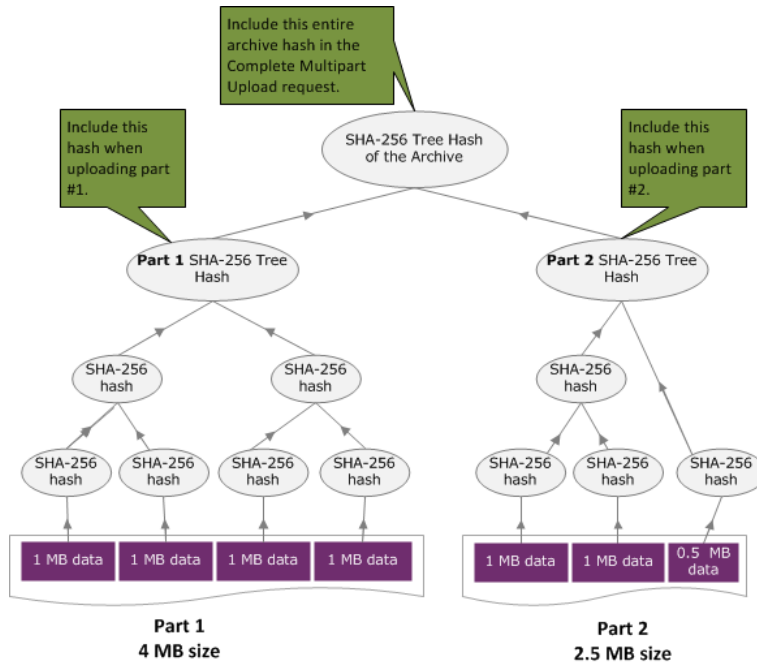
Tree Hash Example 1: Uploading an archive in a single request

When you upload an archive in a single request using the Upload Archive API (see [Upload Archive \(POST archive\)](#) (p. 210)), 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.



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 \(PUT uploadID\)](#) (p. 232) 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 \(POST uploadID\)](#) (p. 216)) 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 AWS SDK to program against Amazon Glacier, the tree hash calculation is done for you and you only need to provide the file reference.

```

* 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
*/
Amazon Glacier Developer Guide
Computing the Tree Hash of a File
public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
throws NoSuchAlgorithmException {

```

Example 1: Java Example

```

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();
}
}

```

```

* 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 \(POST jobs\)](#) (p. 249)), 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 \(GET output\)](#) (p. 243)), 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](#) (p. 162).

Note that a range that is tree-hash aligned is also megabyte aligned. However, a megabyte aligned range is not necessarily tree-hash aligned.

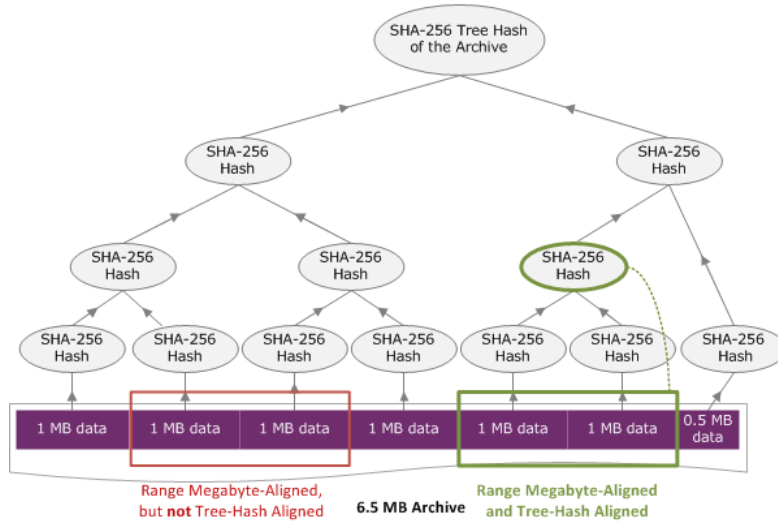
The following cases describe when you receive a checksum value when you download your archive data:

- 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 [Tree Hash Example: Retrieving an archive range that is tree-hash aligned](#) (p. 162). 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 $[P \text{ MB}, Q \text{ MB} - 1 \text{ byte}]$, 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 $[P, P + 1 \text{ MB}]$.
- If P is an even number and k is the maximum number, where P can be written as $2^k * X$, 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 $(0 \leq i \leq k)$ and where $P + 2^i < N$, then $[P, Q + 2^i]$ is a tree-hash aligned range.
 - $P = 0$ is the special case where $A = 2^{\lceil \lg N \rceil} * 0$

Error Responses

In the event of an error, Amazon Glacier 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	403 Forbidden	Client

Code	Description	HTTP Status Code	Type
	used in the request URI. For more information, see Authentication and Access Control for Amazon Glacier (p. 119).		
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
InvalidParameterValueException	Returned if a parameter of the request is incorrectly specified.	400 Bad Request	Client
InvalidSignatureException	Returned if the request signature is invalid.	400 Bad Request	Client
LimitExceededException	Returned if the request results in a vault limit or tags limit being exceeded.	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
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 (p. 140).	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 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 \(GET JobID\)](#) (p. 236) request for a job that does not exist. That is, you specify a job ID that does not exist.

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYws0j1b5gq1JZ55yHgt5vP54ZShjoQzQVVEXAMPLEbadJobID
HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
Date: 20141123T120000Z
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: Sun, 23 Nov 2014 12:00:00 GMT
{
  "code": "ResourceNotFoundException",
  "message": "The job ID was not found: HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYws0j1b5gq1JZ55yHgt5vP54ZShjoQzQVVEXAMPLEbadJobID",
  "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: Sun, 23 Nov 2014 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 \(GET jobs\) \(p. 257\)](#) 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: 20141123T120000Z
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: Sun, 23 Nov 2014 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

- [Abort Vault Lock \(DELETE lock-policy\)](#) (p. 167)
- [Add Tags To Vault \(POST tags add\)](#) (p. 169)
- [Create Vault \(PUT vault\)](#) (p. 171)
- [Complete Vault Lock \(POST lockId\)](#) (p. 173)
- [Delete Vault \(DELETE vault\)](#) (p. 175)
- [Delete Vault Access Policy \(DELETE access-policy\)](#) (p. 177)
- [Delete Vault Notifications \(DELETE notification-configuration\)](#) (p. 179)
- [Describe Vault \(GET vault\)](#) (p. 181)
- [Get Vault Access Policy \(GET access-policy\)](#) (p. 183)
- [Get Vault Lock \(GET lock-policy\)](#) (p. 186)
- [Get Vault Notifications \(GET notification-configuration\)](#) (p. 189)
- [Initiate Vault Lock \(POST lock-policy\)](#) (p. 191)
- [List Tags For Vault \(GET tags\)](#) (p. 194)
- [List Vaults \(GET vaults\)](#) (p. 196)
- [Remove Tags From Vault \(POST tags remove\)](#) (p. 201)
- [Set Vault Access Policy \(PUT access-policy\)](#) (p. 203)
- [Set Vault Notification Configuration \(PUT notification-configuration\)](#) (p. 205)

Abort Vault Lock (DELETE lock-policy)

Description

This operation aborts 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. Aborting 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 \(POST lock-policy\)](#) (p. 191). A vault lock is put into the `Locked` state by calling [Complete Vault Lock \(POST lockId\)](#) (p. 173). You can get the state of a vault lock by calling [Get Vault Lock \(GET lock-policy\)](#) (p. 186). For more information about the vault locking process, see [Amazon Glacier Vault Lock](#) (p. 60). For more information about vault lock policies, see [Amazon Glacier Access Control with Vault Lock Policies](#) (p. 132).

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 \(p. 150\)](#).

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 \(p. 153\)](#).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses \(p. 163\)](#).

Examples

The following example demonstrates how to abort 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: 20141123T120000Z
```

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2x-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: Sun, 23 Nov 2014 12:00:00 GMT
```

Related Sections

- [Complete Vault Lock \(POST lockId\)](#) (p. 173)
- [Get Vault Lock \(GET lock-policy\)](#) (p. 186)
- [Initiate Vault Lock \(POST lock-policy\)](#) (p. 191)

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](#) (p. 144).

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 <code>operation</code> with a value of <code>add</code> to distinguish it from Remove Tags From Vault (POST tags remove) (p. 201).	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](#) (p. 150).

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](#) (p. 153).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses](#) (p. 163).

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: 20141123T120000Z
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: Sun, 23 Nov 2014 12:02:00 GMT
```

Related Sections

- [List Tags For Vault \(GET tags\) \(p. 194\)](#)
- [Remove Tags From Vault \(POST tags remove\) \(p. 201\)](#)

Create Vault (PUT vault)

Description

This operation creates a new vault with the specified name. The name of the vault must be unique within a 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 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 \(p. 150\)](#).

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 \(p. 153\)](#).

Name	Description
<i>Location</i>	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](#) (p. 163).

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: 20141123T120000Z
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: Sun, 23 Nov 2014 12:02:00 GMT
Location: /111122223333/vaults/examplevault
```

Related Sections

- [List Vaults \(GET vaults\)](#) (p. 196)
- [Delete Vault \(DELETE vault\)](#) (p. 175)
- [Authentication and Access Control for Amazon Glacier](#) (p. 119)

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 \(POST lock-policy\)](#) (p. 191). You can obtain the state of the vault lock by calling [Get Vault Lock \(GET lock-policy\)](#) (p. 186). For more information about the vault locking process, see [Amazon Glacier Vault Lock](#) (p. 60).

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 \(POST lock-policy\)](#) (p. 191) 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](#) (p. 150).

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](#) (p. 153).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses](#) (p. 163).

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: 20141123T120000Z
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 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: Sun, 23 Nov 2014 12:02:00 GMT
```

Related Sections

- [Abort Vault Lock \(DELETE lock-policy\)](#) (p. 167)
- [Get Vault Lock \(GET lock-policy\)](#) (p. 186)
- [Initiate Vault Lock \(POST lock-policy\)](#) (p. 191)

Delete Vault (DELETE vault)

Description

This operation deletes a vault. 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 \(GET vault\)](#) (p. 181) 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 [Amazon Glacier Access Control with Vault Access Policies](#) (p. 130).

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](#) (p. 150).

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](#) (p. 153).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses](#) (p. 163).

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: 20141123T120000Z
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: Sun, 23 Nov 2014 12:02:00 GMT
```

Related Sections

- [Create Vault \(PUT vault\) \(p. 171\)](#)
- [List Vaults \(GET vaults\) \(p. 196\)](#)
- [Initiate Job \(POST jobs\) \(p. 249\)](#)
- [Authentication and Access Control for Amazon Glacier \(p. 119\)](#)

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 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 [Amazon Glacier Access Control with Vault Access Policies \(p. 130\)](#).

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 \(p. 150\)](#).

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 \(p. 153\)](#).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses \(p. 163\)](#).

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: 20141123T120000Z
```



```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2x-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: Sun, 23 Nov 2014 12:00:00 GMT
```

Related Sections

- [Get Vault Access Policy \(GET access-policy\) \(p. 183\)](#)
- [Set Vault Access Policy \(PUT access-policy\) \(p. 203\)](#)

Delete Vault Notifications (DELETE notification-configuration)

Description

This operation deletes the notification configuration set for a vault [Set Vault Notification Configuration \(PUT notification-configuration\) \(p. 205\)](#). The operation is eventually consistent—that is, it might take some time for 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 \(p. 150\)](#).

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 \(p. 153\)](#).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses \(p. 163\)](#).

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: 20141123T120000Z
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: Sun, 23 Nov 2014 12:00:00 GMT
```

Related Sections

- [Get Vault Notifications \(GET notification-configuration\)](#) (p. 189)
- [Set Vault Notification Configuration \(PUT notification-configuration\)](#) (p. 205)
- [Authentication and Access Control for Amazon Glacier](#) (p. 119)

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 generated (see [Working with Vaults in Amazon Glacier](#) (p. 22)). 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](#) (p. 150).

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 \(p. 153\)](#).

Response Body

The response body contains the following JSON fields.

CreationDate

The UTC date when the vault was created.

Type: A string representation of 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 \(POST jobs\) \(p. 249\)](#).

Type: A string representation of 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 \(p. 163\)](#).

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: 20141123T120000Z
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_XJC1-Q
Date: Sun, 23 Nov 2014 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 \(PUT vault\) \(p. 171\)](#)
- [List Vaults \(GET vaults\) \(p. 196\)](#)
- [Delete Vault \(DELETE vault\) \(p. 175\)](#)
- [Initiate Job \(POST jobs\) \(p. 249\)](#)
- [Authentication and Access Control for Amazon Glacier \(p. 119\)](#)

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 \(PUT access-policy\) \(p. 203\)](#). 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 [Amazon Glacier Access Control with Vault Access Policies \(p. 130\)](#).

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 \(p. 150\)](#).

Request Body

This operation does not have a request body.

Responses

In response, 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 \(p. 153\)](#).

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](#) (p. 163).

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: 20141123T120000Z
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 \(PUT access-policy\)](#) (p. 203) 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_XJC1-Q
Date: Sun, 23 Nov 2014 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 \(DELETE access-policy\)](#) (p. 177)
- [Set Vault Access Policy \(PUT access-policy\)](#) (p. 203)

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 `InProgress` 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 \(POST lock-policy\)](#) (p. 191). A vault lock is put into the `Locked` state by calling [Complete Vault Lock \(POST lockId\)](#) (p. 173). You can abort the vault locking process by calling [Abort Vault Lock \(DELETE lock-policy\)](#) (p. 167). For more information about the vault locking process, see [Amazon Glacier Vault Lock](#) (p. 60).

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 [Amazon Glacier Access Control with Vault Lock Policies](#) (p. 132).

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 \(p. 150\)](#).

Request Body

This operation does not have a request body.

Responses

In response, 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 \(p. 153\)](#).

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 of 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 of 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 \(p. 163\)](#).

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: 20141123T120000Z
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 \(POST lock-policy\) \(p. 191\)](#) 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: Sun, 23 Nov 2014 12:00:00 GMT
Content-Type: application/json
Content-Length: length

{
  "Policy": "
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "example-vault-lock-policy",
        "Principal": {
          "AWS": "*"
        },
        "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 \(DELETE lock-policy\)](#) (p. 167)
- [Complete Vault Lock \(POST lockId\)](#) (p. 173)
- [Initiate Vault Lock \(POST lock-policy\)](#) (p. 191)

Get Vault Notifications (GET notification-configuration)

Description

This operation retrieves the `notification-configuration` subresource set on the vault (see [Set Vault Notification Configuration \(PUT notification-configuration\)](#) (p. 205). 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](#) (p. 47).

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 \(p. 150\)](#).

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 \(p. 153\)](#).

Response Body

The response body contains the following JSON fields.

Events

A list of one or more events for which 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 \(PUT notification-configuration\) \(p. 205\)](#).

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 \(p. 163\)](#).

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: 20141123T120000Z
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_XJC1-Q
Date: Sun, 23 Nov 2014 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 \(DELETE notification-configuration\)](#) (p. 179)
- [Set Vault Notification Configuration \(PUT notification-configuration\)](#) (p. 205)
- [Authentication and Access Control for Amazon Glacier](#) (p. 119)

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 [Amazon Glacier Access Control with Vault Lock Policies](#) (p. 132).

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 \(POST lockid\)](#) (p. 173) 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 abort the vault locking process by calling [Abort Vault Lock \(DELETE lock-policy\)](#) (p. 167). You can get the state of the vault lock by calling [Get Vault Lock \(GET lock-policy\)](#) (p. 186). For more information about the vault locking process, see [Amazon Glacier Vault Lock](#) (p. 60).

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 \(DELETE lock-policy\)](#) (p. 167) 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](#) (p. 150).

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 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 \(p. 153\)](#).

Name	Description
<i>x-amz-lock-id</i>	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 \(p. 163\)](#).

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: 20141123T120000Z
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: Sun, 23 Nov 2014 12:02:00 GMT
x-amz-lock-id: AE863rKkWZU53SLW5be4DUcW
```

Related Sections

- [Abort Vault Lock \(DELETE lock-policy\) \(p. 167\)](#)
- [Complete Vault Lock \(POST lockId\) \(p. 173\)](#)
- [Get Vault Lock \(GET lock-policy\) \(p. 186\)](#)

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 \(p. 144\)](#).

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 \(p. 150\)](#).

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 \(p. 153\)](#).

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 \(p. 163\)](#).

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: 20141123T120000Z
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 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_XJC1-Q
Date: Sun, 23 Nov 2014 12:02:00 GMT
Content-Type: application/json
Content-Length: length

{
  "Tags",
  {
    "examplekey1": "examplevalue1",
    "examplekey2": "examplevalue2"
  }
}
```

Related Sections

- [Add Tags To Vault \(POST tags add\)](#) (p. 169)
- [Remove Tags From Vault \(POST tags remove\)](#) (p. 201)

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 1,000 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 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
<code>limit</code>	The maximum number of vaults to be returned. The default limit is 1000. 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 1000.	No
<code>marker</code>	A string used for pagination. <code>marker</code> specifies the vault ARN after which the listing of vaults should begin. (The vault specified by <code>marker</code> is not included in the returned list.) Get the <code>marker</code> value from a previous List Vaults response. You need to include the <code>marker</code> 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 \(p. 150\)](#).

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 \(p. 153\)](#).

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 of 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 \(POST jobs\) \(p. 249\)](#).

Type: A string representation of 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](#) (p. 163).

Examples

Example: List All Vaults

The following example lists vaults. Because the `marker` and `limit` parameters are not specified in the request, up to 1,000 vaults are returned.

Example Request

```
GET /-/vaults HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20141123T120000Z
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: Sun, 23 Nov 2014 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: 20141123T120000Z
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: Sun, 23 Nov 2014 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 \(PUT vault\) \(p. 171\)](#)
- [Delete Vault \(DELETE vault\) \(p. 175\)](#)
- [Initiate Job \(POST jobs\) \(p. 249\)](#)
- [Authentication and Access Control for Amazon Glacier \(p. 119\)](#)

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 \(p. 144\)](#).

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
<code>operation=remove</code>	A single query string parameter <code>operation</code> with a value of <code>remove</code> to distinguish it from Add Tags To Vault (POST tags add) (p. 169) .	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 \(p. 150\)](#).

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 \(p. 153\)](#).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses \(p. 163\)](#).

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: 20141123T120000Z
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 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: Sun, 23 Nov 2014 12:02:00 GMT
```

Related Sections

- [Add Tags To Vault \(POST tags add\) \(p. 169\)](#)
- [List Tags For Vault \(GET tags\) \(p. 194\)](#)

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 [Amazon Glacier Access Control with Vault Access Policies \(p. 130\)](#).

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 \(p. 150\)](#).

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 \(p. 153\)](#).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses \(p. 163\)](#).

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: 20141123T120000Z
```

```
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 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: Sun, 23 Nov 2014 12:02:00 GMT
```

Related Sections

- [Delete Vault Access Policy \(DELETE access-policy\) \(p. 177\)](#)
- [Get Vault Access Policy \(GET access-policy\) \(p. 183\)](#)

Set Vault Notification Configuration (PUT notification-configuration)

Description

Retrieving an archive and a vault inventory are asynchronous operations in Amazon Glacier for which you must first initiate a job and wait for the job to complete before you can download the job output. Most Amazon Glacier jobs take about four hours to complete. 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 \(p. 47\)](#).

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 \(POST jobs\) \(p. 249\)](#)). 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 \(GET JobID\) \(p. 236\)](#).
- **InventoryRetrievalCompleted**— This event occurs when a job that was initiated for an inventory retrieval is completed ([Initiate Job \(POST jobs\) \(p. 249\)](#)). 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 \(GET JobID\)](#) (p. 236).

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](#) (p. 150).

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 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 \(p. 150\)](#).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses \(p. 163\)](#).

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: 20141123T120000Z
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_XJC1-Q
Date: Sun, 23 Nov 2014 12:00:00 GMT
```

Related Sections

- [Get Vault Notifications \(GET notification-configuration\)](#) (p. 189)
- [Delete Vault Notifications \(DELETE notification-configuration\)](#) (p. 179)
- [Authentication and Access Control for Amazon Glacier](#) (p. 119)

Archive Operations

The following are the archive operations available for use in Amazon Glacier.

Topics

- [Delete Archive \(DELETE archive\)](#) (p. 208)
- [Upload Archive \(POST archive\)](#) (p. 210)

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](#) (p. 34).

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, 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](#) (p. 79).

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 \(p. 150\)](#).

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 \(p. 153\)](#).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses \(p. 163\)](#).

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-
```

```
TjhgG6eGoOY9Z8i1_AUyUsubPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfG1qrEXAMPLEA
HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20141123T120000Z
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: Sun, 23 Nov 2014 12:00:00 GMT
```

Related Sections

- [Initiate Multipart Upload \(POST multipart-uploads\) \(p. 219\)](#)
- [Upload Archive \(POST archive\) \(p. 210\)](#)
- [Authentication and Access Control for Amazon Glacier \(p. 119\)](#)

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 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 \(p. 157\)](#).

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 \(p. 62\)](#).

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](#) (p. 150).

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) (p. 249)), 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 decimal or 0x20—0x7E hexadecimal.	No

Name	Description	Required
<code>x-amz-content-sha256</code>	The SHA256 checksum (a linear hash) of the payload. This is not the same value as you specify in the <code>x-amz-sha256-tree-hash</code> header. Type: String Default: None Constraints: None	Yes
<code>x-amz-sha256-tree-hash</code>	The user-computed checksum, SHA256 tree hash, of the payload. For information on computing the SHA256 tree hash, see Computing Checksums (p. 157) . If Amazon Glacier computes 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 \(p. 153\)](#).

Name	Description
<code>Location</code>	The relative URI path of the newly added archive resource. Type: String
<code>x-amz-archive-id</code>	The ID of the archive. This value is also included as part of the <code>Location</code> header. Type: String

Name	Description
<code>x-amz-sha256-tree-hash</code>	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 \(p. 163\)](#).

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: 20141123T120000Z
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: Sun, 23 Nov 2014 12:00:00 GMT
x-amz-sha256-tree-hash:
  beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60
Location: /111122223333/vaults/examplevault/
  archives/NkbByEejwEggmBz2fTHgJrg0XBODfjp4q6iu87-
  TjhqG6eGoOY9Z8i1_AUyUshPadTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEA
x-amz-archive-id: NkbByEejwEggmBz2fTHgJrg0XBODfjp4q6iu87-
  TjhqG6eGoOY9Z8i1_AUyUshPadTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEA
```

Related Sections

- [Working with Archives in Amazon Glacier \(p. 62\)](#)

- [Uploading Large Archives in Parts \(Multipart Upload\)](#) (p. 70)
- [Delete Archive \(DELETE archive\)](#) (p. 208)
- [Authentication and Access Control for Amazon Glacier](#) (p. 119)

Multipart Upload Operations

The following are the multipart upload operations available for use in Amazon Glacier.

Topics

- [Abort Multipart Upload \(DELETE uploadID\)](#) (p. 214)
- [Complete Multipart Upload \(POST uploadID\)](#) (p. 216)
- [Initiate Multipart Upload \(POST multipart-uploads\)](#) (p. 219)
- [List Parts \(GET uploadID\)](#) (p. 222)
- [List Multipart Uploads \(GET multipart-uploads\)](#) (p. 227)
- [Upload Part \(PUT uploadID\)](#) (p. 232)

Abort Multipart Upload (DELETE uploadID)

Description

This multipart upload operation aborts 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. Aborting a completed multipart upload fails. However, aborting an already-aborted upload will succeed, for a short time.

This operation is idempotent.

For information about multipart upload, see [Uploading Large Archives in Parts \(Multipart Upload\)](#) (p. 70).

Requests

To abort 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 \(p. 150\)](#).

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 \(p. 153\)](#).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses \(p. 163\)](#).

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: 20141123T120000Z
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: Sun, 23 Nov 2014 12:00:00 GMT
```

Related Sections

- [Initiate Multipart Upload \(POST multipart-uploads\) \(p. 219\)](#)

- [Upload Part \(PUT uploadID\)](#) (p. 232)
- [Complete Multipart Upload \(POST uploadID\)](#) (p. 216)
- [List Multipart Uploads \(GET multipart-uploads\)](#) (p. 227)
- [List Parts \(GET uploadID\)](#) (p. 222)
- [Uploading Large Archives in Parts \(Multipart Upload\)](#) (p. 70)
- [Authentication and Access Control for Amazon Glacier](#) (p. 119)

Complete Multipart Upload (POST uploadID)

Description

You call this multipart upload operation to inform 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 \(Multipart Upload\)](#) (p. 70).

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](#) (p. 157). 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 \(GET uploadID\)](#) (p. 222) 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 that causes an aborted connection 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](#) (p. 150).

Name	Description	Required
<i>x-amz-archive-size</i>	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
<i>x-amz-sha256-tree-hash</i>	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. Type: String Default: None Constraints: None	Yes

Request Elements

This operation does not use request elements.

Responses

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 \(p. 153\)](#).

Name	Description
<i>Location</i>	The relative URI path of the newly created archive. This URL includes the archive ID that is generated by Amazon Glacier. Type: String
<i>x-amz-archive-id</i>	The ID of the archive. This value is also included as part of the <i>Location</i> 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: 20141123T120000Z
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: Sun, 23 Nov 2014 12:00:00 GMT
Location: /111122223333/vaults/examplevault/
archives/NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhgG6eGoOY9Z8il_AUyUshuPadTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfG1qrEXAMPLEA
```



```
x-amz-archive-id: NkbByEejwEggmBz2fTHgJrg0XBODfjP4q6iu87-  
TjhgG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pT15nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEA
```

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 \(POST multipart-uploads\)](#) (p. 219)
- [Upload Part \(PUT uploadID\)](#) (p. 232)
- [Abort Multipart Upload \(DELETE uploadID\)](#) (p. 214)
- [List Multipart Uploads \(GET multipart-uploads\)](#) (p. 227)
- [List Parts \(GET uploadID\)](#) (p. 222)
- [Uploading Large Archives in Parts \(Multipart Upload\)](#) (p. 70)
- [Delete Archive \(DELETE archive\)](#) (p. 208)
- [Authentication and Access Control for Amazon Glacier](#) (p. 119)

Initiate Multipart Upload (POST multipart-uploads)

Description

This operation initiates a multipart upload (see [Uploading Large Archives in Parts \(Multipart Upload\)](#) (p. 70)). 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 megabyte (1024 KB) multiplied by a power of 2—for example, 1048576 (1 MB), 2097152 (2 MB), 4194304 (4 MB), 8388608 (8 MB), and so on. The minimum allowable part size is 1 MB, and the maximum is 4 GB.

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 MB file. If you initiate the multipart upload with a part size of 4 MB, you will upload four parts of 4 MB each and one part of 0.2 MB.

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](#) (p. 150).

Name	Description	Required
<i>x-amz-part-size</i>	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 megabyte (1024 KB) multiplied by a power of 2—for example, 1048576 (1 MB), 2097152 (2 MB), 4194304 (4 MB), 8388608 (8 MB), and so on. The minimum allowable part size is 1 MB, and the maximum is 4 GB (4096 MB).	Yes
<i>x-amz-archive-description</i>	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) (p. 249)), the inventory includes this description for each of the archives it returns in response. Leading whitespace in archive descriptions is removed. Type: String 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.	No

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 \(p. 153\)](#).

Name	Description
<i>Location</i>	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. Type: String
<i>x-amz-multipart-upload-id</i>	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 \(p. 163\)](#).

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 MB (4194304 bytes) and the optional archive description.

```
POST /-/vaults/examplevault/multipart-uploads
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20141123T120000Z
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_XJC1-Q
Date: Sun, 23 Nov 2014 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 \(PUT uploadID\)](#) (p. 232).

Related Sections

- [Upload Part \(PUT uploadID\)](#) (p. 232)
- [Complete Multipart Upload \(POST uploadID\)](#) (p. 216)
- [Abort Multipart Upload \(DELETE uploadID\)](#) (p. 214)
- [List Multipart Uploads \(GET multipart-uploads\)](#) (p. 227)
- [List Parts \(GET uploadID\)](#) (p. 222)
- [Delete Archive \(DELETE archive\)](#) (p. 208)
- [Uploading Large Archives in Parts \(Multipart Upload\)](#) (p. 70)
- [Authentication and Access Control for Amazon Glacier](#) (p. 119)

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 \(Multipart Upload\)](#) (p. 70).

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 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 \(POST multipart-uploads\) \(p. 219\)](#)). 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
<i>limit</i>	<p>The maximum number of parts to be returned. The default limit is 1000. The number of parts returned might be fewer than the specified limit, but the number of returned parts never exceeds the limit.</p> <p>Type: String</p> <p>Constraints: Minimum integer value of 1. Maximum integer value of 1000.</p>	No
<i>marker</i>	<p>An opaque string used for pagination. <code>marker</code> specifies the part at which the listing of parts should begin. Get the <code>marker</code> value from the response of a previous List Parts response. You need only include the <code>marker</code> if you are continuing the pagination of results started in a previous List Parts request.</p> <p>Type: String</p> <p>Constraints: None</p>	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 \(p. 153\)](#).

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 \(p. 153\)](#).

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 of 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](#) (p. 163).

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_396UW9zLFUWVrnRHapjUJddQ50xSHVXjYtrN47NBZ-  
khxOjyEXAMPLE HTTP/1.1  
Host: glacier.us-west-2.amazonaws.com  
x-amz-Date: 20141123T120000Z  
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: Sun, 23 Nov 2014 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: 20141123T120000Z
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: Sun, 23 Nov 2014 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 \(POST multipart-uploads\) \(p. 219\)](#)
- [Upload Part \(PUT uploadID\) \(p. 232\)](#)
- [Complete Multipart Upload \(POST uploadID\) \(p. 216\)](#)
- [Abort Multipart Upload \(DELETE uploadID\) \(p. 214\)](#)
- [List Multipart Uploads \(GET multipart-uploads\) \(p. 227\)](#)
- [Uploading Large Archives in Parts \(Multipart Upload\) \(p. 70\)](#)
- [Authentication and Access Control for Amazon Glacier \(p. 119\)](#)

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 \(POST multipart-uploads\) \(p. 219\)](#) request, but has not yet been completed or aborted. 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 1,000 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 returned in response to your previous List Multipart Uploads request.

Note the difference between this operation and the [List Parts \(GET uploadID\) \(p. 222\)](#) 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 \(Multipart Upload\) \(p. 70\)](#).

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
<i>limit</i>	Specifies the maximum number of uploads returned in the response body. If not specified, the List Uploads operation returns up to 1,000 uploads. Type: String Constraints: Minimum integer value of 1. Maximum integer value of 1000.	No
<i>marker</i>	An opaque string used for pagination. <i>marker</i> specifies the upload at which the listing of uploads should begin. Get the <i>marker</i> value from a previous List Uploads response. You need only include the <i>marker</i> 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 \(p. 153\)](#).

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](#) (p. 153).

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 of 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 \(POST multipart-uploads\)](#) (p. 219) 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](#) (p. 163).

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: 20141123T120000Z
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: Sun, 23 Nov 2014 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_zqwB7zLDMesWhwo65re4C",
      "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-
_jNyM6RjVMFfV291FqZ3rNsSaWBugg6OP92pRtufeHdQH7ClIpSF6uJc",
      "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_7yO8gVIONIBsAxx2t-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_zqwB7zLD  
 HTTP/1.1  
Host: glacier.us-west-2.amazonaws.com  
x-amz-Date: 20141123T120000Z  
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 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: Sun, 23 Nov 2014 12:00:00 GMT  
Content-Type: application/json  
Content-Length: 470  
  
{  
  "Marker": "qt-RBst_7yO8gVIONIBsAxx2t-db0pE4s8MNeGjKjGdNpuU-  
cdSACqG62guwV9r5jh5mLyFPzFEitTpNE7iQfHiu1XoV",  
  "UploadsList" : [  
    {  
      "ArchiveDescription": "archive 2",  
      "CreationDate": "2012-04-01T15:00:00.000Z",  
      "MultipartUploadId": "nPyGOnyFcx67qqX7E-0tSGiRi88hHMOwOxR-  
_jNyM6RjVMFfV291FqZ3rNsSaWbugg6OP92pRtufeHdQH7ClIpsF6uJc",  
      "PartSizeInBytes": 4194304,  
      "VaultARN": "arn:aws:glacier:us-west-2:012345678901:vaults/  
examplevault"  
    }  
  ]  
}
```

```
} ]
```

Related Sections

- [Initiate Multipart Upload \(POST multipart-uploads\)](#) (p. 219)
- [Upload Part \(PUT uploadID\)](#) (p. 232)
- [Complete Multipart Upload \(POST uploadID\)](#) (p. 216)
- [Abort Multipart Upload \(DELETE uploadID\)](#) (p. 214)
- [List Parts \(GET uploadID\)](#) (p. 222)
- [Uploading Large Archives in Parts \(Multipart Upload\)](#) (p. 70)
- [Authentication and Access Control for Amazon Glacier](#) (p. 119)

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 \(Multipart Upload\)](#) (p. 70).

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](#) (p. 157).
- **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](#) (p. 157).
- **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 \(POST multipart-uploads\)](#) (p. 219) 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](#) (p. 150).

Name	Description	Required
<code>Content-Length</code>	Identifies the length of the part in bytes. Type: String Default: None Constraints: None	No
<code>Content-Range</code>	Identifies the range of bytes in the assembled archive that will be uploaded in this part. Amazon Glacier uses this information to assemble the archive in the proper sequence. The format of this header follows RFC 2616 . An example header is <code>Content-Range:bytes 0-4194303/*</code> . Type: String Default: None Constraints: The range cannot be greater than the part size that you specified when you initiated the multipart upload.	Yes

Name	Description	Required
<code>x-amz-content-sha256</code>	The SHA256 checksum (a linear hash) of the uploaded payload. This is not the same value as you specify in the <code>x-amz-sha256-tree-hash</code> header. Type: String Default: None Constraints: None	Yes
<code>x-amz-sha256-tree-hash</code>	Specifies a SHA256 tree hash of the data being uploaded. For information about computing a SHA256 tree hash, see Computing Checksums (p. 157) . 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 \(p. 153\)](#).

Name	Description
<code>x-amz-sha256-tree-hash</code>	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: Sun, 23 Nov 2014 12:00:00 GMT  
Content-Range: bytes 0-4194303/*  
x-amz-sha256-tree-hash: c06f7cd4baacb087002a99a5f48bf953  
x-amz-  
contentsha256: 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: Sun, 23 Nov 2014 12:00:00 GMT  
Content-Range: bytes 4194304-8388607/*  
Content-Length: 4194304  
x-amz-sha256-tree-hash: f10e02544d651e2c3ce90a4307427493  
x-amz-  
contentsha256: 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: Sun, 23 Nov 2014 12:00:00 GMT
```

Related Sections

- [Initiate Multipart Upload \(POST multipart-uploads\) \(p. 219\)](#)

- [Upload Part \(PUT uploadID\)](#) (p. 232)
- [Complete Multipart Upload \(POST uploadID\)](#) (p. 216)
- [Abort Multipart Upload \(DELETE uploadID\)](#) (p. 214)
- [List Multipart Uploads \(GET multipart-uploads\)](#) (p. 227)
- [List Parts \(GET uploadID\)](#) (p. 222)
- [Uploading Large Archives in Parts \(Multipart Upload\)](#) (p. 70)
- [Authentication and Access Control for Amazon Glacier](#) (p. 119)

Job Operations

The following are the job operations available in Amazon Glacier.

Topics

- [Describe Job \(GET JobID\)](#) (p. 236)
- [Get Job Output \(GET output\)](#) (p. 243)
- [Initiate Job \(POST jobs\)](#) (p. 249)
- [List Jobs \(GET jobs\)](#) (p. 257)

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 completes the job. For more information about initiating a job, see [Initiate Job \(POST jobs\)](#) (p. 249).

Note

This operation enables you to check the status of your job. However, it is strongly recommended 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 \(GET jobs\) \(p. 257\)](#).

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 \(p. 150\)](#).

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,
  "ArchiveSizeInBytes": Number,
  "ArchiveSHA256TreeHash": String,
  "Completed": Boolean,
  "CompletionDate": String,
  "CreationDate": String,
  "InventorySizeInBytes": Number,
  "JobDescription": String,
  "JobId": String,
  "RetrievalByteRange": String,
  "SHA256TreeHash": String,
  "SNSTopic": String,
  "StatusCode": String,
  "StatusMessage": String,
  "Tier": String,
  "VaultARN": String,
  "InventoryRetrievalParameters": {
    "Format": String,
    "StartDate": String,
    "EndDate": String,
    "Limit": 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 \(p. 153\)](#).

Response Body

The response body contains the following JSON fields.

Action

The job type. It is either `ArchiveRetrieval` or `InventoryRetrieval`.

Type: String

Archiveld

For an `ArchiveRetrieval` job, this is the archive ID requested for download. Otherwise, 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

ArchiveSHA256TreeHash

The SHA256 tree hash of the entire archive for an archive retrieval job. For inventory retrieval jobs, this field is `null`.

Type: String

Completed

The job status. When a job is completed you get the job's output using the [Get Job Output \(GET output\) \(p. 243\)](#).

Type: Boolean

CompletionDate

The UTC time that the job request completed. While the job is in progress, the value will be `null`.

Type: String

CreationDate

The UTC time that the job was created.

Type: A string representation of ISO 8601 date format, for example, `2013-03-20T17:03:43.221Z`.

EndDate

The end of the date range in UTC for vault inventory retrieval that includes archives created before this date.

Valid Values: A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

Type: String. A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

Format

The output format for the vault inventory list, which is set by the [Initiate Job \(POST jobs\) \(p. 249\)](#) request when initiating a job to retrieve a vault inventory.

Valid Values: CSV | JSON

Required: no

Type: String

InventorySizeInBytes

For an `InventoryRetrieval` job, this is the size in bytes of the inventory requested for download. For the `ArchiveRetrieval` job, the value is `null`.

Type: Number

JobDescription

The job description you provided when you initiated the job.

Type: String

JobId

The ID that represents the job in Amazon Glacier.

Type: String

Limit

Specifies the maximum number of inventory items returned per vault inventory retrieval request. This limit is set when initiating the job with the a [Initiate Job \(POST jobs\) \(p. 249\)](#) request.

Valid Values: An integer value greater than or equal to 1.

Type: String

Marker

An opaque string that represents where to continue pagination of the vault inventory retrieval results. You use the marker in a new `Initiate Job` request to obtain additional inventory items. If there are no more inventory items, this value is `null`. For information about using the marker, see [Range Inventory Retrieval \(p. 250\)](#).

Type: String

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 jobs, this field is `null`.

Type: String

SHA256TreeHash

The SHA256 tree hash value for the requested range of an archive. If the [Initiate Job \(POST jobs\) \(p. 249\)](#) 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 \(p. 162\)](#).

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.

Type: String

SNSTopic

An Amazon Simple Notification Service (Amazon SNS) topic that receives notification.

Type: String

StartDate

The start of the date range in UTC for vault inventory retrieval that includes archives created on or after this date.

Valid Values: A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

Type: String. A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

StatusCode

The status code can be `InProgress`, `Succeeded`, or `Failed`, and indicates the status of the job.

Type: String

StatusMessage

A friendly message that describes the job status.

Type: String

Tier

The retrieval option to use for the archive retrieval.

Valid Values: `Bulk` | `Expedited` | `Standard`

Type: String

VaultARN

The Amazon Resource Name (ARN) of the vault from which the archive retrieval was requested.

Type: String

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses](#) (p. 163).

Examples

The following example shows the request for a job that retrieves an archive.

Example Request: Get job description

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0j1b5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20141123T120000Z
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: Sun, 23 Nov 2014 12:00:00 GMT
Content-Type: application/json
Content-Length: 419
{
  "Action": "ArchiveRetrieval",
  "ArchiveId": "NkbByEejwEggmBz2fTHgJrg0XBodfjP4q6iu87-
TjhgG6eGoOY9Z8il_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfG1qrEXAMPLE",
  "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, has not fully prepared the inventory for download. If the job was complete prior to 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_uykQXFMLa
"},
  }
}

```

Related Sections

- [Get Job Output \(GET output\) \(p. 243\)](#)
- [Authentication and Access Control for Amazon Glacier \(p. 119\)](#)

Get Job Output (GET output)

Description

This operation downloads the output of the job you initiated using [Initiate Job \(POST jobs\)](#) (p. 249). 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. 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](#) (p. 150).

Name	Description	Required
<i>Range</i>	<p>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 <code>bytes=0-1048575</code>. For more information, go to Range Header Field Definition. The range is relative to any range specified in the Initiate Job request. By default, this operation downloads the entire output.</p> <p>If the job output is large, then you can use the <code>Range</code> 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:</p> <ol style="list-style-type: none"> 1. Download a 128 MB chunk of output by specifying the appropriate byte range using the <code>Range</code> 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) (p. 236) 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. <p>Type: String Default: None Constraints: None</p>	No

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
<i>Content-Range</i>	<p>The range of bytes returned by Amazon Glacier. If only partial output is downloaded, the response provides the range of bytes Amazon Glacier returned.</p> <p>For example, bytes 0-1048575/8388608 returns the first 1 MB from 8 MB.</p> <p>For more information about the <i>Content-Range</i> header, go to Content-Range Header Field Definition.</p> <p>Type: String</p>
<i>Content-Type</i>	<p>The Content-Type depends on whether the job output is an archive or a vault inventory.</p> <ul style="list-style-type: none"> For archive data, the Content-Type is <code>application/octet-stream</code>. For vault inventory, if you requested CSV format when you initiated the job, the Content-Type is <code>text/csv</code>. Otherwise, by default, vault inventory is returned as JSON, and the Content-Type is <code>application/json</code>. <p>Type: String</p>
<i>x-amz-sha256-tree-hash</i>	<p>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 information about tree hash aligned ranges, see Receiving Checksums When Downloading Data (p. 162).</p> <p>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:</p> <ul style="list-style-type: none"> 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 <i>x-amz-sha256-tree-hash</i> 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

Header	Description
	<p>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 <code>x-amz-sha256-tree-hash</code> is returned as a response header.</p> <p>Type: String</p>

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 of 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 of 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](#) (p. 163).

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-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYws0j1b5gq1JZ55yHgt5vP54ZShjoQzQVvh7vEXAMPLEjobID/
output HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20141123T120000Z
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_XJC1-Q
x-amz-sha256-tree-hash:
  beb0fe31a1c7ca8c6c04d574ea906e3f97b31fdca7571defb5b44dca89b5af60
Date: Sun, 23 Nov 2014 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_XJC1-Q
Date: Sun, 23 Nov 2014 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_9VBWrW4Jw4zsvg5kehAPDVK
oA",
      "ArchiveDescription": "my archive1",
      "CreationDate": "2012-05-15T17:19:46.700Z",
      "Size": 2140123,
      "SHA256TreeHash":
"6b9d4cf8697bd3af6aa1b590a0b27b337da5b18988dbcc619a3e608a554a1e62"
    },
    {
      "ArchiveId": "2lHzwhKhgF2JHyvCS-
ZRuF08IQLuyB4265Hs3AXj9MoAIhz7tbXAvCFeHusGU_hViO1WeCBe0N5lsYYHRyZ7rrmRkNRuYrXUs_sjl2K8ume_7
uHEloHqaW9d37pabXrSA",
      "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-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYws0j1b5gq1JZ55yHgt5vP54ZShjoQzQVvh7vEXAMPLEjobID/
output HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20141123T120000Z
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: Sun, 23 Nov 2014 12:00:00 GMT
Content-Range: bytes 0-1023/8388608
Content-Type: application/octet-stream
Content-Length: 1024

[Archive data.]
```

Related Sections

- [Describe Job \(GET JobID\) \(p. 236\)](#)
- [Initiate Job \(POST jobs\) \(p. 249\)](#)
- [Authentication and Access Control for Amazon Glacier \(p. 119\)](#)

Initiate Job (POST jobs)

This operation initiates a job of the specified type, which can be an archive retrieval or vault inventory retrieval.

Initializing a Data Retrieval Job

Retrieving an archive or a vault inventory are asynchronous operations that require you to initiate a job. Retrieval is a two-step process:

1. Initiate a retrieval job.

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 \(p. 140\)](#). For more information about the `PolicyEnforcedException` exception, see [Error Responses \(p. 163\)](#).

2. After the job completes, download the bytes.

The retrieval request is executed 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 \(GET output\) \(p. 243\)](#) 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 Simple Notification Service (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 \(PUT notification-configuration\) \(p. 205\)](#).
- **Get job details**— You can make a [Describe Job \(GET JobID\) \(p. 236\)](#) 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 \(GET JobID\) \(p. 236\)](#).

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 \(PUT notification-configuration\) \(p. 205\)](#).

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. 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. For more information about the data fields returned in an inventory job output, see [Response Body \(p. 246\)](#).

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` will be returned. If you only provide the `StartDate` without the `EndDate`, you will retrieve the inventory for all archives created on or after the `StartDate`. If you only provide the `EndDate` without the `StartDate`, you will 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 will contain 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 \(GET JobID\) \(p. 236\)](#) operation to get a marker that you use in a subsequent **Initiate**

Job request. The marker will indicate 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, 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; that is, 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 you get checksum values for data you download using Get Job Output ([Get Job Output \(GET output\)](#) (p. 243)), the range must be tree-hash aligned. For more information about tree-hash aligned ranges, see [Receiving Checksums When Downloading Data](#) (p. 162).

Expedited and Bulk Archive Retrievals

When retrieving an archive, you can specify one of the following options in the `Tier` field of the request body:

- **Expedited** - Expedited retrievals allow you to quickly access your data when occasional urgent requests for a subset of archives are required. For all but the largest archives (250MB+), data accessed using Expedited retrievals are typically made available within 1–5 minutes.
- **Standard** - Standard retrievals allow you to access any of your archives within several hours. Standard retrievals typically complete within 3–5 hours. This is the default option for retrieval requests that do not specify the retrieval option.
- **Bulk** - Bulk retrievals are Amazon Glacier lowest-cost retrieval option, enabling you to retrieve large amounts, even petabytes, of data inexpensively in a day. Bulk retrievals typically complete within 5–12 hours.

For more information about expedited and bulk retrievals, see [Retrieving Amazon Glacier Archives](#) (p. 79).

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 and you can optionally 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 does not already exist. Amazon Glacier does not create the topic for you. The topic must have permissions to receive publications from an Amazon Glacier vault. Amazon Glacier does not 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 syntax is for an archive retrieval.

```
POST /AccountId/vaults/VaultName/jobs HTTP/1.1
Host: glacier.Region.amazonaws.com
```

```
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01

{
  "Type": "archive-retrieval",
  "ArchiveId": String,
  "Description": String,
  "RetrievalByteRange": String,
  "SNSTopic": String,
  "Tier": String
}
```

The following syntax is for an inventory retrieval.

```
POST /AccountId/vaults/VaultName/jobs HTTP/1.1
Host: glacier.Region.amazonaws.com
Date: Date
Authorization: SignatureValue
x-amz-glacier-version: 2012-06-01

{
  "Type": "inventory-retrieval",
  "Description": String,
  "Format": String,
  "SNSTopic": String,
  "InventoryRetrievalParameters": {
    "StartDate": String,
    "EndDate": String,
    "Limit": String,
    "Marker": 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 JSON in the request body contains the following fields.

ArchiveId

The ID of the archive that you want to retrieve. This field is required only if `Type` is set to `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.

Required: Yes when `Type` is set to `archive-retrieval`.

Type: String

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.

Required: no

Type: String

EndDate

The end of the date range in UTC for vault inventory retrieval that includes archives created before this date.

Valid Values: A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

Required: no

Type: String. A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

Format

When initiating a job to retrieve a vault inventory, you can optionally add this parameter to your request to specify the output format. If you are initiating an inventory job and do not specify a `Format` field, JSON is the default format.

Valid Values: CSV | JSON

Required: no

Type: String

Limit

The maximum number of inventory items returned per vault inventory retrieval request.

Valid Values: An integer value greater than or equal to 1.

Required: no

Type: String

Marker

An opaque string that represents where to continue pagination of the vault inventory retrieval results. You use the marker in a new `Initiate Job` request to obtain additional inventory items. If there are no more inventory items, this value is null.

Required: no

Type: String

RetrievalByteRange

The byte range to retrieve for an archive retrieval. in the form "*StartByteValue-EndByteValue*". If not specified, the whole archive is retrieved. If specified, the byte range must be megabyte (1024*1024) aligned, which means that *StartByteValue* must be divisible by 1 MB, and the *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 job request.

Required: no

Type: String

SNSTopic

The Amazon SNS topic ARN where Amazon Glacier sends a notification when the job is completed, and the output is ready for you to download. The specified topic publishes the

notification to its subscribers. The SNS topic must exist. If it does not, Amazon Glacier does not 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*.

Required: no

Type: String

StartDate

The start of the date range in UTC for vault inventory retrieval that includes archives created on or after this date.

Valid Values: A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

Required: no

Type: String. A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

Tier

The retrieval option to use for the archive retrieval. `Standard` is the default value used.

Valid Values: `Expedited` | `Standard` | `Bulk`

Required: no.

Type: String

Type

The job type. You can initiate a job to retrieve an archive or get an inventory of a vault.

Valid Values: `archive-retrieval` | `inventory-retrieval`

Required: yes

Type: String

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
```

Response Headers

Header	Description
<code>Location</code>	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) (p. 236). Type: String Default: None


```
"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_XJC1-Q
Date: Sun, 23 Nov 2014 12:00:00 GMT
Location: /111122223333/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYws0j1b5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
x-amz-job-id: HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYws0j1b5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

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: 20141123T120000Z
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_XJC1-Q
Date: Sun, 23 Nov 2014 12:00:00 GMT
Location: /111122223333/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYws0j1b5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
x-amz-job-id: HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYws0j1b5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

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 \(GET JobID\)](#) (p. 236).

```
{
  "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_uykQXFMLa
  },
}
```

Related Sections

- [Describe Job \(GET JobID\)](#) (p. 236)
- [Get Job Output \(GET output\)](#) (p. 243)
- [Authentication and Access Control for Amazon Glacier](#) (p. 119)

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 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 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.

To retrieve an archive or retrieve a vault inventory from Amazon Glacier, you first initiate a job, and after the job completes, you download the data. For an archive retrieval, the output is the archive data. For an inventory retrieval, it is the inventory list. The List Job operation returns a list of these jobs sorted by job initiation time.

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 1000. 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
<code>completed</code>	The state of the jobs to return. You can specify <code>true</code> or <code>false</code> . Type: Boolean Constraints: None	No
<code>limit</code>	The maximum number of jobs to be returned. The default limit is 1000. 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 1000.	No

Name	Description	Required
<i>marker</i>	An opaque string used for pagination that specifies the job at which the listing of jobs should begin. You get the <code>marker</code> value from a previous List Jobs response. You only need to include the <code>marker</code> if you are continuing the pagination of the results started in a previous List Jobs request. Type: String Constraints: None	No
<i>statuscode</i>	The type of job status to return. Type: String Constraints: One of the values <code>InProgress</code> , <code>Succeeded</code> , or <code>Failed</code> .	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 \(p. 153\)](#).

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,
      "ArchiveSizeInBytes": Number,
      "ArchiveSHA256TreeHash": String,
      "Completed": Boolean,
      "CompletionDate": String,
      "CreationDate": String,
      "InventorySizeInBytes": String,
      "JobDescription": String,
      "JobId": String,
      "RetrievalByteRange": String,
      "SHA256TreeHash": String,
      "SNSTopic": String,
      "StatusCode": String,
      "StatusMessage": String,
    }
  ]
}
```

```
"Tier": String,
"VaultARN": String,
"InventoryRetrievalParameters": {
  "Format": String,
  "StartDate": String,
  "EndDate": String,
  "Limit": String,
  "Marker": 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 \(p. 153\)](#).

Response Body

The response body contains the following JSON fields.

Action

For archive retrieval jobs, this value is `ArchiveRetrieval`. For inventory retrieval jobs, this value is `InventoryRetrieval`.

Type: String

ArchiveId

The ID of the archive that was requested in an archive retrieval. This field only appears for archive retrieval job descriptions.

Type: String

ArchiveSizeInBytes

The size of the archive for which the archive retrieval job request was initiated.

Type: Number

ArchiveSHA256TreeHash

The SHA256 tree hash of the entire archive for an archive retrieval. For inventory retrieval jobs, this field is `null`.

Type: String

Completed

`true` if the job is completed; `false` otherwise.

Type: Boolean

CompletionDate

The Universal Coordinated Time (UTC) date when the job completed.

Type: A string representation of 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 of ISO 8601 date format, for example, `2013-03-20T17:03:43.221Z`.

EndDate

The end of the date range in UTC for vault inventory retrieval that includes archives created before this date.

Valid Values: A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

Type: String. A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

Format

The output format for the vault inventory list, which is set by the [Initiate Job \(POST jobs\) \(p. 249\)](#) request when initiating a job to retrieve a vault inventory.

Valid Values: CSV | JSON

Required: no

Type: String

InventorySizeInBytes

The size of the inventory associated with an inventory retrieval job request. This field appears only for inventory retrieval job descriptions.

Type: Number

JobDescription

A description of the job.

Type: String

JobId

The ID that represents the job in Amazon Glacier.

Type: String

JobList

An array of job objects. Each job object contains a set of name-value pairs describing the job.

Type: Array

Limit

Specifies the maximum number of inventory items returned per vault inventory retrieval request. This limit is set when initiating the job with an [Initiate Job \(POST jobs\) \(p. 249\)](#) request.

Valid Values: An integer value greater than or equal to 1.

Type: String

Marker (InventoryRetrievalParameters)

An opaque string that represents where to continue pagination of the vault inventory retrieval results. You use the marker in a new `Initiate Job` request to obtain additional inventory items. If there are no more inventory items, this value is `null`. For information about using the marker, see [Range Inventory Retrieval \(p. 250\)](#).

Type: String

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

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

SHA256TreeHash

The SHA256 tree hash value for the requested range of an archive. If the [Initiate Job \(POST jobs\)](#) (p. 249) request for an archive specified a tree-hash aligned range, then this field returns a value. For more information about tree hash aligned ranges for archive range retrievals, see [Receiving Checksums When Downloading Data](#) (p. 162).

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 not equal to the whole archive and the job status is `InProgress`. After the job completes, the `SHA256TreeHash` field will have a value.
- Inventory 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 \(POST jobs\)](#) (p. 249)).

Type: String

StartDate

The start of the date range in UTC for vault inventory retrieval that includes archives created on or after this date.

Valid Values: A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

Type: String. A string representation of ISO 8601 date format `YYYY-MM-DDThh:mm:ssTZD` in seconds, for example `2013-03-20T17:03:43Z`.

StatusCode

The job status code. The values can be `Succeeded`, `Failed`, or `InProgress`.

Type: String

StatusMessage

The job status message.

Type: String

Tier

The retrieval option to use for the archive retrieval.

Valid Values: `Expedited` | `Standard` | `Bulk`

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](#) (p. 163).

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: 20141123T120000Z
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_XJC1-Q
Date: Sun, 23 Nov 2014 12:00:00 GMT
Content-Type: application/json
Content-Length: 1444

{
  "JobList": [
    {
      "Action": "ArchiveRetrieval",
      "ArchiveId": "BDfaUQul0dVzYwAMr8YSa_6_8abbhZq-
iloT69g8ByClfJyBgAGBkWl2QbF5os851P7Y7KdZDOHWJIn4rh1ZHaOYD3MgFhK_g0oDPesW34uHQoVGwoIqubf6BgU",
      "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":
"2cvVOnBL36btzyP3pobwIceiaJebM1bx9vZ0OUtmNAR0KaVZ4WkWGvjIpldJ73VU7imlm0pnZriBVBebnqaAcirZq
"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_uykQXFMLa
    }
  },
  "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-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0j1b5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID&limit=2
HTTP/1.1
Host: glacier.us-west-2.amazonaws.com
x-amz-Date: 20141123T120000Z
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: Sun, 23 Nov 2014 12:00:00 GMT
Content-Type: application/json
Content-Length: 1744

{
  "JobList": [
    {
      "Action": "ArchiveRetrieval",
      "ArchiveId": "58-3KpZfcMPUzvnMZNPaKyJx9wODCsWTnqcjtx2CjKZ6b-
XgxEuA8yvZYOYTPQfd7gWR4GRm2XR08gcnWbLV4VPV_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": "s4MvaNHih6mOalf8iY4ioG2921SDPihXxh3Kv0FBX-
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_WAdwF61R6w8iSyKdvw",
      "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_domLOX5k8ItFv0wCPN0oaz",
      "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_tf6fOR4jrJCL6lMfk6VM03oY8lmnWK93KK4gLig1UPAbZiN3UV4G_5nq4AfmJHQ_dOMLOX5k8ItFv0wCPN0oaz"  
}
```

Related Sections

- [Describe Job \(GET JobID\) \(p. 236\)](#)
- [Authentication and Access Control for Amazon Glacier \(p. 119\)](#)

Data Retrieval Operations

The following are the data retrieval related operations available in Amazon Glacier.

Topics

- [Get Data Retrieval Policy \(GET policy\) \(p. 266\)](#)
- [List Provisioned Capacity \(GET provisioned-capacity\) \(p. 269\)](#)
- [Purchase Provisioned Capacity \(POST provisioned-capacity\) \(p. 271\)](#)
- [Set Data Retrieval Policy \(PUT policy\) \(p. 273\)](#)

Get Data Retrieval Policy (GET policy)

Description

This operation returns the current data retrieval policy for the account and region specified in the GET request. For more information about data retrieval policies, see [Amazon Glacier Data Retrieval Policies \(p. 140\)](#).

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 \(p. 150\)](#).

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 \(p. 153\)](#).

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](#) (p. 140).

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses](#) (p. 163).

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: 20141123T120000Z
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: Sun, 23 Nov 2014 12:00:00 GMT
Content-Type: application/json
Content-Length: 85

{
  "Policy":
  {
    "Rules": [
      {
        "BytesPerHour": 10737418240,
        "Strategy": "BytesPerHour"
      }
    ]
  }
}
```

Related Sections

- [Set Data Retrieval Policy \(PUT policy\)](#) (p. 273)
- [Initiate Job \(POST jobs\)](#) (p. 249)

List Provisioned Capacity (GET provisioned-capacity)

This operation lists the provisioned capacity for the specified AWS account.

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 \(p. 150\)](#).

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 \(p. 153\)](#).

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 of 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 of 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 \(p. 163\)](#).

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: 20141123T120000Z
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 a HTTP 200 OK with a list of provisioned capacity units for the account as shown in the following example.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 23 Nov 2014 12:02:00 GMT
```

```
Content-Type: application/json
Content-Length: length

{
  "ProvisionedCapacityList",
  {
    "CapacityId": "zSaq7NzHFQDANTfQkDen4V7z",
    "StartDate": "2016-11-11T20:11:51.095Z",
    "ExpirationDate": "2016-12-12T00:00:00.000Z",
  },
  {
    "CapacityId": "yXaq7NzHFQDANTfQkDen4V7z",
    "StartDate": "2016-12-13T20:11:51.095Z",
    "ExpirationDate": "2017-01-15T00:00:00.000Z",
  },
  ...
}
```

Related Sections

- [Purchase Provisioned Capacity \(POST provisioned-capacity\) \(p. 271\)](#)

Purchase Provisioned Capacity (POST provisioned-capacity)

This operation purchases a provisioned capacity unit for an 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 \(p. 150\)](#).

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 \(p. 153\)](#).

Name	Description
<i>x-amz-capacity-id</i>	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(s), 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 \(p. 163\)](#).

Code	Description	HTTP Status Code	Type
<code>LimitExceededException</code>	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: 20141123T120000Z
```

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20141123/us-west-2/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-version,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f52a2
Content-Length: length
x-amz-glacier-version: 2012-06-01
```

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: Sun, 23 Nov 2014 12:02:00 GMT
x-amz-capacity-id: zSaq7NzHFQDANTfQkDen4V7z
```

Related Sections

- [List Provisioned Capacity \(GET provisioned-capacity\) \(p. 269\)](#)

Set Data Retrieval Policy (PUT policy)

Description

This operation sets and then enacts a data retrieval policy in the region specified in the `PUT` request. You can set one policy per 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 \(p. 140\)](#).

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 \(p. 150\)](#).

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 \(p. 140\)](#).

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 \(p. 153\)](#).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see [Error Responses \(p. 163\)](#).

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: 20141123T120000Z  
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: 20141123T120000Z  
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",  
          "FreeTier": 10737418240  
        }  
      ]  
    }  
}
```

```
    {
      "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: 20141123T120000Z
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 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_XJC1-Q
Date: Sun, 23 Nov 2014 12:02:00 GMT
```

Related Sections

- [Get Data Retrieval Policy \(GET policy\) \(p. 266\)](#)
- [Initiate Job \(POST jobs\) \(p. 249\)](#)

Document History

The following table describes the important changes since the last release of the *Amazon Glacier Developer Guide*.

Relevant Dates to this History:

- **Current product version:** 2012-06-01
- **Last documentation update:** November 21, 2016

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 (p. 81) .	In this release
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 (p. 60) and Amazon Glacier Access Control with Vault Lock Policies (p. 132) .	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 (p. 144) and Tagging Your Amazon Glacier Vaults (p. 58) .	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 Amazon Glacier Access Control with Vault Access Policies (p. 130) .	April 27, 2015

Change	Description	Release Date
Data retrieval policies and audit logging	<p>Amazon Glacier now supports data retrieval policies and audit logging. Data retrieval policies allow you to easily set data retrieval limits and simplify data retrieval cost management. You can define your own data retrieval limits with a few clicks in the AWS console or by using the Amazon Glacier API. For more information, see Amazon Glacier Data Retrieval Policies (p. 140).</p> <p>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 by Using AWS CloudTrail (p. 146).</p>	December 11, 2014
Updates to Java samples	Updated the Java code samples 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 (p. 250) in the Initiate Job (POST jobs) (p. 249) 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	<p>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.</p> <p>For more information, see Downloading an Archive in Amazon Glacier (p. 79).</p> <p>For pricing information, go to the Amazon Glacier detail page.</p>	November 13, 2012
New Guide	This is the first release of the <i>Amazon Glacier Developer Guide</i> .	August 20, 2012

AWS Glossary

For the latest AWS terminology, see the [AWS Glossary](#) in the *AWS General Reference*.