Amazon Glacier

Developer Guide

API Version 2012-06-01

Amazon Glacier: Developer Guide

Table of Contents

What Is Amazon Glacier? ... 1
- Are You a First-Time Amazon Glacier User? ... 1
- Data Model ... 2
  - Vault ... 2
  - 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
  - Amazon Glacier Data Retrieval Charges ... 14
  - 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 ... 31
- Retrieving Vault Metadata ... 31
  - Retrieving Vault Metadata Using Java ... 31
  - Retrieving Vault Metadata Using .NET ... 33
  - Retrieving Vault Metadata Using REST ... 35
- Downloading a Vault Inventory ... 35
  - About the Inventory ... 36
  - Downloading a Vault Inventory Using Java ... 36
  - Downloading a Vault Inventory Using .NET ... 42
  - Downloading a Vault Inventory Using REST ... 47
- Configuring Vault Notifications ... 47
  - Configuring Vault Notifications Using Java ... 49
  - Configuring Vault Notifications Using .NET ... 51
  - Configuring Vault Notifications Using the REST API ... 54

Configuring Vault Notifications Using the Console ... 54
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 ... 59
Working with Archives ... 60
Archive Operations ... 60
Uploading an Archive ... 61
Downloading an Archive ... 61
Deleting an Archive ... 61
Updating an Archive ... 61
Maintaining Client-Side Archive Metadata ... 61
Uploading an Archive ... 62
Introduction ... 62
Uploading an Archive in a Single Operation ... 62
Uploading Large Archives in Parts ... 69
Downloading an Archive ... 77
About Range Retrievals ... 79
Downloading an Archive Using Java ... 79
Downloading an Archive Using .NET ... 92
Downloading an Archive Using REST ... 104
Deleting an Archive ... 104
Deleting an Archive Using Java ... 104
Deleting an Archive Using .NET ... 106
Deleting an Archive Using REST ... 108
Using the AWS SDKs ... 109
What is the Low-Level API? ... 109
What is the High-Level API? ... 110
When to Use High-Level and Low-Level API? ... 110
Using the AWS SDK for Java ... 110
Using the Low-Level API ... 111
Using the High-Level API ... 111
Running Java Examples Using Eclipse ... 112
Setting the Endpoint ... 112
Using the AWS SDK for .NET ... 113
Using the Low-Level API ... 113
Using the High-Level API ... 114
Running .NET Examples ... 114
Setting the Endpoint ... 114
Access Control ... 116
Example Policies ... 118
API Reference ... 120
Common Request Headers ... 120
Common Response Headers ... 123
Signing Requests ... 123
Example Signature Calculation ... 124
Calculating Signatures for the Streaming Operations ... 125
Computing Checksums ... 127
Tree Hash Example 1: Uploading an archive in a single request ... 128
Tree Hash Example 2: Uploading an archive using a multipart upload ... 128
Computing the Tree Hash of a File ... 129
Receiving Checksums When Downloading Data ... 137
Error Responses ... 139
Example 1: Describe Job request with a job ID that does not exist ... 140
Example 2: List Jobs request with an invalid value for the request parameter ... 141
Vault Operations ... 142
Create Vault ... 142

Delete Vault 144
Describe Vault 146
List Vaults 149
Set Vault Notification Configuration 154
Get Vault Notifications 156
Delete Vault Notifications 159
Archive Operations 161
Upload Archive 161
Delete Archive 164
Multipart Upload Operations 166
Initiate Multipart Upload 167
Upload Part 170
Complete Multipart Upload 174
Abort Multipart Upload 177
List Parts 179
List Multipart Uploads 184
Job Operations 189
Initiate Job 189
Describe Job 196
Get Job Output 203
List Jobs 210
Document History 219
AWS Glossary 220

What Is Amazon Glacier?

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)

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).

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.

  Note
  Amazon Glacier provides a management console, which you can use to create and delete vaults. However, all other interactions with Amazon Glacier require programming. For example,

to upload data, such as photos, videos, and other documents, you must write code and make requests using either the REST API directly or the AWS SDK for Java and .NET wrapper libraries.

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. 109)

  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. 60)

  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. 120)

  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.

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 East (Northern Virginia) Region. This vault can then be addressed by the following URI:

```
https://glacier.us-east-1.amazonaws.com/111122223333/vaults/examplevault
```

In the URI,

- `glacier.us-east-1.amazonaws.com` identifies the US East (Northern Virginia) 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. Archive IDs are unique within a vault. 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 East (Northern Virginia) Region:

```
https://glacier.us-east-1.amazonaws.com/111122223333/vaults/exampl
evault/archives/NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-TjhqG6eGoOY9Z8i1_AUyUsuh
PAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchiveId
```

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-east-1.amazonaws.com/111122223333/vaults/exampl
evault/jobs/HkF9p6o7yjhFx-K3CGl6fuSm6VzW9T7esGQfco8nUXVY
wS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

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-east-1:111122223333: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. 60).

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. 77) and Downloading a Vault Inventory in Amazon Glacier (p. 35).

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. 120).

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. 109).

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 supported AWS regions, go to the Regions and Endpoints section in the *Amazon Web Services Glossary*.

Getting Started 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)

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 Amazon Glacier 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. 109).

The last section of the getting started provides steps where you can deep dive and learn more about the developer experience with Amazon Glacier.

> **Note**
> 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 programming. For example, to upload data, such as photos, videos, and other documents, You need to write code and make requests using either the REST API directly or the AWS SDK for Java and .NET wrapper libraries. This getting started exercise provides code examples in Java and C# for you to upload and download an archive.

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.

Note
Amazon Glacier provides a management console, which you can use to create and delete vaults. However, all other interactions with Amazon Glacier require programming. For example, to upload data, such as photos, videos, and other documents, you must write code and make requests using either the REST API directly or the AWS SDK for Java and .NET wrapper libraries.

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. Go to http://aws.amazon.com, and then click **Sign Up**.
2. Follow the on-screen 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

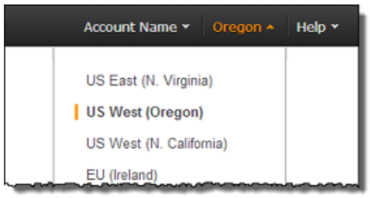
A vault is a container for storing archives. Your first step is to create a vault in one of the supported AWS regions. In this getting started exercise, you create a vault in the US West (Oregon) Region.

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.

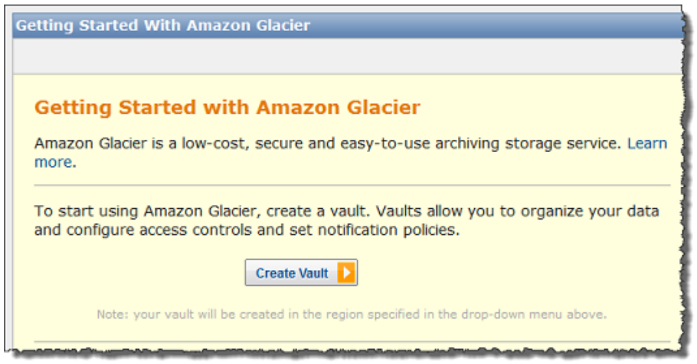
To create a vault

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. Click **Create Vault**.

   Getting Started With Amazon Glacier

   Getting Started with Amazon Glacier

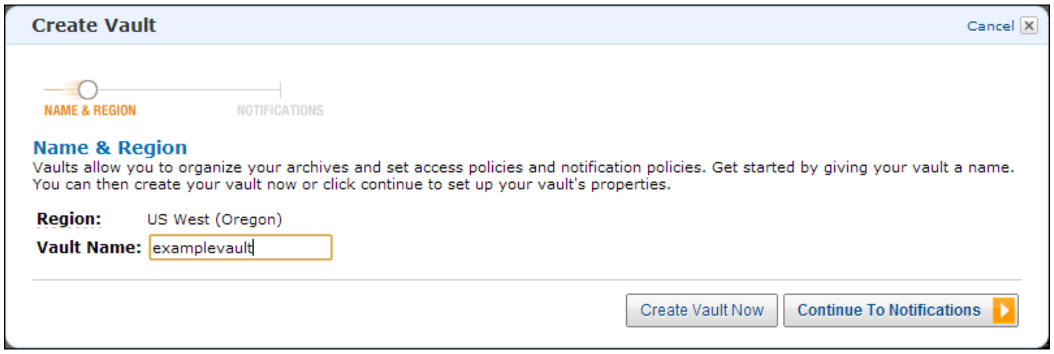
   Amazon Glacier is a low-cost, secure and easy-to-use archiving storage service. Learn more.

   To start using Amazon Glacier, create a vault. Vaults allow you to organize your data and configure access controls and set notification policies.

   Note: your vault will be created in the region specified in the drop-down menu above.

4. Enter **`examplevault`** as the vault name in the **Vault Name** field.

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

   Create Vault

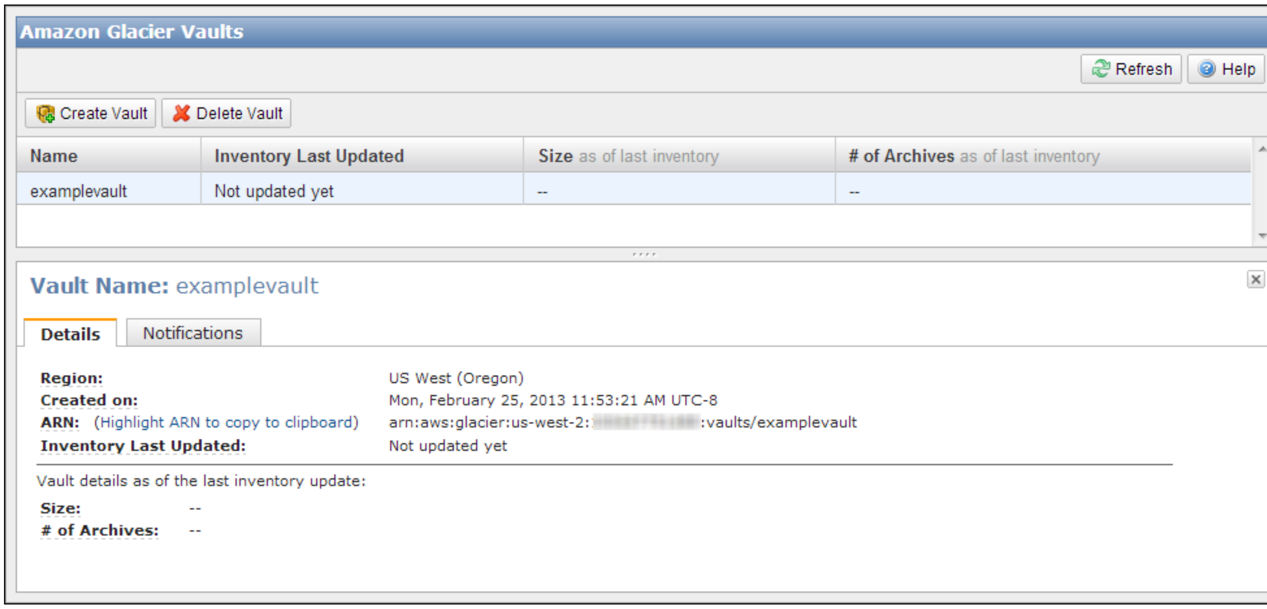
   Name & Region

   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 continue to set up your vault's properties.

   Region: US West (Oregon)
   Vault Name: examplevault

5. Click **Create Vault Now**.

If you wanted to have notifications sent to you or your application whenever certain Amazon Glacier jobs complete, you would click **Continue to Notifications** to set up Amazon Simple Notification Service (Amazon SNS) notifications. For this getting started exercise, you will not configure notifications for the vault. 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.



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

Vault Name: examplevault

Region:	US West (Oregon)
Created on:	Mon, February 25, 2013 11:53:21 AM UTC-8
ARN: (Highlight ARN to copy to clipboard)	arn:aws:glacier:us-west-2:[redacted]:vaults/examplevault
Inventory Last Updated:	Not updated yet

Vault details as of the last inventory update:

Size:	--
# of Archives:	--

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

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)

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 preceding links.

Note
Any archive operation, such as upload, download, and delete, requires programming. There is no console support for archive operations.

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. 69).

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. 112). 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 step-by-step instructions on how to run the following example, see Running Code Examples (p. 114). 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 to upload ***";


        public static void Main(string[] args)
        {
            try
            {
                var manager = new ArchiveTransferManager(Amazon.RegionEnd
point.USWest2);
                // Upload an archive.
                string archiveId = manager.Upload(vaultName, "getting started
archive test", archiveToUpload).ArchiveId;
                Console.WriteLine("Archive ID: (Copy and save this ID for the
next step) : {0}", 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();
        }
    }
}
```

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

Topics

- Amazon Glacier Data Retrieval Charges (p. 14)
- 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)

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).

Note
Any archive operation, such as upload, download, and delete, requires programming. There is no console support for archive operations.

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.

The retrieval job executes 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 data or inventory data). The code examples shown in this step initiate the job, wait for it to complete, and download the archive's data.

Caution
Note that it takes about four hours for most Amazon Glacier jobs to complete. Accordingly, after you initiate your archive retrieval job, it takes about four hours before you can download your archive data.

Amazon Glacier Data Retrieval Charges

Amazon Glacier is designed for long-term archival of data that will be stored for extended periods of time and accessed infrequently. You can retrieve up to 5 percent of your average monthly storage (prorated daily) for free each month. If you choose to retrieve more than this amount of data in a month, you are charged a retrieval fee starting at $0.01 per gigabyte.

For more information about Amazon Glacier data retrieval charges, see the Amazon Glacier detail page.

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. 112). 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).

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

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(glacierCli
ent, 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.

For step-by-step instructions on how to run this example, see Running Code Examples (p. 114). 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).

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

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 file path ***";

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

                var options = new DownloadOptions();
               options.StreamTransferProgress += ArchiveDownloadHighLevel_Get
tingStarted.progress;
                // Download an archive.
                manager.Download(vaultName, archiveId, downloadFilePath, op
tions);

                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).

Note
You cannot delete an archive using the Amazon Glacier console. Any archive operation, such as upload, download, or deletion, requires programming.

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. 104)

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. 112). 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. 114). 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);
        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 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.

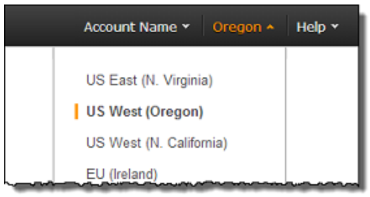
Note
Amazon Glacier prepares an inventory for each vault periodically, every 24 hours. Because the inventory might not reflect the latest information, Amazon Glacier ensures the vault is indeed empty by checking if there were any write operations since the last vault inventory.

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

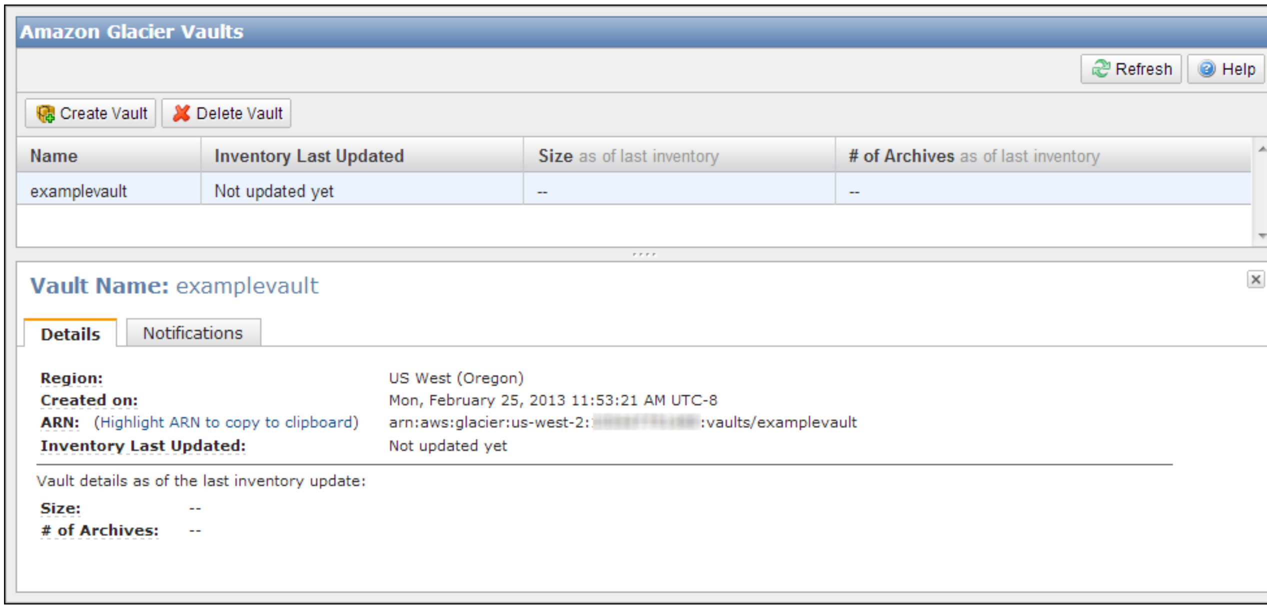
1. Sign into the AWS Management Console and open the Amazon Glacier console at https://console.aws.amazon.com/glacier.
2. From the region selector, select the AWS region where the vault exists that you want to delete.

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

Refresh Help

Create Vault Delete Vault

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

Vault Name: examplevault

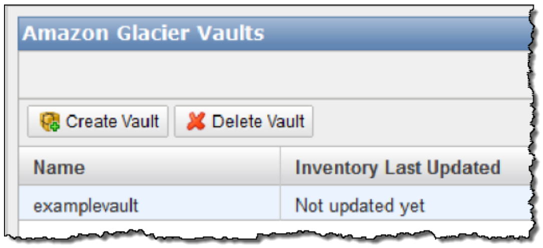
Details Notifications

Region:	US West (Oregon)
Created on:	Mon, February 25, 2013 11:53:21 AM UTC-8
ARN: (Highlight ARN to copy to clipboard)	arn:aws:glacier:us-west-2:[redacted]:vaults/examplevault
Inventory Last Updated:	Not updated yet

Vault details as of the last inventory update:

Size:	--
# of Archives:	--

4. Click **Delete Vault**.



Amazon Glacier Vaults

Create Vault Delete Vault

Name	Inventory Last Updated
examplevault	Not updated yet

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. 60)

Working with Vaults in Amazon Glacier

Topics

- Vault Operations in Amazon Glacier (p. 22)
- Creating a Vault in Amazon Glacier (p. 24)
- Retrieving Vault Metadata in Amazon Glacier (p. 31)
- Downloading a Vault Inventory in Amazon Glacier (p. 35)
- Configuring Vault Notifications in Amazon Glacier (p. 47)
- Deleting a Vault in Amazon Glacier (p. 56)

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.

> **Note**
> Amazon Glacier provides a management console. You can use the console to create and delete vaults. However, all other interactions with Amazon Glacier require programming. For example, to upload data, such as photos, videos, and other documents, You need to write code and make requests using either the REST API directly or the AWS SDK for Java and .NET wrapper libraries.

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 information on creating more vaults, go to the Amazon Glacier product detail page. For more information about supported regions, see Accessing Amazon Glacier (p. 5).

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. 31).

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. 35).

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

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. 31)

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 supported regions, see Accessing Amazon Glacier (p. 5). 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.

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. 110).
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-east-1` 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-east-1.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. 142).

Example: Creating a Vault Using the AWS SDK for Java

The following Java code example creates a vault in the `us-east-1` 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. 112).

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

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


public class AmazonGlacierVaultOperations {

    public static AmazonGlacierClient client;

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

     ProfileCredentialsProvider credentials = new ProfileCredentialsProvider();


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

        String vaultName = "examplevaultfordelete";

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

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

    private static void createVault(AmazonGlacierClient client, String vaultName)
```

```
 {
        CreateVaultRequest createVaultRequest = new CreateVaultRequest()
            .withVaultName(vaultName);
       CreateVaultResult createVaultResult = client.createVault(createVaultRe
quest);

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

    private static void describeVault(AmazonGlacierClient client, String
vaultName) {
        DescribeVaultRequest describeVaultRequest = new DescribeVaultRequest()

            .withVaultName(vaultName);
        DescribeVaultResult describeVaultResult  = client.describeVault(de
scribeVaultRequest);

        System.out.println("Describing the vault: " + vaultName);
        System.out.print(
                "CreationDate: " + describeVaultResult.getCreationDate() +
               "\nLastInventoryDate: " + describeVaultResult.getLastInventory
Date() +
                "\nNumberOfArchives: " + describeVaultResult.getNumberO
fArchives() +
                "\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

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)

Both the high-level and low-level APIs (p. 109) provided by the AWS SDK for .NET provide a method to create a vault.

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 East (Northern Virginia) 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. 114). 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.RegionEnd
point.USEast1);
          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");
      }
      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. 113).
3. Execute the `CreateVault` method by providing the request object as a parameter.

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

The following C# code snippet illustrates the preceding steps. The snippet creates a vault in the `US East (Northern Virginia) Region`. The `Location` it prints is the relative URI of the vault that includes your account ID, the region, and the vault name.

```
AmazonGlacier client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1);
CreateVaultRequest request = new CreateVaultRequest()
{
  VaultName = "*** Provide vault name ***"
};

CreateVaultResponse response = client.CreateVault(request);
Console.WriteLine("Vault created: {0}\n", response.CreateVaultResult.Location);
```

Note
For information about the underlying REST API, see Create Vault (PUT vault) (p. 142).

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

The following C# code example creates a vault in the US East (Northern Virginia) Region. 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 Code Examples (p. 114). 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 AmazonGlacier client;

    public static void Main(string[] args)
    {
       try
      {
       using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1))

        {
          Console.WriteLine("Creating a vault.");
          CreateAVault();
          DescribeVault();
          GetVaultsList();
          Console.WriteLine("\nVault created. Now press Enter to delete the
vault...");
          Console.ReadKey();
          DeleteVault();
        }
      }
      catch (AmazonGlacierException e) { Console.WriteLine(e.Message); }
      catch (AmazonServiceException e) { Console.WriteLine(e.Message); }
      catch (Exception e) { Console.WriteLine(e.Message); }
      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

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

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

      DescribeVaultResponse describeVaultResponse = client.DescribeVault(de
scribeVaultRequest);
      DescribeVaultResult describeVaultResult = describeVaultResponse.De
scribeVaultResult;
```

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

        ListVaultsResult result = response.ListVaultsResult;

        foreach (DescribeVaultOutput output in result.VaultList)
        {
          Console.WriteLine("Vault Name: {0} \tCreation Date: {1} \t #of
archives: {2}",
                            output.VaultName, output.CreationDate, output.Num
berOfArchives);
        }
        lastMarker = result.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. 142).

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

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. 33)
- Retrieving Vault Metadata Using the REST API (p. 35)

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.

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. 32)
- Example: Retrieving Vault Metadata Using the AWS SDK for Java (p. 33)

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. 110).
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. 146).

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-east-1` 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-east-1.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. 149)).

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. 33)
- Retrieve Vault Metadata for All Vaults in a Region (p. 34)
- Example: Retrieving Vault Metadata Using the Low-Level API of the AWS SDK for .NET (p. 95)

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. 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 C# code snippet illustrates the preceding steps. The snippet retrieves metadata information of an existing vault in the US East (Northern Virginia) Region.

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

DescribeVaultRequest describeVaultRequest = new DescribeVaultRequest()
{
  VaultName = "*** Provide vault name ***"
};
DescribeVaultResponse describeVaultResponse = client.DescribeVault(de
scribeVaultRequest);
DescribeVaultResult describeVaultResult = describeVaultResponse.DescribeVaultRes
ult;
Console.WriteLine("\nVault description...");
Console.WriteLine(
   "\nVaultName: " + describeVaultResult.VaultName +
   "\nVaultARN: " + describeVaultResult.VaultARN +
   "\nVaultCreationDate: " + describeVaultResult.CreationDate +
   "\nNumberOfArchives: " + describeVaultResult.NumberOfArchives +
   "\nSizeInBytes: " + describeVaultResult.SizeInBytes +
   "\nLastInventoryDate: " + describeVaultResult.LastInventoryDate
   );
```

Note
For information about the underlying REST API, see Describe Vault (GET vault) (p. 146).

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 East (Northern Virginia) 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.

```
AmazonGlacier client;
client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1);
string lastMarker = null;
do
{
  ListVaultsRequest request = new ListVaultsRequest()
  {
    Limit = 5,
    Marker = lastMarker
  };
  ListVaultsResponse response = client.ListVaults(request);

  ListVaultsResult result = response.ListVaultsResult;
  Console.WriteLine("\n List of vaults in your account in the specific region
...");
  foreach (DescribeVaultOutput output in result.VaultList)
  {
    Console.WriteLine("Vault Name: {0} \tCreation Date: {1} \t #of archives:
{2}",
                      output.VaultName, output.CreationDate, output.NumberO
fArchives);
  }
  lastMarker = result.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. 149)).

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. 149). To describe one vault, see Describe Vault (GET vault) (p. 146).

Downloading a Vault Inventory in Amazon Glacier

Topics

- About the Inventory (p. 36)
- 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. 42)
- Downloading a Vault Inventory Using the REST API (p. 47)

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.
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. 196)) 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.

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. 190).

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. 206) 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. 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. 189).

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.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 success
fully."); }

            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(qRe
quest);
        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).withMaxNumberOfMes
sages(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").text
Value();

                    JsonParser jpDesc = factory.createJsonParser(jobMessage);
                    JsonNode jobDescNode = mapper.readTree(jpDesc);
                 String retrievedJobId = jobDescNode.get("JobId").textValue();

                    String statusCode = jobDescNode.get("StatusCode").text
Value();
                    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(getJobOut
putRequest);

        FileWriter fstream = new FileWriter(fileName);
        BufferedWriter out = new BufferedWriter(fstream);
       BufferedReader in = new BufferedReader(new InputStreamReader(getJobOut
putResult.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.

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

InitiateJobRequest initJobRequest = new InitiateJobRequest()
{
  VaultName = vaultName,
  JobParameters = new JobParameters()
  {
    Type = "inventory-retrieval",
    SNSTopic = "*** Provide Amazon SNS topic ***",

  }
};
InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);
string jobId = initJobResponse.InitiateJobResult.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. The output that Amazon Glacier returns is available in the `GetJobOutputResponse` object.

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

GetJobOutputResponse getJobOutputResponse = client.GetJobOutput(getJobOutpu
tRequest);
GetJobOutputResult result = getJobOutputResponse.GetJobOutputResult;

// GetJobOutputResult.Body provides the output stream.
using (Stream webStream = result.Body)
{
  using (Stream fileToSave = File.OpenWrite(fileName))
  {
    CopyStream(webStream, fileToSave);
  }
}
```

Note
For information about the job related underlying REST API, see Job Operations (p. 189).

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.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 = "examplevault";
    static string fileName  = "*** Provide file name to store inventory ***";
    static AmazonSimpleNotificationService snsClient;
    static AmazonSQS 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)
    {
      AmazonGlacier client;
      try
      {
       using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1))

        {
          // Setup SNS topic and SQS queue.
          SetupTopicAndQueue();
          GetVaultInventory(client);
        }
        Console.WriteLine("Operations successful. To continue, press Enter");
      }
      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
});
      }

      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static void SetupTopicAndQueue()
    {
      snsClient = new AmazonSimpleNotificationServiceClient(Amazon.RegionEnd
point.USEast1);
      sqsClient = new AmazonSQSClient(Amazon.RegionEndpoint.USEast1);

      long ticks = DateTime.Now.Ticks;
      topicArn = snsClient.CreateTopic(new CreateTopicRequest { Name = "Glaci
erDownload-" + ticks }).CreateTopicResult.TopicArn;
      queueUrl = sqsClient.CreateQueue(new CreateQueueRequest() { QueueName =
"GlacierDownload-" + ticks }).CreateQueueResult.QueueUrl;
      queueArn = sqsClient.GetQueueAttributes(new GetQueueAttributesRequest()
{ QueueUrl = queueUrl, AttributeName = new List<string> { "QueueArn" }
}).GetQueueAttributesResult.QueueARN;

      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("{QuernArn}", queueArn).Replace("{Topi
cArn}", topicArn);
      sqsClient.SetQueueAttributes(new SetQueueAttributesRequest()
      {
        QueueUrl = queueUrl,
        Attribute = new List<Amazon.SQS.Model.Attribute>()
        {
           new Amazon.SQS.Model.Attribute()
           {
             Name = "Policy",
             Value = policy
           }
        }
      });
    }

    static void GetVaultInventory(AmazonGlacier client)
    {
      // Initiate job.
      InitiateJobRequest initJobRequest = new InitiateJobRequest()
      {
        VaultName = vaultName,
        JobParameters = new JobParameters()
        {
          Type = "inventory-retrieval",
          Description = "This job is to download archive updated as part of
getting started",
          SNSTopic = topicArn,
        }
      };
     InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);

      string jobId = initJobResponse.InitiateJobResult.JobId;

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

    private static void ProcessQueue(string jobId, AmazonGlacier client)
    {
      var receiveMessageRequest = new ReceiveMessageRequest() { QueueUrl =
queueUrl, MaxNumberOfMessages = 1 };
      bool jobDone = false;
      while (!jobDone)
      {
       var receiveMessageResponse = sqsClient.ReceiveMessage(receiveMessageRe
quest);
        if (receiveMessageResponse.ReceiveMessageResult.Message.Count == 0)
        {
          Thread.Sleep(1000 * 60);
          continue;
        }
       Message message = receiveMessageResponse.ReceiveMessageResult.Message[0];

       Dictionary<string, string> outerLayer = JsonConvert.DeserializeObject<Dic
tionary<string, string>>(message.Body);
        Dictionary<string, string> fields = JsonConvert.DeserializeObject<Dic
tionary<string, string>>(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); // 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, AmazonGlacier client)
    {
      GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
      {

        JobId = jobId,
        VaultName = vaultName
      };
     GetJobOutputResponse getJobOutputResponse = client.GetJobOutput(getJobOut
putRequest);
      GetJobOutputResult result = getJobOutputResponse.GetJobOutputResult;

      using (Stream webStream = result.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 a Job (POST jobs) (p. 189).
2. After the job completes, download the inventory data. For more information, see Get Job Output (GET output) (p. 203).

Configuring Vault Notifications in Amazon Glacier

Topics

- Configuring Vault Notifications in Amazon Glacier Using the AWS SDK for Java (p. 49)
- 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. 54)
- Configuring Vault Notifications Using the Amazon Glacier Console (p. 54)

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.

The 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-east-1:012345678901: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-K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZSh
joQzQVVh7vEXAMPLEjobID",
 "SHA256TreeHash":null,
 "SNSTopic": "arn:aws:sns:us-east-1:012345678901:mytopic",
 "StatusCode":"Succeeded",
 "StatusMessage": "Succeeded",
 "VaultARN": "arn:aws:glacier:us-east-1:012345678901: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. 154)).
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", "InventoryRetrievalCom
pleted")
         );
client.setVaultNotifications(request);
```

Note
For information about the underlying REST API, see Vault Operations (p. 142).

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. 110).

```
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", "InventoryRetrievalCom
pleted");

       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(re
quest);
        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 DeleteVaultNotifica
tionsRequest()
                .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. 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. 154)).
3. Execute the `SetVaultNotifications` method by providing the request object as a parameter.

The following C# 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.

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

SetVaultNotificationsRequest request = new SetVaultNotificationsRequest()
 {
   VaultName = "*** Provide vault name. ***",
   VaultNotificationConfig = new VaultNotificationConfig()
   {
     Events = new List<string>() { "ArchiveRetrievalCompleted", "InventoryRe
trievalCompleted" },
     SNSTopic = "*** Provide an existing SNS topic ***"
   }
 };

 SetVaultNotificationsResponse response = client.SetVaultNotifications(request);
```

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.

Note
For information about the underlying REST API, see Vault Operations (p. 142).

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

The following C# code examples sets the notification configuration on the vault ("`examplevault`") in the US East (Northern Virginia) Region, retrieves the configuration, and then deletes it. The configuration identifies the `ArchiveRetrievalCompleted` and the `InventoryRetrievalCompleted` events. Whenever you create a job to download an archive or request vault inventory, Amazon Glacier sends a message to the specified Amazon SNS topic when it completes the job.

For step-by-step instructions to run the following example, see Running Code Examples (p. 114). 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 AmazonGlacier client;

    public static void Main(string[] args)
    {
      try
      {
       using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1))

        {
         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", "Invent
oryRetrievalCompleted" },
          SNSTopic = snsTopicARN
        }
      };
     SetVaultNotificationsResponse response = client.SetVaultNotifications(re
quest);
    }

    static void GetVaultNotificationConfig()
    {
     GetVaultNotificationsRequest request = new GetVaultNotificationsRequest()

      {
        VaultName = vaultName,
        AccountId = "-"
      };
     GetVaultNotificationsResponse response = client.GetVaultNotifications(re
quest);
     Console.WriteLine("SNS Topic ARN: {0}", response.GetVaultNotificationsRes
ult.VaultNotificationConfig.SNSTopic);
      foreach (string s in response.GetVaultNotificationsResult.VaultNotifica
tionConfig.Events)
        Console.WriteLine("Event : {0}", s);
    }

    static void DeleteVaultNotificationConfig()
    {

      DeleteVaultNotificationsRequest request = new DeleteVaultNotification
sRequest()
      {

        VaultName = vaultName
      };
      DeleteVaultNotificationsResponse response = client.DeleteVaultNotifica
tions(request);
    }
  }
}
```

Configuring Vault Notifications in Amazon Glacier Using the REST API

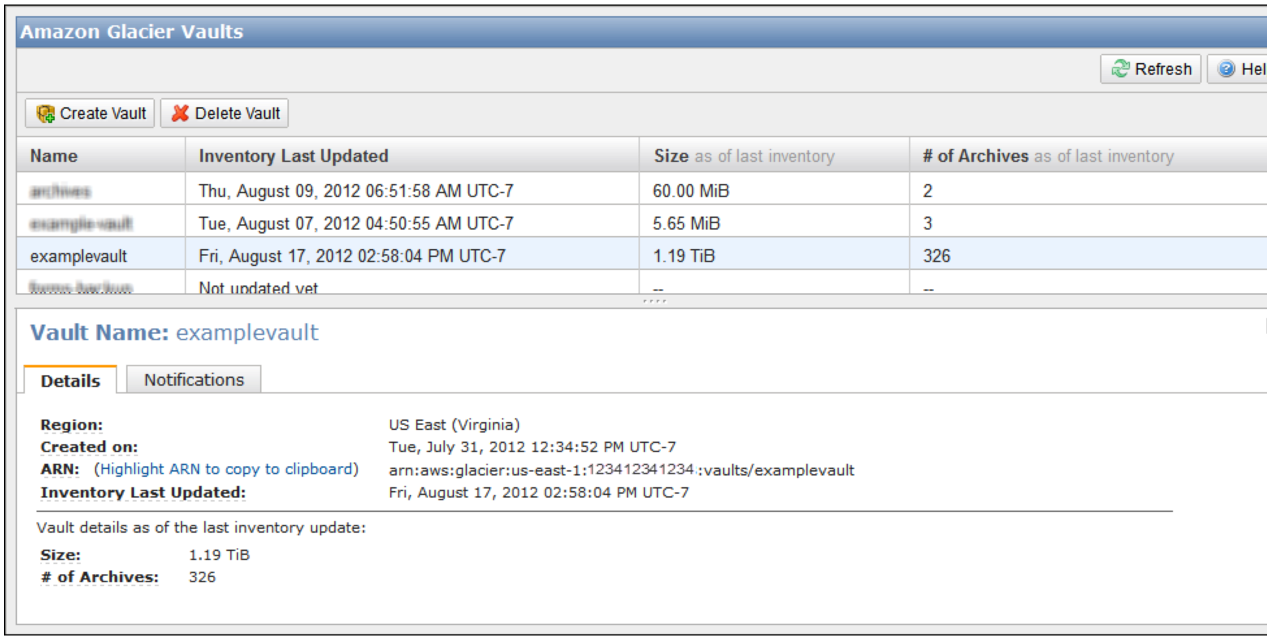
To configure vault notifications using the REST API, see Set Vault Notification Configuration (PUT notification-configuration) (p. 154). Additionally, you can also get vault notifications (Get Vault Notifications (GET notification-configuration) (p. 156)) and delete vault notifications (Delete Vault Notifications (DELETE notification-configuration) (p. 159)).

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.



Name	Inventory Last Updated	Size as of last inventory	# of Archives as of last inventory
[redacted]	Thu, August 09, 2012 06:51:58 AM UTC-7	60.00 MiB	2
[redacted]	Tue, August 07, 2012 04:50:55 AM UTC-7	5.65 MiB	3
examplevault	Fri, August 17, 2012 02:58:04 PM UTC-7	1.19 TiB	326
[redacted]	Not updated yet	--	--

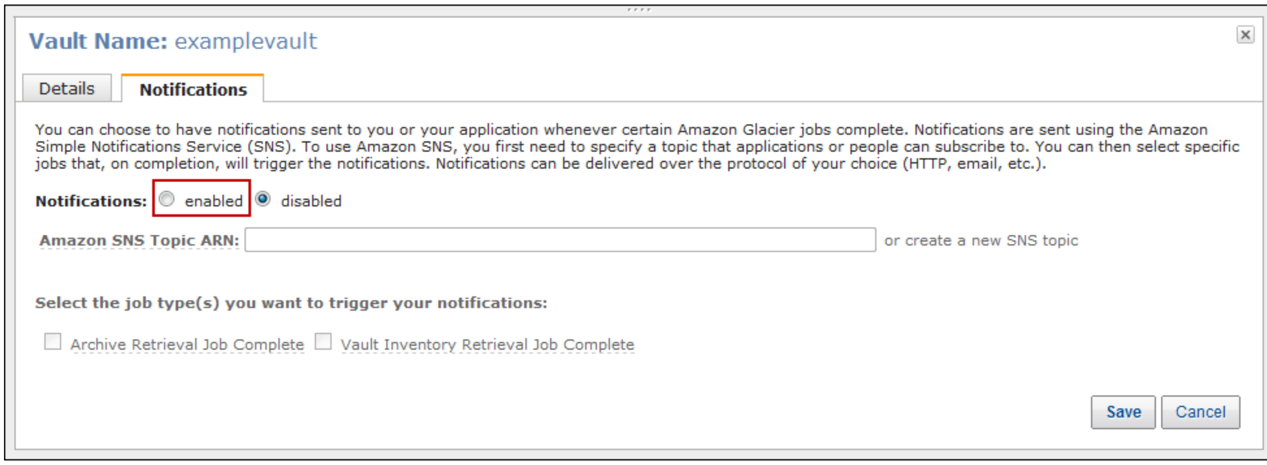
Vault Name: examplevault

Region:	US East (Virginia)
Created on:	Tue, July 31, 2012 12:34:52 PM UTC-7
ARN: (Highlight ARN to copy to clipboard)	arn:aws:glacier:us-east-1:123412341234:vaults/examplevault
Inventory Last Updated:	Fri, August 17, 2012 02:58:04 PM UTC-7

Vault details as of the last inventory update:

Size:	1.19 TiB
# of Archives:	326

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



Vault Name: examplevault

Details | 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.).

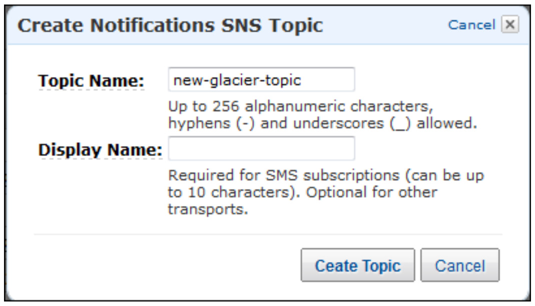
Notifications: enabled [✓ disabled]

Amazon SNS Topic ARN: or create a new SNS topic

Select the job type(s) you want to trigger your notifications:

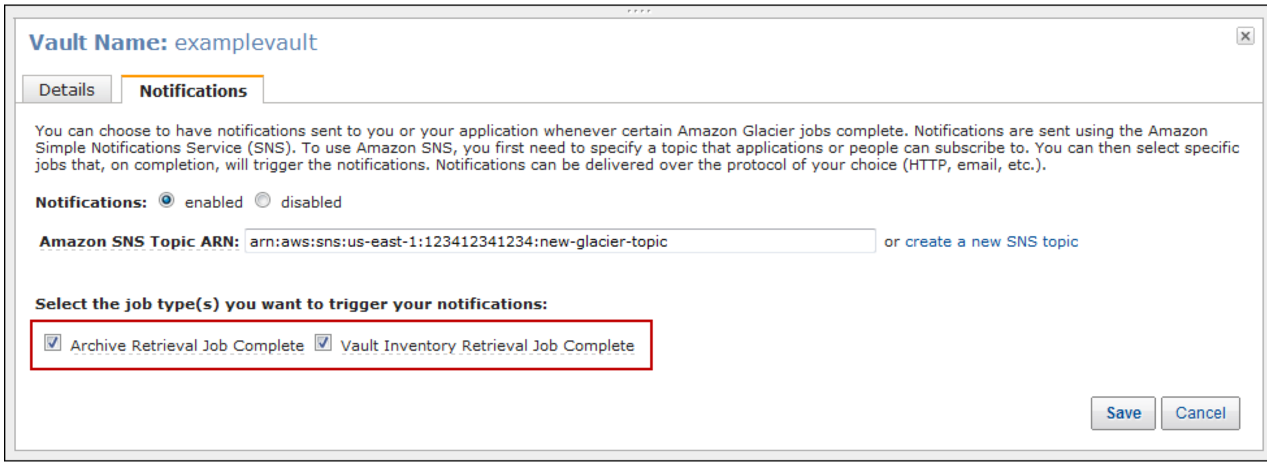
☐ Archive Retrieval Job Complete ☐ Vault Inventory Retrieval Job Complete

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

To...	Do this...
Specify an existing Amazon SNS topic	Enter the Amazon SNS topic in the **Amazon SNS Topic ARN** text box. The topic is an Amazon Resource Name (ARN) that has the form shown below. `arn:aws:sns:`*`region`*`:`*`accountId`*`:`*`topicname`* You can find the an Amazon SNS topic ARN from the Amazon Simple Notification Service (Amazon SNS) console.
Create a new Amazon SNS topic	a. Click **create a new SNS topic**. A **Create Notifications SNS Topic** dialog box appears. b. In the **Topic Name** field, specify the name of the new topic. If you will subscribe to the topic using SMS subscriptions, put a name in the **Display Name** field.  **Create Notifications SNS Topic** **Topic Name:** new-glacier-topic Up to 256 alphanumeric characters, hyphens (-) and underscores (_) allowed. **Display Name:** Required for SMS subscriptions (can be up to 10 characters). Optional for other transports. c. Click **Create Topic**. The **Amazon SNS Topic ARN** text box is populated with the ARN of the new topic.

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**.



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.).

Notifications: [✓ enabled] disabled

Amazon SNS Topic ARN: arn:aws:sns:us-east-1:123412341234:new-glacier-topic or create a new SNS topic

Select the job type(s) you want to trigger your notifications:

✓ Archive Retrieval Job Complete ✓ Vault Inventory Retrieval 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

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. 59)

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.

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.

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. 110).
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. 144).

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. 144).

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

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. 58)

Both the high-level and low-level APIs (p. 109) provided by the AWS SDK for .NET provide a method to delete a vault.

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. 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. 144).

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. 144).

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. 144).

Deleting a Vault Using the Amazon Glacier Console

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**.

Working with Archives in Amazon Glacier

Topics

- Archive Operations in Amazon Glacier (p. 60)
- Maintaining Client-Side Archive Metadata (p. 61)
- Uploading an Archive in Amazon Glacier (p. 62)
- Downloading an Archive in Amazon Glacier (p. 77)
- Deleting an Archive in Amazon Glacier (p. 104)

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-
3p6LCJClYytFT_CBhT9CwNxbRaM5MetS3I-GqwxI3Y8QtgbJbhEQPs0mJ3KExample
```

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

Note
Amazon Glacier provides a management console. You can use the console to create and delete vaults. However, all other interactions with Amazon Glacier require programming. For example, to upload data, such as photos, videos, and other documents, You need to write code and make requests using either the REST API directly or the AWS SDK for Java and .NET wrapper libraries.

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. 62).

> **Note**
> Amazon Glacier provides a management console. You can use the console to create and delete vaults. However, all other interactions with Amazon Glacier require programming. For example, to upload data, such as photos, videos, and other documents, You need to write code and make requests using either the REST API directly or the AWS SDK for Java and .NET wrapper libraries.

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. 77).

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. 104).

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

Topics

- Introduction (p. 62)
- Uploading an Archive in a Single Operation (p. 62)
- Uploading Large Archives in Parts (Multipart Upload) (p. 69)

Introduction

Amazon Glacier provides a management console that you can use to create and delete vaults. However, all other interactions with Amazon Glacier require programming. To upload data, you'll need to write code and make requests using either the REST API directly or the AWS SDK for Java and .NET wrapper libraries.

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. 62).
- **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. 69).

Note
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

Topics

- Uploading an Archive in a Single Operation Using the AWS SDK for Java (p. 63)
- Uploading an Archive in a Single Operation Using the AWS SDK for .NET in Amazon Glacier (p. 66)
- Uploading an Archive in a Single Operation Using the REST API (p. 69)

As described in Uploading an Archive in Amazon Glacier (p. 62), 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.

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

Topics

- Uploading an Archive Using the High-Level API of the AWS SDK for Java (p. 63)
- Uploading an Archive in a Single Operation Using the Low-Level API of the AWS SDK for Java (p. 64)

Both the high-level and low-level APIs (p. 109) provided by the AWS SDK for Java provide a method to upload an archive.

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. 112). 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. 110).
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.get
Location());
```

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 (`exampl-evault`). For step-by-step instructions on how to run this example, see Running Java Examples for Amazon Glacier Using Eclipse (p. 112). 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(re
quest);

            System.out.println("ArchiveID: " + uploadArchiveResult.getArchi
veId());

        } 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

Topics

- Uploading an Archive Using the High-Level API of the AWS SDK for .NET (p. 66)
- Uploading an Archive in a Single Operation Using the Low-Level API of the AWS SDK for .NET (p. 67)

Both the high-level and low-level APIs (p. 109) provided by the AWS SDK for .NET provide a method to upload an archive in a single operation.

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 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 East (Northern Virginia) Region.

For step-by-step instructions on how to run this example, see Running Code Examples (p. 114). 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 to upload ***";

    public static void Main(string[] args)
    {
       try
      {
          var manager = new ArchiveTransferManager(Amazon.RegionEnd
point.USEast1);
          // Upload an archive.
          string archiveId = manager.Upload(vaultName, "Tax 2012 document",
archiveToUpload).ArchiveId;
          Console.WriteLine("Archive ID: (Copy and save this ID for the next
step) : {0}", 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();
    }
  }
}
```

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. 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 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);

UploadArchiveRequest request = new UploadArchiveRequest()
{
  VaultName = "*** provide vault name ***",
  Body      = fileStream,
  Checksum  = treeHash

};
UploadArchiveResponse response = client.UploadArchive(request);
UploadArchiveResult result = response.UploadArchiveResult;
Console.WriteLine("Archive ID: {0}", result.ArchiveId);
```

Note
For information about the underlying REST API to upload an archive in a single request, see Upload Archive (POST archive) (p. 161).

Example: Uploading an Archive in a Single Operation Using the Low-Level API of 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. 114). 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 to upload ***";

    public static void Main(string[] args)
    {
      AmazonGlacier client;
      try
      {
       using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1))

        {
          Console.WriteLine("Uploading an archive.");
          string archiveId = UploadAnArchive(client);
          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();
    }

    static string UploadAnArchive(AmazonGlacier 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);
        UploadArchiveResult result = response.UploadArchiveResult;
        string archiveID = result.ArchiveId;
        return archiveID;
```

```
            }
         }
      }
   }
```

Uploading an Archive in a Single Operation Using the REST API

Topics

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. 161).

Uploading Large Archives in Parts (Multipart Upload)

Topics

- Multipart Upload Process (p. 69)
- Quick Facts (p. 70)
- Uploading Large Archives in Parts Using the AWS SDK for Java (p. 71)
- Uploading Large Archives Using the AWS SDK for .NET (p. 74)
- Uploading Large Archives in Parts Using the REST API (p. 77)

Multipart Upload Process

As described in Uploading an Archive in Amazon Glacier (p. 62), 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

Topics

- Uploading Large Archives in Parts Using the High-Level API of the AWS SDK for Java (p. 71)
- Upload Large Archives in Parts Using the Low-Level API of the AWS SDK for Java (p. 71)

Both the high-level and low-level APIs (p. 109) provided by the AWS SDK for Java provide a method to upload a large archive (see Uploading an Archive in Amazon Glacier (p. 62)).

- 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. 110).

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. 63).

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 (`exampl-evault`). For step-by-step instructions on how to run this example, see Running Java Examples for Amazon Glacier Using Eclipse (p. 112). 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 InitiateMultipartUploadRe
quest()
            .withVaultName(vaultName)
            .withArchiveDescription("my archive " + (new Date()))
            .withPartSize(partSize);

       InitiateMultipartUploadResult result = client.initiateMultipartUpload(re
quest);

        System.out.println("ArchiveID: " + result.getUploadId());
        return result.getUploadId();
    }

    private static String uploadParts(String uploadId) throws AmazonServiceEx
ception, 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 ByteArray
InputStream(bytesRead));
            byte[] binaryChecksum = BinaryUtils.fromHex(checksum);
            binaryChecksums.add(binaryChecksum);
            System.out.println(contentRange);

            //Upload part.
           UploadMultipartPartRequest partRequest = new UploadMultipartPartRe
quest()
            .withVaultName(vaultName)
            .withBody(new ByteArrayInputStream(bytesRead))
            .withChecksum(checksum)
            .withRange(contentRange)
            .withUploadId(uploadId);

            UploadMultipartPartResult partResult = client.uploadMultipart
```

```
Part(partRequest);
          System.out.println("Part uploaded, checksum: " + partResult.getCheck
sum());

            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 CompleteMultipartUp
loadRequest()
            .withVaultName(vaultName)
            .withUploadId(uploadId)
            .withChecksum(checksum)
            .withArchiveSize(String.valueOf(file.length()));

        CompleteMultipartUploadResult compResult = client.completeMultipartUp
load(compRequest);
        return compResult.getLocation();
    }
}
```

Uploading Large Archives Using the AWS SDK for .NET

Topics

- Uploading Large Archives in Parts Using the High-Level API of the AWS SDK for .NET (p. 74)
- Uploading Large Archives in Parts Using the Low-Level API of the AWS SDK for .NET (p. 75)

Both the high-level and low-level APIs (p. 109) provided by the AWS SDK for .NET provide a method to upload large archives in parts (see Uploading an Archive in Amazon Glacier (p. 62)).

- 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. 113).

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. 66).

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. 114). 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 to upload ***";
    static long partSize          = 4194304; // 4 MB.

    public static void Main(string[] args)
    {
      AmazonGlacier client;
      List<string> partChecksumList = new List<string>();
      try
      {
       using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1))

        {
          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(AmazonGlacier client)
    {
     InitiateMultipartUploadRequest initiateMPUrequest = new InitiateMultipar
tUploadRequest()
      {

        VaultName = vaultName,
        PartSize = partSize,
        ArchiveDescription = "Test doc uploaded using MPU"
      };

     InitiateMultipartUploadResponse initiateMPUresponse = client.InitiateMul
tipartUpload(initiateMPUrequest);

      return initiateMPUresponse.InitiateMultipartUploadResult.UploadId;
    }

    static List<string> UploadParts(string uploadID, AmazonGlacier 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(upload
PartStream);
          partChecksumList.Add(checksum);
          // Upload part.
          UploadMultipartPartRequest uploadMPUrequest = new UploadMultipart
PartRequest()
          {
            VaultName = vaultName,
            Body = uploadPartStream,
            Checksum = checksum,
            UploadId = uploadID
          };
          uploadMPUrequest.SetRange(currentPosition, currentPosition + upload
PartStream.Length - 1);
          client.UploadMultipartPart(uploadMPUrequest);

          currentPosition = currentPosition + uploadPartStream.Length;
        }
      }
      return partChecksumList;
    }

   static string CompleteMPU(string uploadID, AmazonGlacier client, List<string>
 partChecksumList)
    {
      long fileLength = new FileInfo(archiveToUpload).Length;
     CompleteMultipartUploadRequest completeMPUrequest = new CompleteMultipar
tUploadRequest()
      {
        UploadId = uploadID,
        ArchiveSize = fileLength.ToString(),
        Checksum = TreeHashGenerator.CalculateTreeHash(partChecksumList),
        VaultName = vaultName
      };

     CompleteMultipartUploadResponse completeMPUresponse = client.CompleteMul
tipartUpload(completeMPUrequest);
      return completeMPUresponse.CompleteMultipartUploadResult.ArchiveId;
    }
  }
}
```

Uploading Large Archives in Parts Using the REST API

As described in Uploading Large Archives in Parts (Multipart Upload) (p. 69), 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. 167)
- Upload Part (PUT uploadID) (p. 170)
- Complete Multipart Upload (POST uploadID) (p. 174)
- Abort Multipart Upload (DELETE uploadID) (p. 177)
- List Parts (GET uploadID) (p. 179)
- List Multipart Uploads (GET multipart-uploads) (p. 184)

Downloading an Archive in Amazon Glacier

Topics

- About Range Retrievals (p. 79)

- Downloading an Archive in Amazon Glacier Using the AWS SDK for Java (p. 79)
- Downloading an Archive in Amazon Glacier Using the AWS SDK for .NET (p. 92)
- Downloading an Archive Using the REST API (p. 104)

Retrieving an archive from Amazon Glacier is a two-step process:

1. Initiate an archive retrieval job.

   When you initiate a job, Amazon Glacier returns a job ID in the response and executes the job asynchronously. After the job completes, you can download the job output.

   You can initiate a job requesting Amazon Glacier to prepare an entire archive or a portion of the archive for subsequent download. When an archive is very large, you may find it cost effective to initiate several sequential jobs to prepare your archive. For example, to retrieve a 1 GB archive, you may 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 About Range Retrievals (p. 79).

   To initiate an archive retrieval job you must know 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. 35).
2. After the job completes, download the bytes.

   You can download all the bytes or specify a byte range to download only a portion of the 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.

Most Amazon Glacier jobs take about four hours to complete. Amazon Glacier must complete a job before you can get its output. 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 these 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. Amazon Glacier sends notification only after it completes the job. You can specify an SNS topic per job; that is, you can specify an SNS topic when you initiate a job. In addition to specifying an SNS topic in your job request, if your vault has notifications configuration set for archive retrieval events, then Amazon Glacier publishes a notification to that SNS topic as well. For more information, see Configuring Vault Notifications in Amazon Glacier (p. 47).
- Request job information explicitly—Amazon Glacier provides a describe job operation (Describe Job (GET JobID) (p. 196)) 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.

About Range 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:

- **Control bandwidth costs** – Each month, Amazon Glacier allows you to retrieve up to 5 percent of your average monthly storage (pro-rated daily) for free. Scheduling range retrievals can help you in two ways. First, it may help you meet the monthly free allowance of 5 percent by spreading out the data you request. Second, if the amount of data you want to retrieve is such that you don't meet the free allowance percentage, scheduling range retrievals enables you to reduce your peak retrieval rate, which determines your retrieval fees. For examples of retrieval fee calculations, go to Amazon Glacier FAQs.
- **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. This means that the byte range can start at zero (which would be the beginning of your archive), or at any 1 MB interval thereafter (1 MB, 2 MB, 3 MB, etc.). 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. 137).

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

Topics

- Downloading an Archive Using the High-Level API of the AWS SDK for Java (p. 79)
- Downloading an Archive Using the Low-Level API of the AWS SDK for Java (p. 80)

Both the high-level and low-level APIs (p. 109) provided by the AWS SDK for Java provide a method to download an archive.

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.

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. 112). 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(glacierCli
ent, 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 InitiateJobRe
quest()
    .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 InitiateJobRe
quest()
    .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. 137).

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. 87).

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. 87).

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 success
fully."); }

            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(qRe
quest);
        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).withMaxNumberOfMes
sages(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").getText
Value();

                    JsonParser jpDesc = factory.createJsonParser(jobMessage);
                    JsonNode jobDescNode = mapper.readTree(jpDesc);
                    String retrievedJobId = jobDescNode.get("JobId").getText
Value();
                    String statusCode = jobDescNode.get("StatusCode").getText
Value();
                    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(getJobOut
putRequest);

        InputStream input = new BufferedInputStream(getJobOutputResult.get
Body());
        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(qRe
quest);
        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).withMaxNumberOfMes
sages(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").text
Value();

                    JsonParser jpDesc = factory.createJsonParser(jobMessage);
                    JsonNode jobDescNode = mapper.readTree(jpDesc);
                 String retrievedJobId = jobDescNode.get("JobId").textValue();

                    String statusCode = jobDescNode.get("StatusCode").text
Value();
                    archiveSizeInBytes = jobDescNode.get("ArchiveSizeIn
Bytes").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(get
JobOutputRequest);

            BufferedInputStream is = new BufferedInputStream(getJobOutputRes
ult.getBody());
            byte[] buffer = new byte[(int)(endRange - startRange + 1)];

            System.out.println("Checksum received: " + getJobOutputRes
ult.getChecksum());
           System.out.println("Content range " + getJobOutputResult.getConten
tRange());


            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.cal
culateTreeHash(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

Topics

- Downloading an Archive Using the High-Level API of the AWS SDK for .NET (p. 92)
- Downloading an Archive Using the Low-Level API of the AWS SDK for .NET (p. 93)

Both the high-level and low-level APIs (p. 109) provided by the AWS SDK for .NET provide a method to download an archive.

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.

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 East (Northern Virginia) Region.

For step-by-step instructions on how to run this example, see Running Code Examples (p. 114). 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 file path ***";

    public static void Main(string[] args)
    {
      try
      {
       var manager = new ArchiveTransferManager(Amazon.RegionEndpoint.USEast1);


        var options = new DownloadOptions();
        options.StreamTransferProgress += ArchiveDownloadHighLevel.progress;
        // Download an archive.
        manager.Download(vaultName, archiveId, downloadFilePath, options);

        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 to retrieve a vault inventory using the low-level AP 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.

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

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.InitiateJobResult.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.

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

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.InitiateJobResult.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(getJobOutpu
tRequest);
GetJobOutputResult result = getJobOutputResponse.GetJobOutputResult;

// GetJobOutputResult.Body provides the output stream.
using (Stream webStream = result.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. 137).

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. 99).

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. 99).

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 file name to store inventory ***";
    static AmazonSimpleNotificationService snsClient;
    static AmazonSQS 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)
    {
      AmazonGlacier client;
      try
      {
       using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1))

        {
          // Setup SNS topic and SQS queue.
          SetupTopicAndQueue();
          RetrieveArchive(client);
        }
        Console.WriteLine("Operations successful. To continue, press Enter");
      }
      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
});
      }

      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static void SetupTopicAndQueue()
    {
      snsClient = new AmazonSimpleNotificationServiceClient(Amazon.RegionEnd
point.USEast1);
      sqsClient = new AmazonSQSClient(Amazon.RegionEndpoint.USEast1);

      long ticks = DateTime.Now.Ticks;
      topicArn = snsClient.CreateTopic(new CreateTopicRequest { Name = "Glaci
erDownload-" + ticks }).CreateTopicResult.TopicArn;
      queueUrl = sqsClient.CreateQueue(new CreateQueueRequest() { QueueName =
"GlacierDownload-" + ticks }).CreateQueueResult.QueueUrl;
      queueArn = sqsClient.GetQueueAttributes(new GetQueueAttributesRequest()
{ QueueUrl = queueUrl, AttributeName = new List<string> { "QueueArn" }
}).GetQueueAttributesResult.QueueARN;

      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("{QuernArn}", queueArn).Replace("{Topi
cArn}", topicArn);
      sqsClient.SetQueueAttributes(new SetQueueAttributesRequest()
      {
        QueueUrl = queueUrl,
        Attribute = new List<Amazon.SQS.Model.Attribute>()
        {
           new Amazon.SQS.Model.Attribute()
           {
             Name = "Policy",
             Value = policy
           }
        }
      });
    }

    static void RetrieveArchive(AmazonGlacier client)
    {
      // Initiate job.
      InitiateJobRequest initJobRequest = new InitiateJobRequest()
      {
        VaultName = vaultName,
        JobParameters = new JobParameters()
        {
          Type = "archive-retrieval",
          ArchiveId = archiveID,
          Description = "This job is to download archive updated as part of
getting started",
          SNSTopic = topicArn,
        }
      };
     InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);

      string jobId = initJobResponse.InitiateJobResult.JobId;

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

    private static void ProcessQueue(string jobId, AmazonGlacier client)
    {
      var receiveMessageRequest = new ReceiveMessageRequest() { QueueUrl =
queueUrl, MaxNumberOfMessages = 1 };
      bool jobDone = false;
      while (!jobDone)
      {
       var receiveMessageResponse = sqsClient.ReceiveMessage(receiveMessageRe
quest);
        if (receiveMessageResponse.ReceiveMessageResult.Message.Count == 0)
        {
          Thread.Sleep(1000 * 60);
          continue;
        }
       Message message = receiveMessageResponse.ReceiveMessageResult.Message[0];

       Dictionary<string, string> outerLayer = JsonConvert.DeserializeObject<Dic
tionary<string, string>>(message.Body);
        Dictionary<string, string> fields = JsonConvert.DeserializeObject<Dic
```

```
tionary<string, string>>(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); // 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, AmazonGlacier client)
    {
      GetJobOutputRequest getJobOutputRequest = new GetJobOutputRequest()
      {

        JobId = jobId,
        VaultName = vaultName
      };
     GetJobOutputResponse getJobOutputResponse = client.GetJobOutput(getJobOut
putRequest);
      GetJobOutputResult result = getJobOutputResponse.GetJobOutputResult;

      using (Stream webStream = result.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.Runtime;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;
using Amazon.SQS;
using Amazon.SQS.Model;
using Newtonsoft.Json;
using System.Collections.Specialized;
using System.Configuration;

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 file name to store downloaded archive
 ***";

    static AmazonSimpleNotificationService snsClient;
    static AmazonSQS 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)
    {
      AmazonGlacier client;
      NameValueCollection appConfig = ConfigurationManager.AppSettings;
      AmazonGlacierConfig config = new AmazonGlacierConfig();
      config.ServiceURL = appConfig["ServiceURL"];
      try
      {
       //using (client = new AmazonGlacierClient(Amazon.RegionEndpoint.USEast1))
```

```
        using (client = new AmazonGlacierClient(config))
        {
          // Setup SNS topic and SQS queue.
          SetupTopicAndQueue();
          DownloadAnArchive(archiveId, client);
        }
        Console.WriteLine("Operations successful. To continue, press Enter");
      }
      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
});
      }

      Console.WriteLine("To continue, press Enter");
      Console.ReadKey();
    }

    static void SetupTopicAndQueue()
    {
      // Set up Amazon SNS topic and Amazon SQS queue (in the same region where
 vault is located).
      snsClient = new AmazonSimpleNotificationServiceClient(Amazon.RegionEnd
point.USEast1);
      sqsClient = new AmazonSQSClient(Amazon.RegionEndpoint.USEast1);

      long ticks = DateTime.Now.Ticks;
      topicArn = snsClient.CreateTopic(new CreateTopicRequest { Name = "Glaci
erDownload-" + ticks }).CreateTopicResult.TopicArn;
      queueUrl = sqsClient.CreateQueue(new CreateQueueRequest() { QueueName =
"GlacierDownload-" + ticks }).CreateQueueResult.QueueUrl;
      queueArn = sqsClient.GetQueueAttributes(new GetQueueAttributesRequest()
{ QueueUrl = queueUrl, AttributeName = new List<string> { "QueueArn" }
}).GetQueueAttributesResult.QueueARN;

      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("{QuernArn}", queueArn).Replace("{Topi
cArn}", topicArn);
      sqsClient.SetQueueAttributes(new SetQueueAttributesRequest()
      {
        QueueUrl = queueUrl,
        Attribute = new List<Amazon.SQS.Model.Attribute>()
        {
           new Amazon.SQS.Model.Attribute()
          {
            Name = "Policy",
            Value = policy
          }
        }
      });
    }

    static void DownloadAnArchive(string archiveId, AmazonGlacier client)
    {
      // Initiate job.
      InitiateJobRequest initJobRequest = new InitiateJobRequest()
      {

        VaultName = vaultName,
        JobParameters = new JobParameters()
        {
          Type = "archive-retrieval",
          ArchiveId = archiveId,
          Description = "This job is to download archive updated as part of
getting started",
          SNSTopic = topicArn,
        }
      };
     InitiateJobResponse initJobResponse = client.InitiateJob(initJobRequest);

      string jobId = initJobResponse.InitiateJobResult.JobId;

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

    private static void ProcessQueue(string jobId, AmazonGlacier client)
    {
      var receiveMessageRequest = new ReceiveMessageRequest() { QueueUrl =
queueUrl, MaxNumberOfMessages = 1 };
      bool jobDone = false;
      while (!jobDone)
      {
       var receiveMessageResponse = sqsClient.ReceiveMessage(receiveMessageRe
quest);
        if (receiveMessageResponse.ReceiveMessageResult.Message.Count == 0)
        {
          Thread.Sleep(1000 * 60);
          continue;
        }
       Message message = receiveMessageResponse.ReceiveMessageResult.Message[0];

       Dictionary<string, string> outerLayer = JsonConvert.DeserializeObject<Dic
tionary<string, string>>(message.Body);
        Dictionary<string, string> fields = JsonConvert.DeserializeObject<Dic
tionary<string, string>>(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,
AmazonGlacier client)
    {
      long partSize = 4 * (long)Math.Pow(2, 20);  // 4 MB.
      using (Stream fileToSave = new FileStream(fileName, FileMode.Create,
FileAccess.Write))
      {
        GetJobOutputResponse getJobOutputResponse = null;
        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 = client.GetJobOutput(getJobOutputRequest);

          GetJobOutputResult result = getJobOutputResponse.GetJobOutputResult;


          using (Stream webStream = result.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 a Job (POST jobs) (p. 189).
2. After the job completes, download the archive data. For more information, see Get Job Output (GET output) (p. 203).

Deleting an Archive in Amazon Glacier

Topics

- Deleting an Archive in Amazon Glacier Using the AWS SDK for Java (p. 104)
- Deleting an Archive in Amazon Glacier Using the AWS SDK for .NET (p. 106)
- Deleting an Archive Using the REST API (p. 108)

You cannot delete an archive using the Amazon Glacier management console. To delete an archive you must write code to make a delete request using either the REST API directly or the AWS SDK for Java and .NET wrapper libraries.

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. 35).

After you submit a request to delete an archive, subsequent requests to initiate a retrieval of this archive will fail. Any archive retrievals that are in progress for this archive may or may 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.

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. 110).
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. 164).

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. 112). 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

Topics

- Deleting an Archive Using the High-Level API of the AWS SDK for .NET (p. 106)
- Deleting an Archive Using the Low-Level API AWS SDK for .NET (p. 107)

Both the high-level and low-level APIs (p. 109) provided by the AWS SDK for .NET provide a method to delete an archive.

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. 114). 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.USEast1);
```

```
        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. 109).
3. Execute the `DeleteArchive` method by providing the request object as a parameter.

The following C# code snippet illustrates the preceding steps.

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

DeleteArchiveRequest request = new DeleteArchiveRequest()
{
  VaultName = vaultName,
  ArchiveId = archiveId
};

DeleteArchiveResponse response = client.DeleteArchive(request);
```

Note
For information about the underlying REST API, see Delete Archive (DELETE archive) (p. 164).

Example: Deleting an Archive Using the Low-Level API of the AWS SDK for .NET

The following C# code example uses the low-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. 114). 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.USEast1))

        {
          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. 164).
- For information about using the Amazon Glacier REST API, see API Reference for Amazon Glacier (p. 120).

Using the AWS SDKs with Amazon Glacier

Topics

- What is the Low-Level API? (p. 109)
- What is the High-Level API? (p. 110)
- When to Use High-Level and Low-Level API? (p. 110)
- Using the AWS SDK for Java with Amazon Glacier (p. 110)
- Using the AWS SDK for .NET with Amazon Glacier (p. 113)

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 libraries, you only need to provide your AWS security credentials in your code and the libraries compute the necessary signature and include it in the request sent to 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.

Note
Amazon Glacier is supported by the AWS SDK for Java, .NET, PHP, and Python (Boto). Examples of working with Amazon Glacier using the Java and .NET SDKs are provided throughout this developer guide. For more information on the AWS SDKs, see Sample Code & Libraries.

The AWS SDK 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. 120)) 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. 110) and Using the AWS SDK for .NET with Amazon Glacier (p. 113).

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. 110) and Using the AWS SDK for .NET with Amazon Glacier (p. 113).

When to Use 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

Topics

- Using the Low-Level API (p. 111)
- Using the High-Level API (p. 111)
- Running Java Examples for Amazon Glacier Using Eclipse (p. 112)
- Setting the Endpoint (p. 112)

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. 109), 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.

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. 120)). 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. 142)). 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-east-1.amazonaws.com/");

CreateVaultRequest request = new CreateVaultRequest()
    .withAccountId("-")
    .withVaultName(vaultName);
CreateVaultResult result = client.createVault(createVaultRequest);
```

All the low-level samples in the guide use this pattern.

Note
The preceding code segment specifies `AccountID` when creating the request. However, when using the AWS SDK for Java, the `AccountId` in the request is optional, and therefore all the low-level examples in this guide don't set this value. The `AccountId` is the AWS Account ID. This value must match the AWS Account ID associated with the credentials used to sign the request. You can specify either the AWS Account ID or optionally a '-', in which case Amazon Glacier uses the AWS Account ID associated with the credentials used to sign the request. If you specify your Account ID, do not include hyphens in it. When using AWS SDK for Java, if you don't provide the account ID, the library sets the account ID to '-'.

Using the High-Level API

To further simplify your application development, the AWS SDK for Java provides the `ArchiveTransferManager` class that implements a higher-level abstraction for the some of the methods in the low-level API. It provides useful methods, such as the `upload` and `download` methods for archive operations.

For example, the following Java code snippet uses the `upload` high-level method to upload an archive.

```
String vaultName = "examplevault";
String archiveToUpload = "c:/folder/exampleArchive.zip";

ArchiveTransferManager atm = new ArchiveTransferManager(client, credentials);
String archiveId = atm.upload(vaultName, "Tax 2012 documents", new
File(archiveToUpload)).getArchiveId();
```

Note that any operations you perform apply to the region you specified when creating the `ArchiveTransferManager` object. If you don't specify any region, the AWS SDK for Java sets the `US East (Northern Virginia) Region` 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 `AmazonGlacier-Client` 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

Topics

- Using the Low-Level API (p. 113)
- Using the High-Level API (p. 114)
- Running Code Examples (p. 114)
- Setting the Endpoint (p. 114)

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. 109), 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.

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. 120)). 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. 142)). 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", archiveToUp
load).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 *AWS Empty Project* template.
3	Replace the code in the project file, `Program.cs`, with the code in the section you are reading.
4	Run the code. Verify that the object is created using the AWS Management Console. For more information about AWS Management Console, go to http://aws.amazon.com/console/.

Setting the Endpoint

By default, the AWS SDK for .NET sets the endpoint to the US East (Northern Virginia) Region (`https://glacier.us-east-1.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.USW
est2);
```

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).

Access Control Using AWS Identity and Access Management (IAM)

An AWS account has full permission to perform all actions on the vaults in the account. However, the AWS Identity and Access Management (IAM) users don't have any permissions by default.

IAM helps you securely control access to Amazon Web Services and your account resources. With IAM, you can create multiple IAM users under the umbrella of your AWS account. To learn more about IAM and its features, go to What Is IAM?

Every user you create in the IAM system starts with no permissions. In other words, by default, users can do nothing. A *permission* is a general term we use to mean the ability to perform an action against a resource. The Amazon Glacier API (see API Reference for Amazon Glacier (p. 120)) enables a list of actions you can perform. However, unless you explicitly grant a user permissions, that user cannot perform any of these actions. You grant a permission to a user with a policy. A policy is a document that formally states one or more permissions. For more information about IAM policies, go to Overview of Policies.

You write a policy using the access policy language that IAM uses, and then attach the policy to the user or a group in your AWS account. For more information about the policy language, go to The Access Policy Language in the *Using IAM* guide.

The Element Descriptions section of the *Using IAM* guide describes elements you can use in a policy. The following information about some of the policy elements is specific to Amazon Glacier:

- **Resource—**The object or objects the policy covers. You identify resources using the following Amazon Resource Name (ARN) format.

  ```
  arn:aws:<vendor>:<region>:<namespace>:<relative-id>
  ```

  In Amazon Glacier, `vendor` is the product name "glacier", `namespace` is the account ID and the `<relative-id>` is always "vaults/<vault-name>", "vaults/<vault-name-prefix*>" or "vaults/*".

  Amazon Glacier supports policies at vault level. That is, in an IAM policy, the resource you specify can be a specific vault or a set of vaults in a specific AWS region. Amazon Glacier does not support archive-level permissions. The following table shows an example ARNs for the vault-level resources.

Resource	Example ARN
Vault	arn:aws:glacier:us-east-1:012345678901:vaults/examplevault
Vaults names starting with "example"	arn:aws:glacier:us-east-1:012345678901:vaults/example*
All vaults in a region	arn:aws:glacier:us-east-1:012345678901:vaults/*

- **Action—**The specific type or types of action allowed or denied. For a complete list of Amazon Glacier actions, see API Reference for Amazon Glacier (p. 120).

Action	API Link
glacier:AbortMultipartUpload	Abort Multipart Upload (DELETE uploadID) (p. 177)
glacier:CompleteMultipartUpload	Complete Multipart Upload (POST uploadID) (p. 174)
glacier:CreateVault	Create Vault (PUT vault) (p. 142)
glacier:DeleteArchive	Delete Archive (DELETE archive) (p. 164)
glacier:DeleteVault	Delete Vault (DELETE vault) (p. 144)
glacier:DeleteVaultNotifications	Delete Vault Notifications (DELETE notification-configuration) (p. 159)
glacier:DescribeJob	Describe Job (GET JobID) (p. 196)
glacier:DescribeVault	Describe Vault (GET vault) (p. 146)
glacier:GetJobOutput	Get Job Output (GET output) (p. 203)
glacier:GetVaultNotifications	Get Vault Notifications (GET notification-configuration) (p. 156)
glacier:InitiateMultipartUpload	Initiate Multipart Upload (POST multipart-uploads) (p. 167)
glacier:InitiateJob	Initiate a Job (POST jobs) (p. 189)
glacier:ListJobs	List Jobs (GET jobs) (p. 210)
glacier:ListMultipartUploads	List Multipart Uploads (GET multipart-uploads) (p. 184)
glacier:ListParts	List Parts (GET uploadID) (p. 179)
glacier:ListVaults	List Vaults (GET vaults) (p. 149)
glacier:SetVaultNotifications	Set Vault Notification Configuration (PUT notification-configuration) (p. 154)
glacier:UploadArchive	Upload Archive (POST archive) (p. 161)
glacier:UploadMultipartPart	Upload Part (PUT uploadID) (p. 170)

This section provides example IAM policies that illustrate how to grant a user permission to perform specific Amazon Glacier actions. You can then attach these policies to a user to whom you want to grant access permissions.

Example Policies

Example 1: Grant user permission to initiate jobs and get job output

The following user policy allows the user to initiate jobs and get job output. Note that the `glacier:InitiateJob` action allows this user to both get a vault inventory (list of archives in the `examplevault` vault) and also download any archive from the vault. The policy identifies the vault resource by its ARN. You can get the vault ARN from the Amazon Glacier console or programmatically by calling either the `Describe Vault` (see Describe Vault (GET vault) (p. 146) or the `List Vaults` (see List Vaults (GET vaults) (p. 149)) API. The policy also grants `glacier:DescribeJob` to allow the use to get job information after initiating a job.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
     "Resource": "arn:aws:glacier:us-east-1:111111111111:vaults/examplevault",

      "Action":["glacier:InitiateJob",
                "glacier:GetJobOutput",
                "glacier:DescribeJob"]
    }
  ]
}
```

Example 2: Grant user permission to create a vault and configure notifications

The following user policy allows the user to create a vault and configure notifications. The policy also allows the user to obtain list of vaults in a specific region created under the AWS account, or get vault description of a specific vault.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": "arn:aws:glacier:us-east-1:111111111111:vaults/*",
      "Action":["glacier:CreateVault",
                "glacier:SetVaultNotifications",
                "glacier:GetVaultNotifications",
                "glacier:DeleteVaultNotifications",
                "glacier:DescribeVault",
                "glacier:ListVaults"]
    }
  ]
}
```

Example 3: Grant user permission to upload archives to a specific vault

The following user policy allows the user to upload archives to a specific vault. The user can upload archives in a single API call or use the multipart upload API to upload large archives. The policy grants a user permission to perform various multipart upload operations. Note that the user can upload an archive; however, the policy does not allow the user to download an archive from the vault. The user can download an archive only if the policy allows the `glacier:InitiateJob` and the `glacier:GetJobOutput` operations.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
     "Resource": "arn:aws:glacier:us-east-1:111111111111:vaults/examplevault",

      "Action":["glacier:UploadArchive",
                "glacier:InitiateMultipartUpload",
                "glacier:UploadMultipartPart",
                "glacier:ListParts",
                "glacier:ListMultipartUploads",
                "glacier:CompleteMultipartUpload"]
    }
  ]
}
```

Example 4: Grant user permission full permission on a specific vault

The following user policy allows the user to perform all the operations on `examplevault`, including deleting the vault.

```
{
 "Version":"2012-10-17",
 "Statement": [
    {
      "Effect": "Allow",
     "Resource": "arn:aws:glacier:us-east-1:111111111111:vaults/examplevault",

      "Action":["glacier:*"]
    }
  ]
}
```

API Reference for Amazon Glacier

Topics

- Common Request Headers (p. 120)
- Common Response Headers (p. 123)
- Signing Requests (p. 123)
- Computing Checksums (p. 127)
- Error Responses (p. 139)
- Vault Operations (p. 142)
- Archive Operations (p. 161)
- Multipart Upload Operations (p. 166)
- Job Operations (p. 189)

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. 123).

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. 109).

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`	The header that is required to sign requests. Amazon Glacier requires Signature Version 4. For more information, see Signing Requests (p. 123). Type: String	Yes
`Content-Length`	The length of the request body (without the headers). Type: String Condition: Required only for the Upload Archive (POST archive) (p. 161) API.	Conditional
`Date`	The date that can be used to create the signature contained in the `Authorization` header. If the `Date` header is to be used for signing it must be specified in the ISO 8601 basic format. In this case, the `x-amz-date` header is not needed. Note that when `x-amz-date` is present, it always overrides the value of the `Date` header. If the Date header is not used for signing, it can be one of the full date formats specified by RFC 2616, section 3.3. For example, the following date/time `Sun, 25 Mar 2012 12:00:00 GMT` is a valid date/time header for use with Amazon Glacier. If you are using the `Date` header for signing, then it must be in the ISO 8601 basic `YYYYMMDD'T'HHMMSS'Z'` format. Type: String Condition: If `Date` is specified but is not in ISO 8601 basic format, then you must also include the `x-amz-date` header. If `Date` is specified in ISO 8601 basic format, then this is sufficient for signing requests and you do not need the `x-amz-date` header. For more information, see Handling Dates in Signature Version 4 in the *Amazon Web Services Glossary*.	Conditional
`Host`	This header specifies the service endpoint to which you send your requests. The value must be of the form "`glacier.`*`region`*`.amazonaws.com`", where region is replaced with a region designation such as `us-east-1`. Type: String	Yes

Header Name	Description	Required
`x-amz-content-sha256`	The computed SHA256 checksum of an entire payload that is uploaded with either Upload Archive (POST archive) (p. 161) or Upload Part (PUT uploadID) (p. 170). This header is not the same as the `x-amz-sha256-tree-hash` header, though, for some small payloads the values are the same. When `x-amz-content-sha256` is required, both `x-amz-content-sha256` and `x-amz-sha256-tree-hash` must be specified. Type: String Condition: Required for streaming API, Upload Archive (POST archive) (p. 161) and Upload Part (PUT uploadID) (p. 170).	Conditional
`x-amz-date`	The date used to create the signature in the Authorization header. The format must be ISO 8601 basic in the `YYYYMMDD'T'HHMMSS'Z'` format. For example, the following date/time `20120325T120000Z` is a valid `x-amz-date` for use with Amazon Glacier. Type: String Condition: `x-amz-date` is optional for all requests; it can be used to override the date used for signing requests. If the `Date` header is specified in the ISO 8601 basic format, then `x-amz-date` is not needed. When `x-amz-date` is present, it always overrides the value of the `Date` header. For more information, see Handling Dates in Signature Version 4 in the *Amazon Web Services Glossary*.	Conditional
`x-amz-glacier-version`	The Amazon Glacier API version to use. The current version is `2012-06-01`. Type: String	Yes
`x-amz-sha256-tree-hash`	The computed SHA256 tree-hash checksum for an uploaded archive (Upload Archive (POST archive) (p. 161)) or archive part (Upload Part (PUT uploadID) (p. 170)). For more information about calculating this checksum, see Computing Checksums (p. 127). Type: String Default: None Condition: Required for Upload Archive (POST archive) (p. 161) and Upload Part (PUT uploadID) (p. 170).	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, `Sun, 25 Mar 2012 12:00:00 GMT`. The format of the date must be one of the full date formats specified by RFC 2616, section 3.3. Note that `Date` returned may drift slightly from other dates, so for example, the date returned from an Upload Archive (POST archive) (p. 161) 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. 127). Type: String

Signing Requests

Topics

- Example Signature Calculation (p. 124)
- Calculating Signatures for the Streaming Operations (p. 125)

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.

Example Signature Calculation

The following example walks you through the details of creating a signature for Create Vault (PUT vault) (p. 142). 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 (Northern 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. 123) is:

```
PUT
/-/vaults/examplevault

host:glacier.us-east-1.amazonaws.com
x-amz-date:20120525T002453Z
x-amz-glacier-version:2012-06-01

host;x-amz-date;x-amz-glacier-version
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

The last line of the canonical request is the hash of the request body. Also, note the empty third line in the canonical request. This is because there are no query parameters for this API.

The *string to sign* for Task 2: Create a String to Sign (p. 123) is:

```
AWS4-HMAC-SHA256
20120525T002453Z
20120525/us-east-1/glacier/aws4_request
5f1da1a2d0feb614dd03d71e87928b8e449ac87614479332aced3a701f916743
```

The first line of the *string to sign* is the algorithm, the second line is the time stamp, the third line is the *credential scope*, and the last line is a hash of the canonical request from Task 1: Create a Canonical Request (p. 123). The service name to use in the credential scope is `glacier`.

For Task 3: Create a Signature (p. 124), 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 `AKIAIOS-FODNN7EXAMPLE`, 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. 161) and Upload Part (PUT uploadID) (p. 170) 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. 127)). 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. 161), 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 (Northern 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. 123) 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. 124). 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

Topics

- Tree Hash Example 1: Uploading an archive in a single request (p. 128)
- Tree Hash Example 2: Uploading an archive using a multipart upload (p. 128)
- Computing the Tree Hash of a File (p. 129)
- Receiving Checksums When Downloading Data (p. 137)

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. 126).

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.

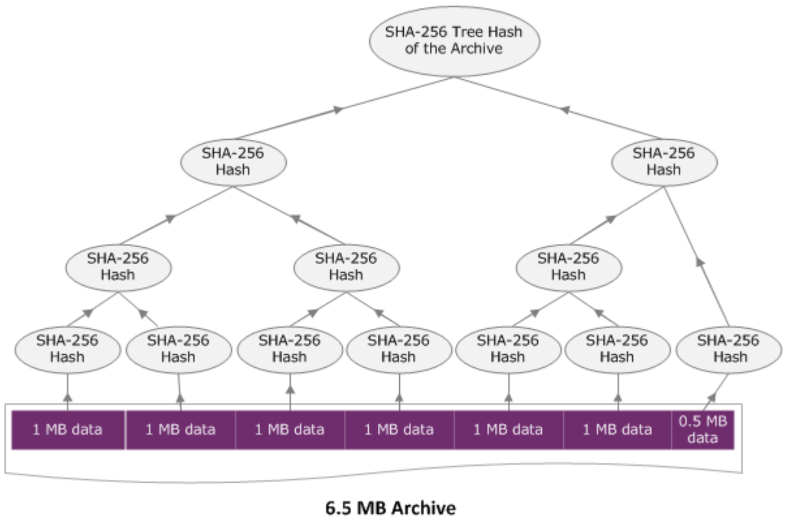
Note
This section describes a way to compute the SHA-256 tree hash. However, you may use any procedure as long as it produces the same result.

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.

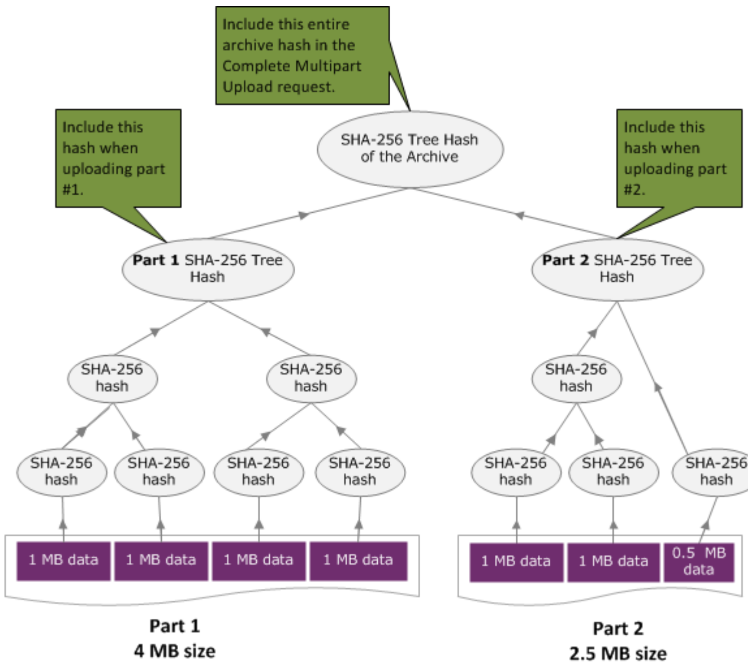
Tree Hash Example 1: Uploading an archive in a single request

When you upload an archive in a single request using the Upload Archive API (see Upload Archive (POST archive) (p. 161)), 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. 170) 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. 174)) request with a tree hash of the entire archive in the `x-amz-sha256-tree-hash` request header.


Include this entire archive hash in the Complete Multipart Upload request.
SHA-256 Tree Hash of the Archive
Include this hash when uploading part #1.
Include this hash when uploading part #2.
Part 1 SHA-256 Tree Hash
Part 2 SHA-256 Tree Hash
SHA-256 hash
1 MB data
0.5 MB data
Part 1
4 MB size
Part 2
2.5 MB size

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.

Example 1: Java Example

The following example shows how to calculate the SHA256 tree hash of a file using Java. You can run this example by either supplying a file location as an argument or you can use the `TreeHashExample.computeSHA256TreeHash` method directly from your code.

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class TreeHashExample {

static final int ONE_MB = 1024 * 1024;

    /**
    * Compute the Hex representation of the SHA-256 tree hash for the specified

     * File
     *
     * @param args
     *            args[0]: a file to compute a SHA-256 tree hash for
     */
    public static void main(String[] args) {

        if (args.length < 1) {
            System.err.println("Missing required filename argument");
            System.exit(-1);
        }

        File inputFile = new File(args[0]);
        try {

            byte[] treeHash = computeSHA256TreeHash(inputFile);
            System.out.printf("SHA-256 Tree Hash = %s\n", toHex(treeHash));

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

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

    /**
     * Computes the SHA-256 tree hash for the given file
     *
     * @param inputFile
     *            a File to compute the SHA-256 tree hash for
     * @return a byte[] containing the SHA-256 tree hash
     * @throws IOException
     *             Thrown if there's an issue reading the input file
     * @throws NoSuchAlgorithmException
     */
    public static byte[] computeSHA256TreeHash(File inputFile) throws IOExcep
tion,
            NoSuchAlgorithmException {

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

    /**
     * Computes a SHA256 checksum for each 1 MB chunk of the input file. This
     * includes the checksum for the last chunk even if it is smaller than 1
MB.
     *
     * @param file
     *            A file to compute checksums on
     * @return a byte[][] containing the checksums of each 1 MB chunk
     * @throws IOException
     *             Thrown if there's an IOException when reading the file
     * @throws NoSuchAlgorithmException
     *             Thrown if SHA-256 MessageDigest can't be found
     */
    public static byte[][] getChunkSHA256Hashes(File file) throws IOException,

            NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");

        long numChunks = file.length() / ONE_MB;
        if (file.length() % ONE_MB > 0) {
            numChunks++;
        }

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

        byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
        FileInputStream fileStream = null;

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

            int bytesRead;
            int idx = 0;

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

            return chunkSHA256Hashes;

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

    /**
     * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
     * checksums.
     *
     * This method uses a pair of arrays to iteratively compute the tree hash
     * level by level. Each iteration takes two adjacent elements from the
     * previous level source array, computes the SHA-256 hash on their
     * concatenated value and places the result in the next level's destination

     * array. At the end of an iteration, the destination array becomes the
     * source array for the next level.
     *
     * @param chunkSHA256Hashes
     *            An array of SHA-256 checksums
     * @return A byte[] containing the SHA-256 tree hash for the input chunks
     * @throws NoSuchAlgorithmException
     *             Thrown if SHA-256 MessageDigest can't be found
     */
    public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
            throws NoSuchAlgorithmException {

        MessageDigest md = MessageDigest.getInstance("SHA-256");

        byte[][] prevLvlHashes = chunkSHA256Hashes;

        while (prevLvlHashes.length > 1) {

            int len = prevLvlHashes.length / 2;
            if (prevLvlHashes.length % 2 != 0) {
                len++;
            }

            byte[][] currLvlHashes = new byte[len][];

            int j = 0;
            for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

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

                    // Calculate a digest of the concatenated nodes
                    md.reset();
                    md.update(prevLvlHashes[i]);
                    md.update(prevLvlHashes[i + 1]);
                    currLvlHashes[j] = md.digest();
                } else { // Take care of remaining odd chunk
                    currLvlHashes[j] = prevLvlHashes[i];
                }
            }

            prevLvlHashes = currLvlHashes;
        }

        return prevLvlHashes[0];
    }

    /**
     * Returns the hexadecimal representation of the input byte array
     *
     * @param data
     *            a byte[] to convert to Hex characters
     * @return A String containing Hex characters
     */
    public static String toHex(byte[] data) {
        StringBuilder sb = new StringBuilder(data.length * 2);

        for (int i = 0; i < data.length; i++) {
            String hex = Integer.toHexString(data[i] & 0xFF);

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

Example 2: C# .NET Example

The following example shows how to calculate the SHA256 tree hash of a file. You can run this example by supplying a file location as an argument.

```
using System;
using System.IO;

using System.Security.Cryptography;

namespace ExampleTreeHash
{
    class Program
    {
        static int ONE_MB = 1024 * 1024;

        /**
        * Compute the Hex representation of the SHA-256 tree hash for the
        * specified file
        *
        * @param args
        *            args[0]: a file to compute a SHA-256 tree hash for
        */
        public static void Main(string[] args)
        {
            if (args.Length < 1)
            {
                Console.WriteLine("Missing required filename argument");
                Environment.Exit(-1);
            }
            FileStream inputFile = File.Open(args[0], FileMode.Open, FileAc
cess.Read);
            try
            {
                byte[] treeHash = ComputeSHA256TreeHash(inputFile);
                Console.WriteLine("SHA-256 Tree Hash = {0}", BitConverter.To
String(treeHash).Replace("-", "").ToLower());
                Console.ReadLine();
                Environment.Exit(-1);
            }
            catch (IOException ioe)
            {
                Console.WriteLine("Exception when reading from file {0}: {1}",

                    inputFile, ioe.Message);
                Console.ReadLine();
                Environment.Exit(-1);
            }
            catch (Exception e)
            {
                Console.WriteLine("Cannot locate MessageDigest algorithm for
SHA-256: {0}",
                    e.Message);
                Console.WriteLine(e.GetType());
                Console.ReadLine();
                Environment.Exit(-1);
            }
            Console.ReadLine();
        }


        /**
         * Computes the SHA-256 tree hash for the given file
         *
         * @param inputFile
         *            A file to compute the SHA-256 tree hash for
         * @return a byte[] containing the SHA-256 tree hash
         */
        public static byte[] ComputeSHA256TreeHash(FileStream inputFile)
        {
            byte[][] chunkSHA256Hashes = GetChunkSHA256Hashes(inputFile);
            return ComputeSHA256TreeHash(chunkSHA256Hashes);
        }


        /**
         * Computes a SHA256 checksum for each 1 MB chunk of the input file.
This
         * includes the checksum for the last chunk even if it is smaller than
 1 MB.
         *
         * @param file
         *            A file to compute checksums on
         * @return a byte[][] containing the checksums of each 1MB chunk
         */
        public static byte[][] GetChunkSHA256Hashes(FileStream file)
        {
            long numChunks = file.Length / ONE_MB;
            if (file.Length % ONE_MB > 0)
            {
                numChunks++;
            }

            if (numChunks == 0)
            {
                return new byte[][] { CalculateSHA256Hash(null, 0) };
            }
            byte[][] chunkSHA256Hashes = new byte[(int)numChunks][];

            try
            {
                byte[] buff = new byte[ONE_MB];

                int bytesRead;
                int idx = 0;

                while ((bytesRead = file.Read(buff, 0, ONE_MB)) > 0)
                {
                    chunkSHA256Hashes[idx++] = CalculateSHA256Hash(buff,
bytesRead);
                }
                return chunkSHA256Hashes;
            }
            finally
            {
                if (file != null)
                {
                    try
                    {
                        file.Close();
                    }
                    catch (IOException ioe)
                    {
                        throw ioe;
                    }
                }
            }

        }

        /**
         * Computes the SHA-256 tree hash for the passed array of 1MB chunk
         * checksums.
         *
         * This method uses a pair of arrays to iteratively compute the tree
hash
         * level by level. Each iteration takes two adjacent elements from the

         * previous level source array, computes the SHA-256 hash on their
        * concatenated value and places the result in the next level's destin
ation
        * array. At the end of an iteration, the destination array becomes the

         * source array for the next level.
         *
         * @param chunkSHA256Hashes
         *            An array of SHA-256 checksums
         * @return A byte[] containing the SHA-256 tree hash for the input
chunks
         */
        public static byte[] ComputeSHA256TreeHash(byte[][] chunkSHA256Hashes)

        {
            byte[][] prevLvlHashes = chunkSHA256Hashes;
            while (prevLvlHashes.GetLength(0) > 1)
            {

                int len = prevLvlHashes.GetLength(0) / 2;
                if (prevLvlHashes.GetLength(0) % 2 != 0)
                {
                    len++;
                }

                byte[][] currLvlHashes = new byte[len][];

                int j = 0;
               for (int i = 0; i < prevLvlHashes.GetLength(0); i = i + 2, j++)

                {

                    // If there are at least two elements remaining
                    if (prevLvlHashes.GetLength(0) - i > 1)
                    {
                        // Calculate a digest of the concatenated nodes
                        byte[] firstPart = prevLvlHashes[i];
                        byte[] secondPart = prevLvlHashes[i + 1];
                        byte[] concatenation = new byte[firstPart.Length + 
secondPart.Length];
                        System.Buffer.BlockCopy(firstPart, 0, concatenation, 
0, firstPart.Length);
                        System.Buffer.BlockCopy(secondPart, 0, concatenation, 
firstPart.Length, secondPart.Length);

                        currLvlHashes[j] = CalculateSHA256Hash(concatenation, 
concatenation.Length);

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

                prevLvlHashes = currLvlHashes;
            }

            return prevLvlHashes[0];
        }

        public static byte[] CalculateSHA256Hash(byte[] inputBytes, int count)

        {
            SHA256 sha256 = System.Security.Cryptography.SHA256.Create();
            byte[] hash = sha256.ComputeHash(inputBytes, 0, count);
            return hash;
        }
    }
}
```

Receiving Checksums When Downloading Data

When you retrieve an archive using the Initiate Job API (see Initiate a Job (POST jobs) (p. 189)), 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. 203)), 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. 138).

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

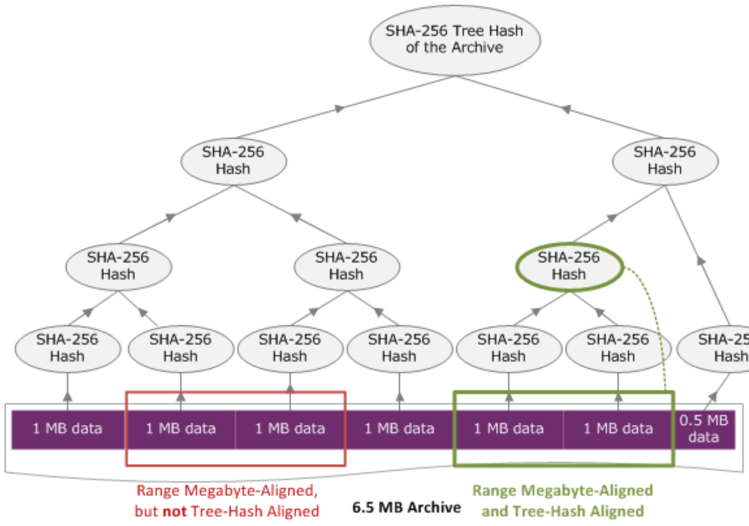
The following cases describe when you receive a checksum value when you download your archive data:

- 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.


SHA-256 Tree Hash of the Archive
SHA-256 Hash
1 MB data
0.5 MB data
Range Megabyte-Aligned, but not Tree-Hash Aligned
6.5 MB Archive
Range Megabyte-Aligned and Tree-Hash Aligned

Tree-Hash Aligned Range Specification

This section gives the exact specification for what constitutes a tree-hash aligned range. Tree-hash aligned ranges are important when you are downloading a portion of an archive and you specify the range of data to retrieve and the range to download from the retrieved data. If both of these ranges are tree-hash aligned, then you will receive checksum data when you download the data.

A range [*A*, *B*] is *tree-hash aligned* with respect to an archive if and only if when a new tree hash is built over [*A*, *B*], the root of the tree hash of that range is equivalent to a node in the tree hash of the whole archive. You can see this shown in the diagram in Tree Hash Example: Retrieving an archive range that is tree-hash aligned (p. 138). In this section, we provide the specification for tree-hash alignment.

Consider [*P*, *Q*) as the range query for an archive of *N* megabytes (MB) and *P* and *Q* are multiples of one MB. Note that the actual inclusive range is [*P* MB, *Q* MB – 1 byte], but for simplicity, we show it as [*P*, *Q*). With these considerations, then

- If *P* is an odd number, there is only one possible tree-hash aligned range—that is [*P*, *P* + 1 MB).
- If *P* is an even number and *k* is the maximum number, where *P* can be written as 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 <= *i* <= *k*) and where $P + 2^i < N$, then $[P, Q + 2^i)$ is a tree-hash aligned range.
  - *P* = 0 is the special case where *A* = 2[lgN]*0

Error Responses

In the event of an error, 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 used in the request URI. For more information, see Access Control Using AWS Identity and Access Management (IAM) (p. 116).	`403 Forbidden`	Client
`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 or account 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

Code	Description	HTTP Status Code	Type
`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. 196) request for a job that does not exist. That is, you specify a job ID that does not exist.

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-K3CGl6fuSm6VzW9T7esGQfco8nUXVY
wS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVEXAMPLEbadJobID HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 12:00:00 GMT
{
  "code": "ResourceNotFoundException",
  "message": "The job ID was not found: HkF9p6o7yjhFx-K3CGl6fuSm6VzW9T7es
GQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVEXAMPLEbadJobID",
  "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, 25 Mar 2012 12:00:00 GMT
{
  "code": "ResourceNotFoundException",
  "message": "Vault not found for ARN: arn:aws:glacier:us-east-
1:012345678901: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. 210) 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-east-1.amazonaws.com
Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
```

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 12:00:00 GMT
{
  "code": "InvalidParameterValueException",
  "message": "The job status code is not valid: finished",
  "type: "Client"
}
```

Vault Operations

Topics

- Create Vault (PUT vault) (p. 142)
- Delete Vault (DELETE vault) (p. 144)
- Describe Vault (GET vault) (p. 146)
- List Vaults (GET vaults) (p. 149)
- Set Vault Notification Configuration (PUT notification-configuration) (p. 154)
- Get Vault Notifications (GET notification-configuration) (p. 156)
- Delete Vault Notifications (DELETE notification-configuration) (p. 159)

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: 0
x-amz-glacier-version: 2012-06-01
```

Note
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 either specify 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 dashes in it.

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. 120).

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. 123).

Name	Description
`Location`	The relative URI path of the vault that was created. Type: String

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses (p. 139).

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-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Content-Length: 0
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 12:02:00 GMT
Location: /111122223333/vaults/examplevault
```

Related Sections

- List Vaults (GET vaults) (p. 149)
- Delete Vault (DELETE vault) (p. 144)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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. 146) 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.

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` 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 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 dashes in it.

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. 120).

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. 123).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses (p. 139).

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-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

Example Response

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 12:02:00 GMT
```

Related Sections

- Create Vault (PUT vault) (p. 142)
- List Vaults (GET vaults) (p. 149)
- Initiate a Job (POST jobs) (p. 189)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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` 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 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 dashes in it.

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. 120).

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. 123).

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 a Job (POST jobs) (p. 189).

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. 139).

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-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

Example Response

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 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-east-1:012345678901:vaults/examplevault",
  "VaultName" : "examplevault"
}
```

Related Sections

- Create Vault (PUT vault) (p. 142)

- List Vaults (GET vaults) (p. 149)
- Delete Vault (DELETE vault) (p. 144)
- Initiate a Job (POST jobs) (p. 189)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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. 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` 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 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 dashes in it.

Request Parameters

This operation uses the following request parameters.

Name	Description	Required
`limit`	The maximum number of items returned in the response. If you don't specify a value, the List Vaults operation returns up to 1,000 items. Type: String Constraints: Minimum integer value of 1. Maximum integer value of 1000.	No

Name	Description	Required
`marker`	A string used for pagination. `marker` specifies the vault ARN after which the listing of vaults should begin. (The vault specified by `marker` is not included in the returned list.) Get the `marker` value from a previous List Vaults response. You need to include the `marker` only if you are continuing the pagination of results started in a previous List Vaults request. Specifying an empty value ("") for the marker returns a list of vaults starting from the first vault. Type: String Constraints: None	No

Request Headers

This operation uses only request headers that are common to all operations. For information about common request headers, see Common Request Headers (p. 120).

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. 123).

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 a Job (POST jobs) (p. 189).

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. 139).

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-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 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-east-1:012345678901: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-east-1:012345678901: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-east-1:012345678901: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-east-1:012345678901:vaults/ex
amplevault1 HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 12:02:00 GMT
Content-Type: application/json
Content-Length: 497

{
  "Marker": "arn:aws:glacier:us-east-1:012345678901: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-east-1:012345678901: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-east-1:012345678901:vaults/examplevault2",

    "VaultName": "examplevault2"
   }
  ]
}
```

Related Sections

- Create Vault (PUT vault) (p. 142)
- Delete Vault (DELETE vault) (p. 144)
- Initiate a Job (POST jobs) (p. 189)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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. So 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 a Job (POST jobs) (p. 189)). 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. 196).
- **`InventoryRetrievalCompleted`**— This event occurs when a job that was initiated for an inventory retrieval is completed (Initiate a Job (POST jobs) (p. 189)). 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. 196).

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` 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 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 dashes in it.

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. 120).

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. 120).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses (p. 139).

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-east-1:012345678901:mytopic`.

```
PUT /-/vaults/examplevault/notification-policy HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
   "Events": ["ArchiveRetrievalCompleted", "InventoryRetrievalCompleted"],
   "SNSTopic": "arn:aws:sns:us-east-1:012345678901:mytopic"
}
```

Example Response

A successful response returns a `204 No Content`.

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 12:00:00 GMT
```

Related Sections

- Get Vault Notifications (GET notification-configuration) (p. 156)
- Delete Vault Notifications (DELETE notification-configuration) (p. 159)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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. 154). 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` 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 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 dashes in it.

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. 120).

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. 123).

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. 154).

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. 139).

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-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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-east-1:012345678901:mytopic`.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 12:00:00 GMT
Content-Type: application/json
Content-Length: 150

{
  "Events": [
    "ArchiveRetrievalCompleted",
    "InventoryRetrievalCompleted"
  ],
  "SNSTopic": "arn:aws:sns:us-east-1:012345678901:mytopic"
}
```

Related Sections

- Delete Vault Notifications (DELETE notification-configuration) (p. 159)
- Set Vault Notification Configuration (PUT notification-configuration) (p. 154)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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. 154). 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` 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 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 dashes in it.

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. 120).

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. 123).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses (p. 139).

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-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

Example Response

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 12:00:00 GMT
```

Related Sections

- Get Vault Notifications (GET notification-configuration) (p. 156)
- Set Vault Notification Configuration (PUT notification-configuration) (p. 154)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

Archive Operations

Topics

- Upload Archive (POST archive) (p. 161)
- Delete Archive (DELETE archive) (p. 164)

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. 127).

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. 60).

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` 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 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 dashes in it.

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. 120).

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 a Job (POST jobs) (p. 189)), 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
`x-amz-content-sha256`	The SHA256 checksum (a linear hash) of the payload. This is not the same value as you specify in the `x-amz-sha256-tree-hash` header. Type: String Default: None Constraints: None	Yes
`x-amz-sha256-tree-hash`	The user-computed checksum, SHA256 tree hash, of the payload. For information on computing the SHA256 tree hash, see Computing Checksums (p. 127). 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. 123).

Name	Description
Location	The relative URI path of the newly added archive resource. Type: String
x-amz-archive-id	The ID of the archive. This value is also included as part of the `Location` header. Type: String
x-amz-sha256-tree-hash	The checksum of the archive computed by Amazon Glacier. Type: String

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses (p. 139).

Examples

Example Request

The following example shows a request to upload an archive.

```
POST /-/vaults/examplevault/archives HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-sha256-tree-hash: beb0fe31a1c7ca8c6c04d574ea906e3f97b31fd
ca7571defb5b44dca89b5af60
```

```
x-amz-content-sha256:
7f2fe580edb35154041fa3d4b41dd6d3adaef0c85d2ff6309f1d4b520eeecda3
Content-Length: 2097152
x-amz-glacier-version: 2012-06-01
Authorization: Authorization=AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EX
AMPLE/20120525/us-east-1/glacier/aws4_request,SignedHeaders=host;x-amz-content-
sha256;x-amz-date;x-amz-glacier-version,Signa
ture=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, 25 Mar 2012 12:00:00 GMT
x-amz-sha256-tree-hash: beb0fe31a1c7ca8c6c04d574ea906e3f97b31fd
ca7571defb5b44dca89b5af60
Location: /111122223333/vaults/examplevault/archives/NkbByEejwEggmBz2fTHgJrg0XBoD
fjP4q6iu87-TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNG
ntHEQL8MBfGlqrEXAMPLEArchiveId
x-amz-archive-id: NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-TjhqG6eGoOY9Z8i1_AUy
UsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchiveId
```

Related Sections

- Working with Archives in Amazon Glacier (p. 60)
- Uploading Large Archives in Parts (Multipart Upload) (p. 69)
- Delete Archive (DELETE archive) (p. 164)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

Delete Archive (DELETE archive)

Description

This operation deletes an archive from a vault. Subsequent requests to initiate a retrieval of this archive will fail. Archive retrievals that are in progress for this archive ID may or may 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. 77). 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` 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 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 dashes in it.

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. 120).

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. 123).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses (p. 139).

Examples

The following example demonstrates how to delete an archive from the vault named `examplevault`.

Example Request

The ID of the archive to be deleted is specified as a subresource of `archives`.

```
DELETE /-/vaults/examplevault/archives/NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-
TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MB
fGlqrEXAMPLEArchiveId HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 12:00:00 GMT
```

Related Sections

- Initiate Multipart Upload (POST multipart-uploads) (p. 167)
- Upload Archive (POST archive) (p. 161)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

Multipart Upload Operations

Topics

- Initiate Multipart Upload (POST multipart-uploads) (p. 167)
- Upload Part (PUT uploadID) (p. 170)
- Complete Multipart Upload (POST uploadID) (p. 174)
- Abort Multipart Upload (DELETE uploadID) (p. 177)
- List Parts (GET uploadID) (p. 179)
- List Multipart Uploads (GET multipart-uploads) (p. 184)

Initiate Multipart Upload (POST multipart-uploads)

Description

This operation initiates a multipart upload (see Uploading Large Archives in Parts (Multipart Upload) (p. 69)). 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-east-1.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` 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 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 dashes in it.

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. 120).

Name	Description	Required
x-amz-part-size	The size of each part except the last, in bytes. The last part can be smaller than this part size. Type: String Default: None Constraints: The part size must be a 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
x-amz-archive-description	Archive description you are uploading in parts. It can be a plain-language description or some unique identifier you choose to assign. When you retrieve a vault inventory (see Initiate a Job (POST jobs) (p. 189)), 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. 123).

Name	Description
Location	The relative URI path of the multipart upload ID Amazon Glacier created. You use this URI path to scope your requests to upload parts, and to complete the multipart upload. Type: String
x-amz-multipart-upload-id	The ID of the multipart upload. This value is also included as part of the `Location` header. Type: String

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses (p. 139).

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-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-archive-description: MyArchive-101
x-amz-part-size: 4194304
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

Example Response

Amazon Glacier creates a multipart upload resource and adds it to the `multipart-uploads` subresource of the vault. The `Location` response header includes the relative URI path to the multipart upload ID.

```
HTTP/1.1 201 Created
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 12:00:00 GMT
Location: /111122223333/vaults/examplevault/multipart-uploads/OW2fM5iVylEp
FEMM9_HpKowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-khxOjyEXAMPLE
```

```
x-amz-multipart-upload-id: OW2fM5iVylEpFEMM9_HpKowRapC3vn5sSL39_396UW9zL
FUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-khxOjyEXAMPLE
```

For information about uploading individual parts, see Upload Part (PUT uploadID) (p. 170).

Related Sections

- Upload Part (PUT uploadID) (p. 170)
- Complete Multipart Upload (POST uploadID) (p. 174)
- Abort Multipart Upload (DELETE uploadID) (p. 177)
- List Multipart Uploads (GET multipart-uploads) (p. 184)
- List Parts (GET uploadID) (p. 179)
- Delete Archive (DELETE archive) (p. 164)
- Uploading Large Archives in Parts (Multipart Upload) (p. 69)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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. 69).

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. 127).
- **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. 127).
- **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. 167) 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` 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 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 dashes in it.

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. 120).

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

Name	Description	Required
x-amz-content-sha256	The SHA256 checksum (a linear hash) of the uploaded payload. This is not the same value as you specify in the `x-amz-sha256-tree-hash` header. Type: String Default: None Constraints: None	Yes
x-amz-sha256-tree-hash	Specifies a SHA256 tree hash of the data being uploaded. For information about computing a SHA256 tree hash, see Computing Checksums (p. 127). 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. 123).

Name	Description
x-amz-sha256-tree-hash	The SHA256 tree hash that Amazon Glacier computed for the uploaded part. Type: String

Response Body

This operation does not return a response body.

Example

The following request uploads a 4 MB part. The request sets the byte range to make this the first part in the archive.

Example Request

The example sends an HTTP PUT request to upload a 4 MB part. The request is sent to the URI of the Upload ID that was returned by the Initiate Multipart Upload request. The `Content-Range` header identifies the part as the first 4 MB data part of the archive.

```
PUT /-/vaults/examplevault/multipart-uploads/OW2fM5iVylEpFEMM9_Hp
KowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
Date: Sun, 25 Mar 2012 12:00:00 GMT
Content-Range:bytes 0-4194303/*
x-amz-sha256-tree-hash:c06f7cd4baacb087002a99a5f48bf953
x-amz-content
sha256:726e392cb4d09924dbad1cc0ba3b00c3643d03d14cb4b823e2f041cff612a628
Content-Length: 4194304
Authorization: Authorization=AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EX
AMPLE/20120525/us-east-1/glacier/aws4_request,SignedHeaders=host;x-amz-content-
sha256;x-amz-date;x-amz-glacier-version,Signa
ture=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_Hp
KowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
Date: Sun, 25 Mar 2012 12:00:00 GMT
Content-Range:bytes 4194304-8388607/*
Content-Length: 4194304
x-amz-sha256-tree-hash:f10e02544d651e2c3ce90a4307427493
x-amz-content
sha256:726e392cb4d09924dbad1cc0ba3b00c3643d03d14cb4b823e2f041cff612a628
x-amz-glacier-version: 2012-06-01
Authorization: Authorization=AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EX
AMPLE/20120525/us-east-1/glacier/aws4_request, SignedHeaders=host;x-amz-content-
sha256;x-amz-date;x-amz-glacier-version, Signa
ture=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, 25 Mar 2012 12:00:00 GMT
```

Related Sections

- Initiate Multipart Upload (POST multipart-uploads) (p. 167)
- Upload Part (PUT uploadID) (p. 170)

- Complete Multipart Upload (POST uploadID) (p. 174)
- Abort Multipart Upload (DELETE uploadID) (p. 177)
- List Multipart Uploads (GET multipart-uploads) (p. 184)
- List Parts (GET uploadID) (p. 179)
- Uploading Large Archives in Parts (Multipart Upload) (p. 69)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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. 69).

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. 127). 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. 179) 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` 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 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 dashes in it.

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. 120).

Name	Description	Required
x-amz-archive-size	The total size, in bytes, of the entire archive. This value should be the sum of all the sizes of the individual parts that you uploaded. Type: String Default: None Constraints: None	Yes
x-amz-sha256-tree-hash	The SHA256 tree hash of the entire archive. It is the tree hash of SHA256 tree hash of the individual parts. If the value you specify in the request does not match the SHA256 tree hash of the final assembled archive as computed by Amazon Glacier, Amazon Glacier returns an error and the request fails. 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. 123).

Name	Description
Location	The relative URI path of the newly created archive. This URL includes the archive ID that is generated by Amazon Glacier. Type: String
x-amz-archive-id	The ID of the archive. This value is also included as part of the `Location` header. Type: String

Response Fields

This operation does not return a response body.

Example

Example Request

In this example, an HTTP POST request is sent to the URI that was returned by an Initiate Multipart Upload request. The request specifies both the SHA256 tree hash of the entire archive and the total archive size.

```
POST /-/vaults/examplevault/multipart-uploads/OW2fM5iVylEpFEMM9_Hp
KowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
z-amz-Date: 20120325T120000Z
x-amz-sha256-tree-hash:1ffc0f54dd5fdd66b62da70d25edacd0
x-amz-archive-size:8388608
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 12:00:00 GMT
```

```
Location: /111122223333/vaults/examplevault/archives/NkbByEejwEggmBz2fTHgJrg0XBoD
fjP4q6iu87-TjhqG6eGoOY9Z8i1_AUyUsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNG
ntHEQL8MBfGlqrEXAMPLEArchiveId
x-amz-archive-id: NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-TjhqG6eGoOY9Z8i1_AUy
UsuhPAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchiveId
```

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. 167)
- Upload Part (PUT uploadID) (p. 170)
- Abort Multipart Upload (DELETE uploadID) (p. 177)
- List Multipart Uploads (GET multipart-uploads) (p. 184)
- List Parts (GET uploadID) (p. 179)
- Uploading Large Archives in Parts (Multipart Upload) (p. 69)
- Delete Archive (DELETE archive) (p. 164)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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. 69).

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` 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 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 dashes in it.

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. 120).

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. 123).

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses (p. 139).

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_Hp
KowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-khxOjyEXAMPLE  HT
TP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

Example Response

```
HTTP/1.1 204 No Content
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 12:00:00 GMT
```

Related Sections

- Initiate Multipart Upload (POST multipart-uploads) (p. 167)
- Upload Part (PUT uploadID) (p. 170)
- Complete Multipart Upload (POST uploadID) (p. 174)
- List Multipart Uploads (GET multipart-uploads) (p. 184)
- List Parts (GET uploadID) (p. 179)
- Uploading Large Archives in Parts (Multipart Upload) (p. 69)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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. 69).

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. By default, this operation returns up to 1,000 uploaded parts 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 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. 167)). 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` 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 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 dashes in it.

Request Parameters

Name	Description	Required
`limit`	Specifies the maximum number of parts returned in the response body. If not specified, the List Parts operation returns up to 1,000 uploads. Type: String Constraints: Minimum integer value of `1`. Maximum integer value of `1000`.	No
`marker`	An opaque string used for pagination. `marker` specifies the part at which the listing of parts should begin. Get the `marker` value from the response of a previous List Parts response. You need only include the `marker` if you are continuing the pagination of results started in a previous List Parts request. Type: String Constraints: None	No

Request Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers (p. 123).

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,
    "CompletionDate" : 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. 123).

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

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

PartSizeInBytes
The part size in bytes. This is the same value that you specified in the Initiate Multipart Upload request.

Type: Number

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

MultipartUploadId
The ID of the upload to which the parts are associated.

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. 139).

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_Hp
KowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-khxOjyEXAMPLE HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 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_396UW9zL
FUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-khxOjyEXAMPLE",
    "PartSizeInBytes" : 4194304,
    "Parts" :
    [ {
      "RangeInBytes" : "0-4194303",
      "SHA256TreeHash" : "01d34dabf7be316472c93b1ef80721f5d4"
      },
      {
      "RangeInBytes" : "4194304-8388607",
      "SHA256TreeHash" : "0195875365afda349fc21c84c099987164"
      }],
    "VaultARN" : "arn:aws:glacier:us-east-1:012345678901: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_Hp
KowRapC3vn5sSL39_396UW9zLFUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-khxOjyEXAMPLE?mark
er=1001&limit=1 HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 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_396UW9zL
FUWVrnRHaPjUJddQ5OxSHVXjYtrN47NBZ-khxOjyEXAMPLE",
    "PartSizeInBytes" : 4194304,
    "Parts" :
    [ {
      "RangeInBytes" : "4194304-8388607",
      "SHA256TreeHash" : "01d34dabf7be316472c93b1ef80721f5d4"
      }],
    "VaultARN" : "arn:aws:glacier:us-east-1:012345678901:vaults/demo1-vault"
}
```

Related Sections

- Initiate Multipart Upload (POST multipart-uploads) (p. 167)
- Upload Part (PUT uploadID) (p. 170)

- Complete Multipart Upload (POST uploadID) (p. 174)
- Abort Multipart Upload (DELETE uploadID) (p. 177)
- List Multipart Uploads (GET multipart-uploads) (p. 184)
- Uploading Large Archives in Parts (Multipart Upload) (p. 69)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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. 167) 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. 179)) 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. 69).

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` 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 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 dashes in it.

Request Parameters

Name	Description	Required
limit	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
marker	An opaque string used for pagination. `marker` specifies the upload at which the listing of uploads should begin. Get the `marker` value from a previous List Uploads response. You need only include the `marker` if you are continuing the pagination of results started in a previous List Uploads request. Type: String Constraints: None	No

Request Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers (p. 123).

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. 123).

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. 167) 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. 139).

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-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 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": "xsQdFIRsfJr20CW2AbZBKpRZAFTZ
SJIMtL2hYf8mvp8dM0m4RUzlaqoEye6g3h3ecqB_zqwB7zLDMeSWhwo65re4C4Ev",
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-east-1:012345678901:vaults/examplevault"

    },
    {
      "ArchiveDescription": "archive 2",
      "CreationDate": "2012-04-01T15:00:00.000Z",
      "MultipartUploadId": "nPyGOnyFcx67qqX7E-0tSGiRi88hHMOwOxR-
_jNyM6RjVMFfV29lFqZ3rNsSaWBugg6OP92pRtufeHdQH7ClIpSF6uJc",
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-east-1:012345678901:vaults/examplevault"

    },
    {
      "ArchiveDescription": "archive 3",
      "CreationDate": "2012-03-20T17:03:43.221Z",
      "MultipartUploadId": "qt-RBst_7yO8gVIonIBsAxr2t-db0pE4s8MNeGjKjGdNpuU-
cdSAcqG62guwV9r5jh5mLyFPzFEitTpNE7iQfHiu1XoV",
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-east-1:012345678901: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=xsQdFIRsfJr20CW2AbZB
KpRZAFTZSJIMtL2hYf8mvp8dM0m4RUzlaqoEye6g3h3ecqB_zqwB7zLDMeSWhwo65re4C4Ev HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 12:00:00 GMT
Content-Type: application/json
Content-Length: 470

{
  "Marker": "qt-RBst_7yO8gVIonIBsAxr2t-db0pE4s8MNeGjKjGdNpuU-cd
SAcqG62guwV9r5jh5mLyFPzFEitTpNE7iQfHiu1XoV",
  "UploadsList" : [
    {
      "ArchiveDescription": "archive 2",
      "CreationDate": "2012-04-01T15:00:00.000Z",
      "MultipartUploadId": "nPyGOnyFcx67qqX7E-0tSGiRi88hHMOwOxR-
_jNyM6RjVMFfV29lFqZ3rNsSaWBugg6OP92pRtufeHdQH7ClIpSF6uJc",
      "PartSizeInBytes": 4194304,
      "VaultARN": "arn:aws:glacier:us-east-1:012345678901:vaults/examplevault"

    }
  ]
}
```

Related Sections

- Initiate Multipart Upload (POST multipart-uploads) (p. 167)
- Upload Part (PUT uploadID) (p. 170)
- Complete Multipart Upload (POST uploadID) (p. 174)
- Abort Multipart Upload (DELETE uploadID) (p. 177)
- List Parts (GET uploadID) (p. 179)
- Uploading Large Archives in Parts (Multipart Upload) (p. 69)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

Job Operations

Topics

- Initiate a Job (POST jobs) (p. 189)
- Describe Job (GET JobID) (p. 196)
- Get Job Output (GET output) (p. 203)
- List Jobs (GET jobs) (p. 210)

Initiate a Job (POST jobs)

Description

This operation initiates a job of the specified type, which can be an archive retrieval or vault inventory retrieval. 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.
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. 203) 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. 154).
- **Get job details**— You can make a Describe Job (GET JobID) (p. 196) 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. 196).

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. 154).

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. 206).

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. 196) 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. 203)), the range must be tree-hash aligned. For more information about tree-hash aligned ranges, see Receiving Checksums When Downloading Data (p. 137).

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
}
```

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` 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 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 dashes in it.

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

Specifies 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 *EndByte-Value* 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`.

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
Location	The relative URI path of the job. You can use this URI path to find the job status. For more information, see Describe Job (GET JobID) (p. 196). Type: String Default: None

Header	Description
x-amz-job-id	The ID of the job. This value is also included as part of the `Location` header. Type: String Default: None

Response Body

This operation does not return a response body.

Errors

For information about Amazon Glacier exceptions and error messages, see Error Responses (p. 139).

Examples

Example Request: Initiate an archive retrieval job

```
POST /-/vaults/examplevault/jobs HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
  "Type": "archive-retrieval",
  "ArchiveId": "NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-TjhqG6eGoOY9Z8i1_AUyUsuh
PAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchiveId"
  "Description": "My archive description",
  "SNSTopic": "arn:aws:sns:us-east-1:111111111111:Glacier-ArchiveRetrieval-
topic-Example"
}
```

The following is an example of the body of a request that specifies a range of the archive to retrieve using the `RetrievalByteRange` field.

```
{
  "Type": "archive-retrieval",
  "ArchiveId": "NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-TjhqG6eGoOY9Z8i1_AUyUsuh
PAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchiveId"
  "Description": "My archive description",
  "RetrievalByteRange": "2097152-4194303",
  "SNSTopic": "arn:aws:sns:us-east-1:111111111111:Glacier-ArchiveRetrieval-
topic-Example"
}
```

Example Response

```
HTTP/1.1 202 Accepted
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 12:00:00 GMT
```

```
Location: /111122223333/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
x-amz-job-id: HkF9p6o7yjhFx-K3CGl6fuSm6VzW9T7esGQfco8nUXVY
wS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

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-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
Content-Type: application/x-www-form-urlencoded
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2

{
  "Type": "inventory-retrieval",
  "Description": "My inventory job",
  "Format": "CSV",
  "SNSTopic": "arn:aws:sns:us-east-1:111111111111:Glacier-InventoryRetrieval-
topic-Example"
}
```

Example Response

```
HTTP/1.1 202 Accepted
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 12:00:00 GMT
Location: /111122223333/vaults/examplevault/jobs/HkF9p6o7yjhFx-
K3CGl6fuSm6VzW9T7esGQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
x-amz-job-id: HkF9p6o7yjhFx-K3CGl6fuSm6VzW9T7esGQfco8nUXVY
wS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID
```

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. 196).

```
{
    "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": "vyS0t2jHQe5qbcDggIeD50chS1SXwYMrkVKo0KHiTUjEYxBGC
qRLKaiySzdN7QXGVVV5XZpNVG67pCZ_uykQXFMLaxOSu2hO_-5C0AtWMDrfo7LgVOyfnveDRuOSec
Uo3Ueq7K0"
    },
}
```

Related Sections

- Describe Job (GET JobID) (p. 196)
- Get Job Output (GET output) (p. 203)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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 a Job (POST jobs) (p. 189).

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` 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 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 dashes in it.

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. 210).

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. 120).

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,
   "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. 123).

Response Body

The response body contains the following JSON fields.

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

Type: String

ArchiveId
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. 203).

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 a Job (POST jobs) (p. 189) 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 a Job (POST jobs) (p. 189) 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. 190).

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 *StartByte-Value* 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 a Job (POST jobs) (p. 189) 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. 137).

For the specific case when the whole archive is retrieved, this value is the same as the `Archive-SHA256TreeHash` 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

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. 139).

Examples

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

Example Request: Get job description

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-K3CGl6fuSm6VzW9T7esGQfco8nUXVY
wS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID HTTP/1.1
```

```
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 12:00:00 GMT
Content-Type: application/json
Content-Length: 419
{
  "Action": "ArchiveRetrieval",
  "ArchiveId": "NkbByEejwEggmBz2fTHgJrg0XBoDfjP4q6iu87-TjhqG6eGoOY9Z8i1_AUyUsuh
PAdTqLHy8pTl5nfCFJmDl2yEZONi5L26Omw12vcs01MNGntHEQL8MBfGlqrEXAMPLEArchiveId",
  "ArchiveSizeInBytes": 16777216,
  "ArchiveSHA256TreeHash": "beb0fe31a1c7ca8c6c04d574ea906e3f97b31fd
ca7571defb5b44dca89b5af60",
  "Completed": false,
  "CreationDate": "2012-05-15T17:21:39.339Z",
  "CompletionDate": "2012-05-15T17:21:43.561Z",
  "InventorySizeInBytes": null,
  "JobDescription": "My ArchiveRetrieval Job",
  "JobId": "HkF9p6o7yjhFx-K3CGl6fuSm6VzW9T7esGQfco8nUXVY
wS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID",
  "RetrievalByteRange": "0-16777215",
  "SHA256TreeHash": "beb0fe31a1c7ca8c6c04d574ea906e3f97b31fd
ca7571defb5b44dca89b5af60",
  "SNSTopic": "arn:aws:sns:us-east-1:012345678901:mytopic",
  "StatusCode": "InProgress",
  "StatusMessage": "Operation in progress.",
  "VaultARN": "arn:aws:glacier:us-east-1:012345678901: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-K3CGl6fuSm6VzW9T7esGQfco8nUXVY
wS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID",
   "RetrievalByteRange": null,
   "SHA256TreeHash": null,
   "SNSTopic": "arn:aws:sns:us-east-1:012345678901:mytopic",
   "StatusCode": "InProgress",
   "StatusMessage": "Operation in progress.",
   "VaultARN": "arn:aws:glacier:us-east-1:012345678901: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-mXi2u3lc49uW6TcE
hDF2D9pB2phx-BN30JaBru7PMyOlfXHdStzu8",
    "NextInventoryRetrievalMarker": null,
    "RetrievalByteRange": null,
    "SHA256TreeHash": null,
    "SNSTopic": null,
    "StatusCode": "Succeeded",
    "StatusMessage": "Succeeded",
    "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": "vyS0t2jHQe5qbcDggIeD50chS1SXwYMrkVKo0KHiTUjEYxBGC
qRLKaiySzdN7QXGVVV5XZpNVG67pCZ_uykQXFMLaxOSu2hO_-5C0AtWMDrfo7LgVOyfnveDRuOSec
Uo3Ueq7K0"
    },
}
```

Related Sections

- Get Job Output (GET output) (p. 203)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

Get Job Output (GET output)

Description

This operation downloads the output of the job you initiated using Initiate a Job (POST jobs) (p. 189). 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. 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. You can compute the checksum on the client and verify that the values match to ensure the portion you downloaded is the correct data.

A job ID will 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` 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 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 dashes in it.

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. 120).

Name	Description	Required
Range	The range of bytes to retrieve from the output. For example, if you want to download the first 1,048,576 bytes, specify `Range: bytes=0-1048575`. 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. If the job output is large, then you can use the `Range` request header to retrieve a portion of the output. This allows you to download the entire output in smaller chunks of bytes. For example, suppose you have 1 GB job output you want to download and you decide to download 128 MB chunks of data at a time, a total of eight Get Job Output requests. You will use the following process to download the job output: 1. Download 128 MB chunk of output by specifying appropriate byte range using the `Range` header. 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 API, obtain job information of the job that provided you the output. The response includes the checksum of the entire archive stored in Amazon Glacier. You compare this value with the checksum you computed to ensure you have downloaded the entire archive content with no errors. Type: String Default: None Constraints: None	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
`Content-Range`	The range of bytes returned by Amazon Glacier. If only partial output is downloaded, the response provides the range of bytes Amazon Glacier returned. For example, `bytes 0-1048575/8388608` returns the first 1 MB from 8 MB. For more information about the `Content-Range` header, go to Content-Range Header Field Definition. Type: String
`Content-Type`	The Content-Type depends on whether the job output is an archive or a vault inventory. • For archive data, the Content-Type is `application/octet-stream`. • For vault inventory, if you requested CSV format when you initiated the job, the Content-Type is `text/csv`. Otherwise, by default, vault inventory is returned as JSON, and the Content-Type is `application/json`. Type: String

Header	Description
x-amz-sha256-tree-hash	The checksum of the data in the response. This header is returned only when retrieving the output for an archive retrieval job. Furthermore, this header appears when the retrieved data range requested in the Initiate Job request is tree hash aligned and the range to download in the Get Job Output is also tree hash aligned. For more information about tree hash aligned ranges, see Receiving Checksums When Downloading Data (p. 137). For example, if in your Initiate Job request you specified a tree hash aligned range to retrieve (which includes the whole archive), then you will receive the checksum of the data you download under the following conditions: • You get the entire range of the retrieved data. • You request a byte range of the retrieved data that has a size of a megabyte (1024 KB) multiplied by a power of 2 and that starts and ends on a multiple of the size of the requested range. For example, if you have 3.1 MB of retrieved data and you specify a range to return that starts at 1 MB and ends at 2 MB, then the `x-amz-sha256-tree-hash` is returned as a response header. • You request a range to return of the retrieved data that goes to the end of the data, and the start of the range is a multiple of the size of the range to retrieve rounded up to the next power of two but not smaller than one megabyte (1024 KB). For example, if you have 3.1 MB of retrieved data and you specify a range that starts at 2 MB and ends at 3.1 MB (the end of the data), then the `x-amz-sha256-tree-hash` is returned as a response header. Type: String

Response Body

Amazon Glacier returns the job output in the response body. Depending on the job type, the output can be the archive contents or the vault inventory. In case of a vault inventory, by default the inventory list is returned as the following JSON body.

```
{
 "VaultARN": String,
 "InventoryDate": String,
 "ArchiveList": [
      {"ArchiveId": String,
       "ArchiveDescription": String,
       "CreationDate": String,
       "Size": Number,
       "SHA256TreeHash": String
      },
      ...
    ]
}
```

If you requested the comma-separated values (CSV) output format when you initiated the vault inventory job, then the vault inventory is returned in CSV format in the body. The CSV format has five columns "ArchiveId", "ArchiveDescription", "CreationDate", "Size", and "SHA256TreeHash" with the same definitions as the corresponding JSON fields.

Note

In the returned CSV format, fields may be returned with the whole field enclosed in double-quotes. Fields that contain a comma or double-quotes are always returned enclosed in double-quotes. For example, `my archive description,1` is returned as `"my archive description,1"`. Double-quote characters that are within returned double-quote enclosed fields are *escaped* by preceding them with a backslash character. For example, `my archive description,1"2` is returned as `"my archive description,1\"2"` and `my archive description,1\"2` is returned as `"my archive description,1\\"2"`. The backslash character is not escaped.

The JSON response body contains the following JSON fields.

ArchiveDescription
The description of an archive.

Type: String

ArchiveId
The ID of an archive.

Type: String

ArchiveList
An array of archive metadata. Each object in the array represents metadata for one archive contained in the vault.

Type: Array

CreationDate
The UTC date and time the archive was created.

Type: A string representation 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. 139).

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-K3CGl6fuSm6VzW9T7esGQfco8nUXVY
wS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID/output HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

Example Response

The following is an example response of an archive retrieval job. Note that the `Content-Type` header is `application/octet-stream` and that `x-amz-sha256-tree-hash` header is included in the response, which means that all the job data is returned.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
x-amz-sha256-tree-hash: beb0fe31a1c7ca8c6c04d574ea906e3f97b31fd
ca7571defb5b44dca89b5af60
Date: Sun, 25 Mar 2012 12:00:00 GMT
Content-Type: application/octet-stream
Content-Length: 1048576

[Archive data.]
```

The following is an example response of an inventory retrieval job. Note that the `Content-Type` header is `application/json`. Also note that the response does not include the `x-amz-sha256-tree-hash` header.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 12:00:00 GMT
Content-Type: application/json
Content-Length: 906

{
 "VaultARN": "arn:aws:glacier:us-east-1:012345678901:vaults/examplevault",
 "InventoryDate": "2011-12-12T14:19:01Z",
 "ArchiveList": [
   {
     "ArchiveId": "DMTmICA2n5Tdqq5BV2z7og-A20xnpAPKt3UXwWxdWsn_D6au
TUrW6kwy5Qyj9xd1MCE1mBYvMQ63LWaT8yTMzMaCxB_9VBWrW4Jw4zsvg5kehAPDVKcp
pUD1X7b24JukOr4mMAq-oA",
     "ArchiveDescription": "my archive1",
     "CreationDate": "2012-05-15T17:19:46.700Z",
     "Size": 2140123,
     "SHA256TreeHash": "6b9d4cf8697bd3af6aa1b590a0b27b337da5b18988db
cc619a3e608a554a1e62"
   },
   {
    "ArchiveId": "2lHzwhKhgF2JHyvCS-ZRuF08IQLuyB4265Hs3AXj9MoAIhz7tbXAvcFeHus
gU_hViO1WeCBe0N5lsYYHRyZ7rrmRkNRuYrXUs_sjl2K8ume_7mKO_0i7C-uHE1oHqaW9d37pabXrSA",

     "ArchiveDescription": "my archive2",
     "CreationDate": "2012-05-15T17:21:39.339Z",
     "Size": 2140123,
     "SHA256TreeHash":
"7f2fe580edb35154041fa3d4b41dd6d3adaef0c85d2ff6309f1d4b520eeecda3"
   }
  ]
}
```

Example 2: Download only partial output

This example retrieves only a portion of the archive prepared by Amazon Glacier in response to your initiate archive retrieval job request. The request uses the optional `Range` header to retrieve only the first 1,024 bytes.

Example Request

```
GET /-/vaults/examplevault/jobs/HkF9p6o7yjhFx-K3CGl6fuSm6VzW9T7esGQfco8nUXVY
wS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID/output HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
Range: bytes=0-1023
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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, 25 Mar 2012 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. 196)
- Initiate a Job (POST jobs) (p. 189)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

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, and 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. By default, this operation returns up to 1,000 jobs in the response. You should always check the response `marker` field for a marker at which to continue the list; if there are no more items the `marker` field is `null`. To return a list of jobs that begins at a specific job, set the `marker` request parameter to the value you obtained from a previous List Jobs request. You can also limit the number of jobs returned in the response by specifying the `limit` parameter in the request.

Additionally, you can filter the jobs list returned by specifying an optional `statuscode` (`InProgress`, `Succeeded`, or `Failed`) and `completed` (`true`, `false`) parameter. The `statuscode` allows you to specify that only jobs that match a specified status are returned. The `completed` parameter allows you to specify that only jobs in specific completion state are returned.

Requests

Syntax

To returns 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` 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 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 dashes in it.

Request Parameters

Name	Description	Required
completed	Specifies the state of the jobs to return. You can specify `true` or `false`. Type: Boolean Constraints: None	No
limit	Specifies that the response be limited to the specified number of items or fewer. If not specified, the List Jobs operation returns up to 1000 jobs. Type: String Constraints: Minimum integer value of 1. Maximum integer value of 1000.	No
marker	An opaque string used for pagination. `marker` specifies the job at which the listing of jobs should begin. Get the `marker` value from a previous List Jobs response. You need only include the `marker` if you are continuing the pagination of results started in a previous List Jobs request. Type: String Constraints: None	No
statuscode	Specifies the type of job status to return. Type: String Constraints: One of the values `InProgress`, `Succeeded`, or `Failed`.	No

Request Headers

This operation uses only response headers that are common to most responses. For information about common response headers, see Common Response Headers (p. 123).

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,
      "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. 123).

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 UTC time the job completed.

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

CreationDate
The UTC time 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 a Job (POST jobs) (p. 189) 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 the a Initiate a Job (POST jobs) (p. 189) 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. 190).

Type: String

Marker

An opaque string that represents where to continue pagination of the results. You use the `marker` in a new List Jobs request to obtain more jobs in the list. If there are no more jobs, 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 a Job (POST jobs) (p. 189) 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. 137).

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 a Job (POST jobs) (p. 189)).

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

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. 139).

Examples

The following examples demonstrate how to use return information about vault jobs. The first example returns up to 1,000 jobs, and the second example returns a subset of jobs.

Example: Return All Jobs

Example Request

The following GET request returns up to 1,000 jobs for a vault.

```
GET /-/vaults/examplevault/jobs  HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,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.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 12:00:00 GMT
Content-Type: application/json
Content-Length: 1444

{
  "JobList": [
    {
      "Action": "ArchiveRetrieval",
      "ArchiveId": "BDfaUQul0dVzYwAMr8YSa_6_8abbhZq-i1oT69g8ByClfJyBgAGB
kWl2QbF5os851P7Y7KdZDOHWJIn4rh1ZHaOYD3MgFhK_g0oDPesW34uHQoVGwoIqubf6BgUE
fQm_wrU4Jlm3cA",
      "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-Jr
Gtu1O_YFKLnzQ64548qJc667BRWTwBLZC76Ygy1jHYruqXkdcAhRsh0hYv4eVRU",
      "RetrievalByteRange": "0-1048575",
      "SHA256TreeHash":
"25499381569ab2f85e1fd0eb93c5406a178ab77c5933056eb5d6e7d4adda609b",
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
      "VaultARN": "arn:aws:glacier:us-east-1:012345678901: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": "2cvVOnBL36btzyP3pobwIceiaJebM1bx9vZOOUtmNAr0KaVZ4Wk
WgVjiPldJ73VU7imlm0pnZriBVBebnqaAcirZq_C5",
      "RetrievalByteRange": null,
      "SHA256TreeHash": null,
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
      "VaultARN": "arn:aws:glacier:us-east-1:012345678901:vaults/examplevault"

      "InventoryRetrievalParameters": {
          "StartDate": "2013-11-12T13:43:12Z",
          "EndDate": "2013-11-20T08:12:45Z",
          "Limit": "120000",
          "Format": "JSON",
          "Marker": "vyS0t2jHQe5qbcDggIeD50chS1SXwYMrkVKo0KHiTUjEYxBGC
qRLKaiySzdN7QXGVVV5XZpNVG67pCZ_uykQXFMLaxOSu2hO_-5C0AtWMDrfo7LgVOyfnveDRuOSec
Uo3Ueq7K0"
    }
  ],
  "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-K3CGl6fuSm6VzW9T7es
GQfco8nUXVYwS0jlb5gq1JZ55yHgt5vP54ZShjoQzQVVh7vEXAMPLEjobID&limit=1  HTTP/1.1
Host: glacier.us-east-1.amazonaws.com
```

```
x-amz-Date: 20120325T120000Z
x-amz-glacier-version: 2012-06-01
Authorization: AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20120525/us-
east-1/glacier/aws4_request,SignedHeaders=host;x-amz-date;x-amz-glacier-ver
sion,Signature=9257c16da6b25a715ce900a5b45b03da0447acf430195dcb540091b12966f2a2
```

Example Response

The following response shows one job returned and the `marker` field returns the next job in the list.

```
HTTP/1.1 200 OK
x-amzn-RequestId: AAABZpJrTyioDC_HsOmHae8EZp_uBSJr6cnGOLKp_XJCl-Q
Date: Sun, 25 Mar 2012 12:00:00 GMT
Content-Type: application/json
Content-Length: 1744

{
  "JobList": [
    {
      "Action": "ArchiveRetrieval",
      "ArchiveId": "58-3KpZfcMPUznvMZNPaKyJx9wODCsWTnqcjtx2CjKZ6b-
XgxEuA8yvZOYTPQfd7gWR4GRm2XR08gcnWbLV4VPV_kDWtZJKi0TFhKKVPzwrZnA4-FXuIBfViY
UIVveeiBE51FO4bvg",
      "ArchiveSizeInBytes": 8388608,
      "ArchiveSHA256TreeHash":
"106086b256ddf0fedf3d9e72f461d5983a2566247ebe7e1949246bc61359b4f4",
      "Completed": true,
      "CompletionDate": "2012-05-01T00:25:20.043Z",
      "CreationDate": "2012-05-01T00:25:16.344Z",
      "InventorySizeInBytes": null,
      "JobDescription": "aaabbbccc",
      "JobId": "s4MvaNHIh6mOa1f8iY4ioG2921SDPihXxh3Kv0FBX-JbNPctpRvE4c2_Bifuh
dGLqEhGBNGeB6Ub-JMunR9JoVa8y1hQ",
      "RetrievalByteRange": "0-8388607",
      "SHA256TreeHash":
"106086b256ddf0fedf3d9e72f461d5983a2566247ebe7e1949246bc61359b4f4",
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
      "VaultARN": "arn:aws:glacier:us-east-1:012345678901:vaults/examplevault"

    },
    {
      "Action": "ArchiveRetrieval",
      "ArchiveId": "2NVGpf83U6qB9M2u-Ihh61yoFLRDEoh7YLZWKBn80A2i1xG8uieBwG
jAr4RkzOHA0E07ZjtI267R03Z-6Hxd8pyGQkBdciCSH1-Lw63Kx9qKpZbPCdU0uTW_WAdwF6lR6w8iSyK
dvw",
      "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_tf6fOR4jrJCL61Mfk6VM03oY8lmnWK93KK4gLig1UPAbZ
iN3UV4G_5nq4AfmJHQ_dOMLOX5k8ItFv0wCPN0oaz5dG",
      "RetrievalByteRange": "0-1048575",
      "SHA256TreeHash": "3d2ae052b2978727e0c51c0a5e32961c6a56650d1f2e4cec
cab6472a5ed4a0",
      "SNSTopic": null,
      "StatusCode": "Succeeded",
      "StatusMessage": "Succeeded",
      "VaultARN": "arn:aws:glacier:us-east-1:012345678901:vaults/examplevault"

    }
  ],
  "Marker": "CQ_tf6fOR4jrJCL61Mfk6VM03oY8lmnWK93KK4gLig1UPAbZ
iN3UV4G_5nq4AfmJHQ_dOMLOX5k8ItFv0wCPN0oaz5dG"
}
```

Related Sections

- Describe Job (GET JobID) (p. 196)
- Access Control Using AWS Identity and Access Management (IAM) (p. 116)

Document History

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

API version: 2012-06-01

Latest documentation update: June 27, 2014

Change	Description	Release Date
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. 190) in the Initiate a Job (POST jobs) (p. 189) topic.	December 31, 2013
Removed outdated URLs	Removed the URLs that pointed to the old security credentials page from code examples.	July 26, 2013
Support for range retrievals	Amazon Glacier now supports retrieval of specific ranges of your archives. You can initiate a job requesting Amazon Glacier to prepare an entire archive or a portion of the archive for subsequent download. When an archive is very large, you may find it cost effective to initiate several sequential jobs to prepare your archive. For more information, see Downloading an Archive in Amazon Glacier (p. 77). For pricing information, go to the Amazon Glacier detail page.	November 13, 2012
New Guide	This is the first release of the *Amazon Glacier Developer Guide*.	August 20, 2012

AWS Glossary

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

Numbers and Symbols

100-continue	A method that enables a client to see if a server can accept a request before actually sending it. For large PUTs, this method can save both time and bandwidth charges.

A

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

access control list	A document that defines who can access a particular bucket or object. Each bucket and object in Amazon S3 has an ACL. The document defines what each type of user can do, such as write and read permissions.
access identifiers	See credentials.
access key ID	A string that uniquely identifies your AWS account as well as individual IAM users in your account. It's used in conjunction with the secret access key (p. 245) to cryptographically sign programmatic AWS requests.
access key rotation	A method to increase security by changing the AWS access key ID. This method enables you to retire an old key at your discretion.
access policy language	A language for writing documents (that is, *policies*) that specify who can access a particular AWS resource and under what conditions.
account	The AWS account associated with a particular AWS login ID and password. IAM: The AWS account that centrally controls all the resources created under its umbrella and pays for all AWS activity for those resources. The AWS account has permission to do anything and everything with all the AWS account resources. This is in contrast to the user (p. 249).
account activity	A web page showing your month-to-date AWS usage and costs. The account activity page is located at http://aws.amazon.com/account-activity/.

action	An API function. Also called *operation* or *call*. The activity the principal (p. 240) has permission to perform. The action is B in the statement "A has permission to do B to C where D applies." For example, Jane sends a request to Amazon SQS with Action=ReceiveMessage. Amazon CloudWatch: The response initiated by the change in an alarm's state: for example, from OK to ALARM. The state change may be triggered by a metric reaching the alarm threshold, or by a SetAlarmState request. Each alarm can have one or more actions assigned to each state. Actions are performed once each time the alarm changes to a state that has an action assigned, such as an Amazon Simple Notification Service notification, an Auto Scaling policy execution or an Amazon EC2 instance stop/terminate action.
activate URL	The location where your customers can generate a new activation key (p. 221) for your product, should you request it. The activate URL is http://www.amazon.com/dp-activate.
activation	The process your product goes through to prepare itself for the customer's use. During activation, your product obtains the required credentials for the customer.
activation key	An encoded string that represents the relationship between a customer and a Amazon DevPay product the customer has purchased. AWS generates this value when the customer completes the purchase of the product. You use the key to obtain credentials related to the customer and product.
active authorization	When you inform customers of a price change for your product, they have to take action to accept the price change. If they don't take the action to accept the price change, their access to your product is canceled when the price change takes effect.
active trusted signers	A list showing each of the trusted signers you've specified and the IDs of the corresponding active key pairs that CloudFront is aware of. To be able to create working signed URLs, a trusted signer must appear in this list with at least one key pair ID.
administrative suspension	Auto Scaling might suspend processes for Auto Scaling group (p. 224) that repeatedly fail to launch instances. Auto Scaling groups that most commonly experience administrative suspension have zero running instances, have been trying to launch instances for more than 24 hours, and have not succeeded in that time.
alarm	An item that watches a single metric over a specified time period, and triggers an Amazon SNS topic (p. 249) or an Auto Scaling policy (p. 239) if the value of the metric crosses a threshold value over a predetermined number of time periods.
allow	An allow results from a statement that has effect=allow, assuming any stated conditions are met. Example: Allow requests received before 1:00 p.m. on April 30, 2010. An allow overrides any default deny (p. 228), but never an explicit deny (p. 231).
Amazon CloudFront	An AWS content delivery service that helps you improve the performance, reliability, and availability of your websites and applications. See Also http://aws.amazon.com/cloudfront.
Amazon CloudSearch	A fully-managed service in the AWS cloud that makes it easy to set up, manage, and scale a search solution for your website or application.
Amazon CloudWatch	A web service that enables you to monitor and manage various metrics, and configure alarm actions based on data from those metrics. See Also http://aws.amazon.com/cloudwatch.

Amazon DevPay	An easy-to-use online billing and account management service that makes it easy for you to sell an Amazon EC2 AMI or an application built on Amazon S3. See Also http://aws.amazon.com/devpay.
Amazon Elastic Block Store	A service that provides block level storage volumes for use with EC2 instances. See Also http://aws.amazon.com/ebs.
Amazon EBS-backed AMI	Instances launched from this type of AMI use an Amazon EBS volume as their root device. Compare this with instances launched from instance store-backed AMIs, which use the instance store as the root device.
Amazon Elastic Compute Cloud	A web service that enables you to launch and manage Linux/UNIX and Windows server instances in Amazon's data centers. See Also http://aws.amazon.com/ec2.
Amazon EC2 VM Import Connector	See http://aws.amazon.com/ec2/vm-import.
Amazon Elastic MapReduce	A web service that makes it easy to process large amounts of data efficiently. Amazon EMR uses Hadoop processing combined with several AWS products to do such tasks as web indexing, data mining, log file analysis, machine learning, scientific simulation, and data warehousing. See Also http://aws.amazon.com/elasticmapreduce.
Amazon Machine Image	An encrypted machine image stored in Amazon Elastic Block Store (p. 222) or Amazon Simple Storage Service. AMIs are like a template of a computer's root drive. They contain the operating system and can also include software and layers of your application, such as database servers, middleware, web servers, and so on.
Mechanical Turk	Provides an on-demand, scalable, human workforce to complete jobs that humans can do better than computers. See Also http://aws.amazon.com/mturk.
Amazon Relational Database Service	A web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks. See Also http://aws.amazon.com/rds.
Amazon Resource Name	A standardized way to refer to an AWS resource. For example: arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Bob.
Amazon Route 53	A web service you can use to create a new DNS service or to migrate your existing DNS service to the cloud. See Also http://aws.amazon.com/route53.
Amazon S3	See Amazon Simple Storage Service.
Amazon S3-Backed AMI	See instance store-backed AMI.
Amazon Simple Email Service	An easy-to-use, cost-effective email solution for applications. See Also http://aws.amazon.com/ses.
Amazon Simple Notification Service	A web service that enables applications, end-users, and devices to instantly send and receive notifications from the cloud. See Also http://aws.amazon.com/sns.
Amazon Simple Queue Service	Reliable and scalable hosted queues for storing messages as they travel between computers.

	See Also http://aws.amazon.com/sqs.
Amazon Simple Storage Service	Storage for the internet. You can use it to store and retrieve any amount of data at any time, from anywhere on the web. See Also http://aws.amazon.com/s3.
Amazon SimpleDB	A highly-available, scalable, and flexible non-relational data store that enables you to store and query data items using web service requests. See Also http://aws.amazon.com/simpledb.
Amazon Virtual Private Cloud	A web service that enables you to create a virtual network for your AWS resources. See Also http://aws.amazon.com/vpc.
Amazon Web Services	An infrastructure web services platform in the cloud for companies of all sizes. See Also http://aws.amazon.com.
AMI	See Amazon Machine Image.
application	A logical collection of AWS Elastic Beanstalk components, including environments, versions, and environment configurations. An application is conceptually similar to a folder.
analysis scheme	Language-specific Amazon CloudSearch text analysis options that are applied to a text field to control stemming and configure stopwords and synonyms.
Application Billing	The location where your customers manage the Amazon DevPay products they've purchased. This is the URL http://www.amazon.com/dp-applications.
application version	A specific, labeled iteration of an application that represents a functionally consistent set of deployable application code. A version points to an Amazon S3 object (a JAVA WAR file) that contains the application code.
approval	If a Worker's response satisfies your Human Intelligence Task (p. 233), you approve the assignment. When you approve an assignment Mechanical Turk transfers the HIT reward from your Mechanical Turk account to the Worker's Amazon Payments account.
ARN	See Amazon Resource Name.
assignment	When a worker (p. 251) finds a Human Intelligence Task (p. 233) (HIT) to complete, the worker accepts the HIT. Mechanical Turk creates an *assignment* to track the work to completion and store the answer the worker submits. The assignment belongs exclusively to the worker who accepted it and guarantees that the worker can submit results and be eligible for a reward—up until the HIT or assignment expires.
asynchronous bounce	A type of bounce (p. 225) that occurs when a receiver (p. 242) initially accepts an email message for delivery and then subsequently fails to deliver it.
attribute	Similar to a column on a spreadsheet, an attribute represents a data category. In Amazon SimpleDB, an attribute has a name (such as *color*), which has a value (such as *blue*) when applied to a data item.
authentication	The process of proving your identity to a system.
Auto Scaling	A web service designed to launch or terminate instance (p. 233)s automatically based on user-defined policies, schedules, and health checks. See Also http://aws.amazon.com/autoscaling.

Auto Scaling group	A representation of multiple Amazon Elastic Compute Cloud (p. 222) instance (p. 233)s that share similar characteristics, and that are treated as a logical grouping for the purposes of instance scaling and management.
Availability Zone	A distinct location within a region (p. 242) that is insulated from failures in other Availability Zones, and provides inexpensive, low-latency network connectivity to other Availability Zones in the same region.
AWS	See Amazon Web Services.
AWS CloudFormation	A service for writing or changing templates that create and delete related AWS resources together as a unit. See Also http://aws.amazon.com/cloudformation.
AWS Consolidated Billing	A billing option that lets you get a single bill for multiple AWS accounts. See Also http://aws.amazon.com/consolidated-billing.
AWS Elastic Beanstalk	See Also http://aws.amazon.com/elasticbeanstalk.
AWS Import/Export	A service for transferring large amounts of data between AWS and portable storage devices. See Also http://aws.amazon.com/importexport.
AWS Identity and Access Management	A web service that enables Amazon Web Services (p. 223) customers to manage users and user permissions within AWS. See Also http://aws.amazon.com/iam.
AWS Management Console	A graphical interface to manage compute, storage, and other cloud resources. See Also http://aws.amazon.com/console.
AWS Multi-Factor Authentication	An optional AWS account security feature. Once you enable AWS MFA, you must provide a six-digit, single-use code in addition to your sign-in credentials whenever you access secure AWS web site pages or the AWS Management Console. You get this single-use code from an authentication device that you keep in your physical possession. See Also http://aws.amazon.com/mfa/.
AWS Resources	See resource.
AWS VPN CloudHub	Enables secure communication between branch offices using a simple hub-and-spoke model, with or without a VPC.

B

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

basic monitoring	Monitoring of AWS-provided metrics derived at a 5-minute frequency.
batch	See document batch.
BGP ASN	Border Gateway Protocol Autonomous System Number. A unique identifier for a network, for use in BGP routing. Amazon EC2 supports all 2-byte ASN numbers in the range of 1 - 65334, with the exception of 7224, which is reserved.
blacklist	A list of IP addresses, email addresses, or domains that an Internet Service Provider (p. 234) suspects to be the source of spam (p. 246). The ISP blocks incoming emails from these addresses or domains.

block	A data set. Amazon EMR breaks large amounts of data into subsets. Each subset is called a data block. Amazon EMR assigns an ID to each block and uses a hash table to keep track of block processing.
block device	A storage device that supports reading and (optionally) writing data in fixed-size blocks, sectors, or clusters.
block device mapping	A mapping structure for every AMI and instance that specifies the block devices attached to the instance.
bootstrap action	A user-specified default or custom action that runs a script or an application on all nodes of a job flow before Hadoop starts.
Border Gateway Protocol Autonomous System Number	See BGP ASN.
bounce	A failed email delivery attempt.
breach	The condition in which a user-set threshold (upper or lower boundary) is passed. If the duration of the breach is significant, as set by a breach duration parameter, it can possibly start a scaling activity (p. 244).
bucket	A container for objects stored in Amazon S3. Every object is contained in a bucket. For example, if the object named `photos/puppy.jpg` is stored in the `johnsmith` bucket, then authorized users can access the object with the URL `http://johnsmith.s3.amazonaws.com/photos/puppy.jpg`.
bucket owner	Just as Amazon is the only owner of the domain name Amazon.com, only one person or organization can own a bucket in Amazon S3.
bundling	A commonly used term for creating an Amazon Machine Image (p. 222). It specifically refers to creating instance store-backed AMIs.

C

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

cache cluster	A logical cache distributed over multiple cache node (p. 225)s. A cache cluster can be set up with a specific number of cache nodes.
cache cluster identifier	Customer-supplied identifier for the cache cluster that must be unique for that customer in an AWS region.
cache engine version	The version of the Memcached service that is running on the cache node.
cache node	A fixed-size chunk of secure, network-attached RAM. Each cache node runs an instance of the Memcached service, and has its own DNS name and port. Multiple types of cache nodes are supported, each with varying amounts of associated memory.
cache node type	EC2 instance type used to run the cache node.
cache parameter group	A container for cache engine parameter values that can be applied to one or more cache clusters.

cache security group	A group maintained by ElastiCache that combines ingress authorizations to cache nodes for hosts belonging to Amazon EC2 security groups specified through the console or the API or command line tools.
canned access policy	A standard access control policy that you can apply to a bucket or object. Options include: private, public-read, public-read-write, and authenticated-read.
canonicalization	The process of converting data into a standard format that a service such as Amazon S3 can recognize.
capacity	Each Auto Scaling group (p. 224) is defined with a minimum and maximum compute size. The amount of available compute size at any time is the current capacity. A scaling activity (p. 244) increases or decreases the capacity—within the defined minimum and maximum values.
Cascading	Cascading is an open-source Java library that provides a query API, a query planner, and a job scheduler for creating and running Hadoop MapReduce applications. Applications developed with Cascading are compiled and packaged into standard Hadoop-compatible JAR files similar to other native Hadoop applications.
certificate	A credential that some AWS products use to authenticate AWS accounts and users. Also known as an X.509 certificate. The certificate is paired with a private key.
chargeable resources	Features or services whose use incurs fees. Although some AWS products are free, others include charges. For example, in an AWS CloudFormation stack (p. 246), AWS resources that have been created incur charges. The amount charged depends on the usage load. Use the Amazon Web Services Simple Monthly Calculator at http://calculator.s3.amazonaws.com/calc5.html to estimate your cost prior to creating instances, stacks, or other resources.
CIDR block	Classless Inter-Domain Routing. An Internet protocol address allocation and route aggregation methodology. See Also http://en.wikipedia.org/wiki/CIDR_notation.
CloudHub	See AWS VPN CloudHub.
cluster compute instance	A type of instance (p. 233) that provides a great amount of CPU power coupled with increased networking performance, making it well suited for High Performance Compute (HPC) applications and other demanding network-bound applications.
cluster placement group	A logical cluster compute instance (p. 226) grouping to provide lower latency and high-bandwidth connectivity between the instances.
CNAME	Canonical Name Record. A type of resource record in the Domain Name System (DNS) that specifies that the domain name is an alias of another, canonical domain name. More simply, it is an entry in a DNS table that lets you alias one fully qualified domain name to another.
complaint	The event in which a recipient (p. 242) who does not want to receive an email message clicks "Mark as Spam" within the email client, and the Internet Service Provider (p. 234) sends a notification to Amazon SES.
compound query	A search request that specifies multiple search criteria using the Amazon CloudSearch structured search syntax.
condition	Any restriction or detail about a permission. The condition is *D* in the statement "A has permission to do B to C where D applies." Conditions are always optional.
conditional parameter	See mapping.

configuration API	The Amazon CloudSearch API that you use to create, configure, and manage search domains.
configuration template	A series of key–value pairs that define parameters for various AWS products so that AWS Elastic Beanstalk can provision them for an environment.
confirmation email	The email Amazon Payments sends to your customers to notify them that a price change you scheduled has taken effect.
consistency model	The method a service uses to achieve high availability. For example, it could involve replicating data across multiple servers in a data center. See Also eventual consistency.
consistent read	When data is written or updated successfully, all copies of the data are updated in all AWS regions. However, it takes time for the data to propagate to all storage locations. A consistent read returns a result that reflects any writes that received a successful response before the read request—regardless of the region. By contrast, an eventually consistent read returns data from only one region and might not show the most recent write information. See Also eventual consistency.
console	See AWS Management Console.
Consolidated Billing	See AWS Consolidated Billing.
cooldown period	Amount of time during which Auto Scaling does not allow the desired size of the Auto Scaling group (p. 224) to be changed by any other notification from a CloudWatch alarm (p. 221).
core node	An EC2 instance (p. 229) that runs Hadoop map and reduce tasks and stores data using the Hadoop Distributed File System (HDFS). Core nodes are managed by the master node (p. 237), which assigns Hadoop tasks to nodes and monitors their status. The EC2 instances you assign as core nodes are capacity that must be allotted for the entire job flow run. Because core nodes store data, you can't remove them from a job flow. However, you can add more core nodes to a running job flow. Core nodes run both the DataNodes and TaskTracker Hadoop daemons.
corpus	A collection of data that you want to search.
credentials	Also called *access credentials* or *security credentials*. In authentication and authorization, a system uses credentials to identify who is making a call and whether to allow the requested access. In AWS, these credentials are typically the access key ID (p. 220) and the secret access key (p. 245).
customer gateway	A router or software application on your side of a VPN tunnel that is managed by Amazon VPC. The internal interfaces of the customer gateway are attached to one or more devices in your home network. The external interface is attached to the VPG (p. 251) across the VPN tunnel.

D

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

dashboard	See service health dashboard.

database engine	The database software and version running on the DB instance (p. 228).
database name	The name of a database hosted in a DB instance (p. 228). A DB instance can host multiple databases, but databases hosted by the same DB instance must each have a unique name within that instance.
DB compute class	Size of the database compute platform used to run the instance.
DB instance	An isolated database environment running in the cloud. A DB instance can contain multiple user-created databases.
DB instance identifier	User-supplied identifier for the DB instance. The identifier must be unique for that user in an AWS region (p. 242).
DB parameter group	A container for database engine parameter values that apply to one or more DB instance (p. 228)s.
DB security group	A method that controls access to the DB instance (p. 228). By default, network access is turned off to DB instances. After ingress is configured for a security group, the same rules apply to all DB instances associated with that group.
DB snapshot	A user-initiated point backup of a DB instance.
Dedicated Instance	An instance that is physically isolated at the host hardware level and launched within a VPC.
Dedicated Reserved Instance	An option you purchase to guarantee that sufficient capacity will be available to launch Dedicated Instances into a VPC.
default deny	The default result from a policy (p. 239) in the absence of an allow (p. 221) or explicit deny (p. 231). For example: if a user (p. 249) requests to use Amazon SQS, but the only policy that applies to the user states that the user can use Amazon SimpleDB, then that policy results in a default deny.
delete marker	An object with a key and version ID, but without content. Amazon S3 inserts delete markers automatically into versioned buckets when an object is deleted.
deliverability	The likelihood that an email message will arrive at its intended destination.
deliveries	The number of emails, sent through Amazon SES, that were accepted by an Internet Service Provider (p. 234) for delivery to recipient (p. 242)s over a period of time.
detailed monitoring	Monitoring of AWS-provided metrics derived at a 1-minute frequency.
Description property	A property added to parameters, resources, resource properties, mappings, and outputs, to help you to document AWS CloudFormation template elements.
DevPay Activity	The location (Amazon DevPay Activity) where you manage the Amazon DevPay products you've created.
dimension	A name/value pair (for example, InstanceType=m1.small, or EngineName=mysql), that contains additional information to identify a metric.
discussion forums	A place where AWS users can post technical questions and feedback to help accelerate their development efforts and to engage with the AWS community. The discussion forums are located at http://aws.amazon.com/forums/.
distributed cache	A Hadoop feature that allow you to transfer files from a distributed file system to the local file system. It can distribute data and text files as well as more complex types such as archives and JARs.

distribution	A link between an origin server (such as an Amazon S3 bucket) and a domain name, which CloudFront automatically assigns. Through this link, CloudFront identifies the object you have stored in your origin server (p. 238).
DKIM	DomainKeys Identified Mail. A standard that email senders use to sign their messages. ISPs use those signatures to verify that messages are legitimate. For more information, see http://www.dkim.org.
DNS	See Domain Name System (DNS).
document	Represents an item that can be returned as a search result in Amazon CloudSearch. Each document has a collection of fields that contain the data that can be searched or returned. The value of a field can be either a string or a number. Each document must have a unique ID and at least one field.
document batch	A collection of add and delete document operations for Amazon CloudSearch. You use the document service API to submit batches to update the data in your search domain.
document service API	The Amazon CloudSearch API that you use to submit document batches to update the data in a search domain.
document service endpoint	The URL that you connect to when sending document updates to an Amazon CloudSearch domain. Each search domain has a unique document service endpoint that remains the same for the life of the domain.
domain	All Amazon SimpleDB information is stored in domains. Domains are like tables that contain similar data. You can execute queries against a domain, but cannot execute joins between domains. See Also search domain.
Domain Name System (DNS)	A distributed naming system that associates network information with human-readable domain names on the Internet.
Donation button	An HTML-coded button to provide an easy and secure way for US-based, IRS-certified 501(c)3 nonprofit organizations to solicit donations.

E

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

EBS	See Amazon Elastic Block Store.
EC2	See Amazon Elastic Compute Cloud.
EC2 compute unit	An AWS standard for compute CPU and memory. This measure enables you to evaluate the CPU capacity of different EC2 instance types.
EC2 instance	In Amazon EC2, this is simply an instance. Other AWS services use the term EC2 instance to distinguish these instances from other types of instances they support.
edge location	A site that CloudFront uses to cache copies of your content for faster delivery to users at any location.
Elastic Block Store	See Amazon Elastic Block Store.

elastic IP address	A fixed (static) IP address that you have allocated in Amazon EC2 or Amazon VPC and then attached to an instance. Elastic IP addresses are associated with your account, not a specific instance. They are *elastic* because you can easily allocate, attach, detach, and free them as your needs change. Unlike traditional static IP addresses, elastic IP addresses allow you to mask instance or Availability Zone failures by rapidly remapping your public IP addresses to another instance.
Elastic Load Balancing	A web service that improves an application's availability by distributing incoming traffic between two or more EC2 instance (p. 229)s. See Also http://aws.amazon.com/elasticloadbalancing.
elastic network interface	An additional network interface that can be attached to an instance (p. 233). ENIs include a primary private IP address, one or more secondary private IP addresses, an elastic IP address (optional), a MAC address, membership in specified security groups, a description, and a source/destination check flag. You can create an ENI, attach it to an instance, detach it from an instance, and attach it to another instance.
endpoint	A URL that identifies a host and port as the entry point for a web service. Every web service request contains an endpoint. Most AWS products provide regional endpoints to enable faster connectivity. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference* ElastiCache: The DNS name of a cache node (p. 225). Amazon RDS: The DNS name of a DB instance (p. 228). AWS CloudFormation: The DNS name or IP address of the server that receives an HTTP request.
endpoint port	ElastiCache: The port number used by a cache node (p. 225). Amazon RDS: The port number used by a DB instance (p. 228).
environment	A specific running instance of an application (p. 223). The application has a CNAME and includes an application version and a customizable configuration (which is inherited from the default container type).
environment configuration	A collection of parameters and settings that define how an environment and its associated resources behave.
ephemeral store	See instance store.
epoch	The date from which time is measured. For most Unix environments, the epoch is January 1, 1970.
eventual consistency	The method through which AWS products achieve high availability, which involves replicating data across multiple servers in Amazon's data centers. When data is written or updated and "Success" is returned, all copies of the data are updated. However, it takes time for the data to propagate to all storage locations. The data will eventually be consistent, but an immediate read might not show the change. Consistency is usually reached within seconds, but a high system load might increase this time.
eventually consistent read	See consistent read.
eviction	An *eviction* occurs when CloudFront deletes an object from an edge location (p. 229) before its expiration time. If an object in an edge location isn't frequently requested, CloudFront might evict the object (remove the object before its expiration date) to make room for objects that are more popular.

expiration	*Expiration* occurs when CloudFront stops serving an object from an edge location (p. 229). The next time the edge location needs to serve that object, CloudFront gets a new copy from the origin server (p. 238).
explicit deny	An *explicit deny* results from a statement that has effect=deny, assuming that any stated conditions are met. Example: *Deny all requests from Antarctica*. Any request that comes from Antarctica will always be denied no matter what any other policy (p. 239) might allow.
explicit launch permission	An Amazon Machine Image (p. 222) launch permission granted to a specific AWS account.
exponential backoff	A strategy that incrementally increases the wait between retry attempts in order to reduce the load on the system and increase the likelihood that repeated requests will succeed. For example, client applications might wait up to 400 milliseconds before attempting the first retry, up to 1600 milliseconds before the second, up to 6400 milliseconds (6.4 seconds) before the third, and so on.
expression	A numeric expression that you can use to control how search hits are sorted. You can construct Amazon CloudSearch expressions using numeric fields, other rank expressions, a document's default relevance _score, and standard numeric operators and functions. When you use the `sort` option to specify an expression in a search request, the expression is evaluated for each search hit and the hits are listed according to their expression values.

F

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

facet	An Amazon CloudSearch index field that represents a category that you want to use to refine and filter search results.
facet enabled	An Amazon CloudSearch index field option that enables facet information to be calculated for the field.
FBL	See feedback loop.
federated identity management	Allows individuals to sign in to different networks or services, using the same group or personal credentials to access data across all networks. With identity federation in AWS, external identities (federated users) are granted secure access to resources in an AWS account without having to create IAM users. These external identities can come from a corporate identity store (such as LDAP or Windows Active Directory) or from a third party (such as Login with Amazon, Facebook, or Google). AWS federation also supports SAML 2.0.
federated user	See federated identity management.
feedback loop	The mechanism by which a mailbox provider (for example, an Internet Service Provider (p. 234)) forwards a recipient (p. 242)'s complaint (p. 226) back to the sender (p. 245).
field weight	The relative importance of a text field in a search index. Field weights control how much matches in particular text fields affect a document's relevance _score.
filter	A criterion you specify to limit the results when you list or describe your Amazon EC2 resources.

filter query	A way to filter search results without affecting how the results are scored and sorted. Specified with the Amazon CloudSearch `fq` parameter.
FIM	See federated identity management.
format version	See template format version.
forums	See discussion forums.
function	See intrinsic function.
fuzzy search	A simple search query that uses approximate string matching (fuzzy matching) to correct for typographical errors and misspellings.

G

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

geospatial search	A search query that uses locations specified as a latitude and longitude to determine matches and sort the results.
gibibyte	A contraction of giga binary byte, a gibibyte is 2^30 bytes or 1,073,741,824 bytes. A gigabyte is 10^9 or 1,000,000,000 bytes.

H

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

Hadoop	See http://hadoop.apache.org.
hard bounce	A persistent email delivery failure such as "mailbox does not exist."
hardware VPN	A hardware-based IPsec VPN connection over the Internet.
HDFS	Hadoop Distributed File System. The HDFS file system stores large files across multiple machines. It achieves reliability by replicating the data across multiple hosts, and hence does not require RAID storage on hosts.
health check	A system call to check on the health status of each instance in an Auto Scaling group.
high-quality email	Email that recipients find valuable and want to receive. Value means different things to different recipients and can come in the form of offers, order confirmations, receipts, newsletters, etc.
highlights	Excerpts returned with Amazon CloudSearch results that show where the search terms appear within the text of the matching documents.
highlight enabled	An Amazon CloudSearch index field option that enables matches within the field to be highlighted.
hit	A document that matches the criteria specified in a search request. Also referred to as a *search result*.

HIT	See Human Intelligence Task.
Hive	An open source, data warehouse and analytic package that runs on top of Hadoop. Hive scripts use an SQL-like language called Hive QL (query language) that abstracts the MapReduce programming model and supports typical data warehouse interactions.
HMAC	Hash-based Message Authentication Code. A specific construction for calculating a message authentication code (MAC) involving a cryptographic hash function in combination with a secret key. You can use it to verify both the data integrity and the authenticity of a message at the same time. AWS calculates the HMAC using a standard, cryptographic hash algorithm, such as SHA-256.
hosted zone	A collection of resource record sets that Amazon Route 53 hosts. Like a traditional DNS zone file, a hosted zone represents a collection of records that are managed together under a single domain name.
Human Intelligence Task	A task that a Requester (p. 243) submits to Mechanical Turk for workers (p. 251) to perform. A HIT represents a single, self-contained task, for example, "Identify the car color in the photo." HITs contain all of the information a worker needs to answer a question, including the kinds of answers you would consider valid.
HVM virtualization	Hardware Virtual Machine virtualization. Allows the guest VM to run as though it is on a native hardware platform, except that it still uses paravirtual (PV) network and storage drivers for improved performance. See Also PV virtualization.

I

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

image	See Amazon Machine Image.
import/export station	A machine that uploads or downloads your data to, or from, Amazon S3.
import log	A report that contains details about how AWS Import/Export processed your data.
index	See search index.
index field	A name-value pair that is included in an Amazon CloudSearch domain's index. An index field can contain text or numeric data, dates, or a location.
indexing options	Configuration settings that define an Amazon CloudSearch domain's index fields, how document data is mapped to those index fields, and how the index fields can be used.
instance	A copy of an Amazon Machine Image running as a virtual server in the AWS cloud.
instance family	A general instance type (p. 234) grouping using either storage or CPU capacity.
instance group	A Hadoop cluster contains one master instance group that contains one master node (p. 237), a core instance group containing one or more core node (p. 227) and an optional task node (p. 248) instance group, which can contain any number of task nodes.

instance store	Disk storage that is physically attached to the host computer for an EC2 instance, and therefore has the same lifespan as the instance. When the instance terminates, you lose any data in the instance store.
instance store-backed AMI	Instances launched from this type of AMI use an instance store volume as the root device. Compare this with instances launched from Amazon EBS-backed AMIs, which use an Amazon EBS volume as the root device.
instance type	A specification that defines the memory, CPU, storage capacity, and hourly cost for an instance. Some instance types are designed for standard applications, whereas others are designed for CPU-intensive, memory-intensive applications, and so on.
Internet gateway	Connects a network to the Internet. You can route traffic for IP addresses outside your VPC (p. 250) to the Internet gateway.
Internet Service Provider	A company that provides subscribers with access to the Internet. Many ISPs are also mailbox provider (p. 236)s. Mailbox providers are sometimes referred to as ISPs, even if they only provide mailbox services.
intrinsic function	A special action in a template that assigns values to properties not available until runtime. These functions follow the format *Fn::Attribute*, such as `Fn::GetAtt`. Arguments for intrinsic functions can be parameters, pseudo parameters, or the output of other intrinsic functions.
IP address	All EC2 instances are assigned two IP addresses at launch, which are directly mapped to each other through network address translation (NAT): a private IP address (following RFC 1918) and a public IP address. Instances launched in a VPC are assigned only a private IP address. Instances launched in your default VPC are assigned both a private IP address and a public IP address.
ISP	See Internet Service Provider.
issuer	The issuer is the person who writes a policy to grant permissions to a resource. The issuer (by definition) is always the resource owner. AWS does not permit Amazon SQS users to create policies for resources they don't own. If John is the resource owner, AWS authenticates John's identity when he submits the policy he's written to grant permissions for that resource.
item	Similar to rows on a spreadsheet, items represent individual objects that contain one or more value-attribute pairs.
item name	An identifier for an item. The identifier must be unique within the domain (p. 229).

J

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

job flow	A job flow specifies the complete processing of the data. It's comprised of one or more steps, which specify all of the functions to be performed on the data.
job ID	A five-character, alphanumeric string that uniquely identifies a storage device in your shipment. AWS issues the job ID in response to a `CREATE JOB` email command.
job prefix	The AWS Import/Export process generates a log file. The log file name always ends with the phrase *import-log-* followed by your Job ID. There is a remote

	chance that you already have an object with this name. To avoid a key collision, you can add an optional prefix to the log file. See Also key prefix.
JSON	JavaScript Object Notation. A lightweight data-interchange format. For information about JSON, see http://www.json.org/.
junk folder	The location where email messages that various filters determine to be of lesser value are collected so that they do not arrive in the recipient (p. 242)'s inbox, but are still accessible to the recipient. This is also referred to as a spam (p. 246) or bulk folder.

K

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

key	A credential that identifies an AWS account or user to AWS (such as the AWS secret access key (p. 245)). Amazon S3, Amazon EMR: The unique identifier for an object in a bucket. Every object in a bucket has exactly one key. Because a bucket and key together uniquely identify each object, you can think of Amazon S3 as a basic data map between the *bucket* + *key*, and the object itself. You can uniquely address every object in Amazon S3 through the combination of the web service endpoint, bucket name, and key, for example: `http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl`, where `doc` is the name of the bucket, and `2006-03-01/AmazonS3.wsdl` is the key. AWS Import/Export: The name of an object in Amazon S3. It is a sequence of Unicode characters whose UTF-8 encoding cannot exceed 1024 bytes. If a key, for example, logPrefix + import-log-JOBID, is longer than 1024 bytes, AWS Elastic Beanstalk returns an `InvalidManifestField` error. IAM: In the context of writing a policy (p. 239): A specific characteristic that is the basis for restricting access (such as the current time, or the IP address of the requester).
key pair	A set of security credentials you use to prove your identity electronically. A key pair consists of a private key and a public key.
key prefix	A logical grouping of the objects in a bucket (p. 225). The prefix value is similar to a directory name that enables you to store similar data under the same directory in a bucket.

L

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

launch configuration	A set of descriptive parameters used to create new EC2 instances in an Auto Scaling activity. A template that an Auto Scaling group (p. 224) uses to launch new EC2 instances. The launch configuration contains information such as the Amazon Machine Image (p. 222) ID, the instance type, key pairs, security groups, and block device mappings, among other configuration settings.

launch permission	An Amazon Machine Image (p. 222) (AMI) attribute that allows users to launch an AMI.
lifecycle	The lifecycle state of the EC2 instance (p. 229) contained in an AutoScalingGroup. EC2 instances progress through several states over their lifespan; these include *Pending*, *InService*, *Terminating* and *Terminated*.
load balancer	A load balancer is a combination of a DNS name and a set of ports, which together provide a destination for all requests intended for your application. A load balancer can distribute traffic to multiple application instances across every Availability Zone (p. 224) within a region (p. 242). Load balancers can span multiple Availability Zones within an Amazon EC2 region, but they cannot span multiple regions.
logical name	A case-sensitive unique string within an AWS CloudFormation template that identifies a resource (p. 243), mapping (p. 236), parameter, or output. In an AWS CloudFormation template, each parameter, resource, property, mapping, and output must be declared with a unique logical name. You use the logical name when dereferencing these items using the `Ref` function.

M

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

machine utilization	The amount of machine capacity used to complete a particular request (for example SELECT, GET, PUT, and so on), normalized to the hourly capacity of a standard processor. Machine utilization is measured in machine hour increments.
Mail Transfer Agent (MTA)	Software that transports email messages from one computer to another by using a client-server architecture.
mailbox provider	An organization that provides email mailbox hosting services. Mailbox providers are sometimes referred to as Internet Service Provider (p. 234)s, even if they only provide mailbox services.
mailbox simulator	A set of email addresses that you can use to test an Amazon SES-based email sending application without sending messages to actual recipients. Each email address represents a specific scenario (such as a bounce or complaint) and generates a typical response that is specific to the scenario.
main route table	The default route table that any new VPC subnet uses for routing. You can associate a subnet with a different route table of your choice. You can also change which route table is the main route table.
manifest	When sending a *create job* request for an import or export operation you describe your job in a text file called a manifest. The manifest file is a YAML-formatted file that specifies how to transfer data between your storage device and the AWS cloud.
MapReduce	See http://hadoop.apache.org/docs/r1.2.0/mapred_tutorial.html.
mapper	An executable that splits the raw data into key/value pairs. The reducer uses the output of the mapper, called the *intermediate results*, as its input.
mapping	A way to add conditional parameter values to an AWS CloudFormation template. You specify mappings in the template's optional Mappings section and retrieve the desired value using the *`FN::FindInMap`* function.

marker	See pagination.
master node	A process running on an Amazon Machine Image (p. 222) that keeps track of the work its core and task nodes complete.
maximum price	The maximum price you will pay to launch one or more Spot Instances. If your maximum price exceeds the current Spot Price (p. 246) and your restrictions are met, Amazon EC2 launches instances on your behalf.
maximum send rate	The maximum number of emails that you can send per second using Amazon SES.
member resources	See resource.
message ID	Amazon SES: A unique identifier that is assigned to every email message that is sent.
	Amazon SQS: The identifier returned when you send a message to a queue.
metadata	Amazon S3, Amazon EMR: A set of name/value pairs that describe the object. These include default metadata such as the date last modified and standard HTTP metadata such as Content-Type. Users can also specify custom metadata at the time they store an object. Amazon EC2: Data about an EC2 instance (p. 229) that the instance can retrieve to determine things about itself, such as, the instance type, the IP address, and so on.
metric	An element of time-series data defined by a unique combination of exactly one namespace, exactly one metric name, and between zero and ten dimensions. Metrics and the statistics derived from them are the basis of Amazon CloudWatch.
metric name	The primary identifier of a metric, used in combination with a namespace and optional dimensions.
MFA	See AWS Multi-Factor Authentication.
micro instance	A type of EC2 instance (p. 229) that is more economical to use if you have occasional bursts of high CPU activity.
MIME	See Multipurpose Internet Mail Extensions (MIME).
MTA	See Mail Transfer Agent (MTA).
Multi-AZ deployment	A primary DB instance (p. 228) that has a synchronous standby replica in a different Availability Zone (p. 224). The primary DB instance is synchronously replicated across Availability Zones to the standby replica.
Multi-Factor Authentication	See AWS Multi-Factor Authentication.
multi-valued attribute	An attribute with more than one value.
multipart upload	A feature that allows you to upload a single object as a set of parts.
Multipurpose Internet Mail Extensions (MIME)	An Internet standard that extends the email protocol to include non-ASCII text and non-text elements like attachments.
Multitool	A Cascading (p. 226) application that provides a simple command-line interface for managing large datasets.

N

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

namespace	An abstract container that provides context for the items (names, or technical terms, or words) it holds, and allows disambiguation of homonym items residing in different namespaces.
NAT	Network address translation.
NAT instance	An instance that is configured to perform NAT (p. 238) in a VPC. A NAT instance enables private instances in the VPC to initiate Internet-bound traffic without being directly reachable from the Internet.
network ACL	An optional layer of security that acts as a firewall for controlling traffic in and out of a subnet. You can associate multiple subnets with a single network ACL, but a subnet can be associated with only one network ACL at a time.
node	After an Amazon Machine Image (p. 222) is launched, the resulting running system is referred to as a node. All instances based on the same AMI are identical at start-up. Any information about the node is lost when the node terminates or fails.
NoEcho	A property of AWS CloudFormation parameters that will prevent the otherwise default reporting of names and values of a template parameter. Declaring the *`NoEcho`* property causes the parameter value to be masked with asterisks in the report by the `cfn-describe-stacks` command.
notification email	The email Amazon Payments sends to your customers to notify them of an upcoming price change you've scheduled.
null object	A null object is one whose version ID is null. Amazon S3 adds a null object to a bucket when versioning (p. 250) for that bucket is suspended. It is possible to have only one null object for each key in a bucket.

O

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

object	Amazon S3: The fundamental entity type stored in Amazon S3. Objects consist of object data and metadata. The data portion is opaque to Amazon S3. CloudFront: Any entity that can be served either over HTTP or a version of RTMP.
on-demand instance	An Amazon EC2 pricing option that charges you for compute capacity by the hour with no long-term commitment.
operation	An API function. Also called an *action*.
origin access identity	Also called OAI. A virtual identity you use when giving your distribution permission to fetch a private object from your origin server (Amazon S3 bucket).
origin server	The Amazon S3 bucket or custom origin containing the definitive original version of the content you deliver through CloudFront.

P

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

pagination	Some APIs that return a potentially large list of records can return a subset by using a value to set the maximum number of returned records. They then provide a marker, which identifies the last record returned so that in a subsequent call, the user can get the next sequence of records.
paid AMI	An Amazon Machine Image (AMI) that you sell to other Amazon EC2 users using Amazon DevPay.
paravirtual virtualization	See PV virtualization.
part	In a multipart upload request, each part is a contiguous portion of the object's data.
passive authorization	A passive authorization happens when you inform customers at least 14 days in advance of a price change for your product, and they don't take any action; the price change is accepted automatically.
PAT	Port address translation.
period	See sampling period.
permission	A statement within a policy (p. 239) that allows or disallows access to a particular resource. You can state any permission like this: "A has permission to do B to C where D applies." For example, Jane (A) has permission to read messages (B) from John's Amazon SQS queue (C), as long as she asks to receive only a maximum of 10 messages from the queue at a time (D). Whenever Jane sends a request to Amazon SQS to use John's queue, the service checks to see if she has permission and if the request satisfies the conditions John set forth in the permission.
persistent identifier	Also called PID. An encoded string that represents the relationship between a customer and the owner of Amazon DevPay products. After a customer purchases one of your products, you can use the PID to confirm the status of the customer's subscription to the product.
persistent storage	A long-term data storage solution. Options within AWS are: Amazon S3, Amazon EBS, and Amazon SimpleDB.
physical name	A unique label AWS CloudFormation assigns to each resource when creating a stack (p. 246). Some AWS CloudFormation commands accept the physical name as a value with the `--physical-name` parameter.
Pig	An open-source Apache library that runs on top of Hadoop. The library takes SQL-like commands written in a language called Pig Latin and converts those commands into MapReduce job flows.
policy	A policy is the formal description of the permissions for a resource. The access policy language distinguishes between a *policy* and a *statement*. A policy is the complete document that can contain many different permissions for a given resource. A statement is the description of an individual permission. Therefore a policy can contain multiple statements. For example, a policy could specify that Jane can use John's queue (one statement), and Bob cannot use John's queue (another statement).

	Auto Scaling: An object that stores the information needed to launch or terminate instances for an Auto Scaling group. Executing the policy causes instances to be launched or terminated. You can configure an alarm (p. 221) to invoke an Auto Scaling policy.
pre-signed URL	A URL that uses query string authentication (p. 241).
prefix	See job prefix.
Premium Support	A one-on-one, fast-response support channel that AWS customers can subscribe to for support for AWS infrastructure services. See Also https://aws.amazon.com/premiumsupport/.
principal	The principal is the person or persons who receive the permission in the policy (p. 239). The principal is A in the statement "A has permission to do B to C where D applies." In a policy, you can set the principal to "anyone" (that is, you can specify a wildcard to represent all people). You might do this, for example, if you don't want to restrict access based on the actual identity of the requester, but instead on some other identifying characteristic such as the requester's IP address. The concept of principals doesn't apply to a IAM policy, because these policies are attached to users or groups.
private IP address	All EC2 instances are assigned two IP addresses at launch, which are directly mapped to each other through Network Address Translation (NAT): a private address (following RFC 1918) and a public address. *Exception:* Instances launched in Amazon VPC are assigned only a private IP address.
private subnet	A VPC subnet whose instances cannot be reached from the Internet.
product activation	See activation.
product code	The product code is an eight-character string that identifies your registered product to AWS.
product identification token	See product token.
product token	The product token is a long encoded string that identifies the product to AWS. You might also see the product token referred to as the *product identification token*.
properties	See resource property.
property rule	A JSON (p. 235)-compliant markup standard for declaring properties, mappings, and output values in an AWS CloudFormation template.
Provisioned IOPS	A storage option designed to deliver fast, predictable, and consistent I/O performance. When you specify an IOPS rate while creating a DB instance, Amazon RDS provisions that IOPS rate for the lifetime of the DB instance.
pseudo parameter	A predefined setting, such as `AWS:StackName` that can be used in AWS CloudFormation templates without having to declare them. You can use pseudo parameters anywhere you can use a regular parameter.
public AMI	An Amazon Machine Image (p. 222) that all AWS accounts have permission to launch.
public data set	A large set of public data that can be seamlessly integrated into AWS cloud-based applications. Amazon stores public data sets at no charge to the community and, like all AWS services, users pay only for the compute and storage they use for

	their own applications. These data sets currently include data from the Human Genome Project, the U.S. Census, Wikipedia, and other sources. See Also http://aws.amazon.com/publicdatasets.
public IP address	All EC2 instances are assigned two IP addresses at launch, which are directly mapped to each other through Network Address Translation (NAT): a private address (following RFC 1918) and a public address. *Exception:* Instances launched in Amazon VPC are assigned only a private IP address.
public subnet	A subnet whose instances can be reached from the Internet.
purchase URL	The URL your customers use to purchase your product. When you advertise your product, you provide the purchase URL as the sign-up link for customers to use.
PV virtualization	Paravirtual virtualization. Allows guest VMs to run on host systems that do not have special support extensions for full hardware and CPU virtualization. Because PV guests run a modified operating system that does not use hardware emulation, they cannot provide hardware-related features such as enhanced networking or GPU support. See Also HVM virtualization.

Q

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

Qualification	A property associated with a worker (p. 251) that represents that worker's skill, ability, or reputation. A Requester (p. 243) can use Qualifications to control which workers can perform HITs. Each worker can have multiple Qualifications.
Qualification requirements	A Human Intelligence Task (p. 233) can have Qualification requirements that a worker's Qualifications (q.v.) must meet before the worker can accept that HIT.
Qualification test	A form, similar to a HIT, containing a set of questions that the worker must complete successfully to receive a particular Qualification (p. 241).
Qualification type	Just as each worker (p. 251) has one or more Qualification (p. 241), each Human Intelligence Task (p. 233) has one or more Qualification type. These types specify what Qualifications the worker must have.
Query	A type of HTTP-based request interface that generally uses only the GET or POST HTTP method and a query string with parameters. See Also REST, REST-Query.
query string authentication	An AWS feature that lets you place the authentication information in the HTTP request query string instead of in the Authorization header. For example: with Amazon DevPay, query string authentication enables your product to give anyone easy, URL-based access to objects in the customer's bucket.
queue	A sequence of messages or jobs held in temporary storage awaiting transmission or processing.
queue URL	A URL that uniquely identifies a queue.
quota	Amazon RDS: The maximum number of DB instance (p. 228)s and available storage you can use. ElastiCache: The maximum number of the following items:

- The number of cache clusters for each AWS account
- The number of cache nodes per cache cluster
- The total number of cache nodes per AWS account across all cache clusters created by that AWS account

R

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

range GET	A range GET specifies a byte range of data to get for a download. If an object is large, you can break up a download into smaller units by sending multiple range GET requests that each specify a different byte range to GET.
raw email	A type of *sendmail* request that allows you to specify the email headers and MIME types.
RDS	See Amazon Relational Database Service.
read replica	An active copy of another DB instance. Any updates to the data on the source DB instance are replicated to the read replica DB instance using the built-in replication feature of MySQL 5.1.
receipt handle	An identifier you get when you receive a message from the queue. This identifier is required to delete a message from the queue or when changing a message's visibility timeout.
receiver	The entity that consists of the network systems, software, and policies that manage email delivery for a recipient (p. 242).
recipient	Amazon SES: The person or entity receiving an email message. For example, a person named in the "To" field of a message.
redirect URL	The page on your own website that you want customers to see at the end of the purchase process for your product. You provide the URL when you register the product with Amazon DevPay.
reducer	An executable in the MapReduce process that uses the intermediate results from the mapper and processes them into the final output.
reference	A means of inserting a property from one AWS resource into another. For example, you could insert an Amazon EC2 security group property into an Amazon RDS resource.
region	A named set of AWS resources in the same geographical area. A region comprises at least two Availability Zones.
reply path	The email address to which an email reply is sent. This is different from the return path (p. 244).
reputation	1. An Amazon SES metric, based on factors that might include bounces, complaints, and other metrics, regarding whether or not a customer is sending high-quality emails.

	2. A measure of confidence, as judged by an Internet Service Provider (p. 234) or other entity that an IP address that they are receiving emails from is not the source of spam (p. 246).
requester	A requester is a person who sends a request to an AWS service and asks for access to a particular resource. The requester sends a request to AWS that essentially says: "Can A do B to C where D applies?" In this question, the requester is A. See Also Requester.
Requester	(Note capitalization) A company, organization, or person that creates and submits tasks (a Human Intelligence Task (p. 233)) to Mechanical Turk; for workers (p. 251) to perform.
Requester Pays	An Amazon S3 feature that allows a bucket owner (p. 225) to specify that anyone who requests access to objects in a particular bucket must pay the data transfer and request costs.
reservation	A collection of EC2 instances started as part of the same launch request. Not to be confused with a Reserved Instance (p. 243).
Reserved Instance	A pricing option that lets you make a low, one-time payment for each instance to reserve and receive a significant discount on the hourly usage charge for that instance.
Reserved Instance Marketplace	Matches sellers who have reserved capacity that they no longer need with buyers who are looking to purchase additional capacity. Reserved Instances that you purchase from third-party sellers will have less than a full standard term remaining and can be sold at different upfront prices. The usage or reoccurring fees will remain the same as the fees set when the Reserved Instances were originally purchased. Full standard terms for Reserved Instances available from AWS run for one year or three years.
resource	1. The objects you work with on AWS. This includes buckets, domains, instances, queues, and so on. 2. Tools, code, and documents that AWS provides to support users. 3. An object that the principal (p. 240) requests access to. The resource is C in the statement "A has permission to do B to C where D applies." 4. A required element of an AWS CloudFormation stack (p. 246). Each stack contains at least one resource, such as an Auto Scaling LaunchConfiguration. All resources in a stack must be created successfully for the stack to be created.
resource property	A value required when including an AWS resource in an AWS CloudFormation stack (p. 246). Each resource may have one or more properties associated with it. For example, an `AWS::EC2::Instance` resource may have a `UserData` property. In an AWS CloudFormation template, resources must declare a properties section, even if the resource has no properties.
resource record	Also called *resource record set*. Standard DNS terminology. See Also http://en.wikipedia.org/wiki/Domain_Name_System.
REST	A type of HTTP-based request interface that generally uses only the GET or POST HTTP method and a query string with parameters. Sometimes known as Query. In some implementations of a REST interface, other HTTP verbs besides GET and POST are used.

REST-Query	Also known as Query or HTTP Query. This is a type of HTTP request that generally uses only the GET or POST HTTP method and a query string with parameters. Compare this with REST, which is a type of HTTP request that uses any HTTP method (GET, DELETE, POST, etc.), a resource, HTTP headers, and possibly a query string with parameters.
return enabled	An Amazon CloudSearch index field option that enables the field's values to be returned in the search results.
return path	The email address to which bounced emails are returned. The return path is specified in the header of the original email. This is different from the reply path (p. 242).
reward	The money a Requester (p. 243) pays a worker (p. 251) for satisfactory work done on the Requester's Human Intelligence Task (p. 233)s.
rollback	A return to a previous state that follows the failure to create an object, such as AWS CloudFormation stack (p. 246). All resources associated with the failure are deleted during the rollback. For AWS CloudFormation, you can override this behavior using the `--disable-rollback` option on the command line.
root device volume	Contains the image used to boot the instance. If you launched the instance from an AMI backed by instance store, this is an instance store volume created from a template stored in Amazon S3. If you launched the instance from an AMI backed by Amazon EBS, this is an Amazon EBS volume created from an Amazon EBS snapshot.
route table	A set of routing rules that controls the traffic leaving any subnet that is associated with the route table. You can associate multiple subnets with a single route table, but a subnet can be associated with only one route table at a time.

S

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

sampling period	A defined duration of time, such as one minute, over which CloudWatch computes a statistic (p. 247).
sandbox	A testing location where you can test the functionality of your application without affecting production, incurring charges, or purchasing products.
	Amazon SES: An Amazon SES environment that is designed for developers to test and evaluate the service. In the sandbox, you have full access to the Amazon SES API, but you can only send messages to verified email addresses and the mailbox simulator. To get out of the sandbox, you need to apply for production access. Accounts in the sandbox also have lower sending limits (p. 245) than production accounts.
scaling activity	A process that changes the size, configuration, or makeup of an Auto Scaling group (p. 224) by launching or terminating instances. For more information, see Auto Scaling Concepts in the Auto Scaling Developer Guide.
search API	The Amazon CloudSearch API that you use to submit search requests to a search domain.

search domain	Encapsulates your searchable data and the search instances that handle your search requests. You typically set up a separate Amazon CloudSearch domain for each different collection of data that you want to search.
search domain configuration	An Amazon CloudSearch domain's indexing options, analysis schemes, expressions, suggesters, access policies, and scaling and availability options.
search enabled	An Amazon CloudSearch index field option that enables the field data to be searched.
search endpoint	The URL that you connect to when sending search requests to a search domain. Each Amazon CloudSearch domain has a unique search endpoint that remains the same for the life of the domain.
search index	A representation of your searchable data that facilitates fast and accurate data retrieval.
search instance	A compute resource that indexes your data and processes search requests. An Amazon CloudSearch domain has one or more search instances, each with a finite amount of RAM and CPU resources. As your data volume grows, more search instances or larger search instances are deployed to contain your indexed data. When necessary, your index is automatically partitioned across multiple search instances. As your request volume or complexity increases, each search partition is automatically replicated to provide additional processing capacity.
search request	A request that is sent to an Amazon CloudSearch domain's search endpoint to retrieve documents from the index that match particular search criteria.
search result	A document that matches a search request. Also referred to as a *search hit*.
secret access key	A key that is used in conjunction with the access key ID (p. 220) to cryptographically sign programmatic AWS requests. Signing a request identifies the sender and prevents the request from being altered. You can generate secret access keys for your AWS account, individual IAM users, and temporary sessions.
security group	A named set of allowed inbound network connections for an instance. (Security groups in Amazon VPC also include support for outbound connections.) Each security group consists of a list of protocols, ports, and IP address ranges. A security group can apply to multiple instances, and multiple groups can regulate a single instance.
sender	The person or entity sending an email message.
Sender ID	A Microsoft-controlled version of SPF. An email authentication and anti-spoofing system. For more information about Sender ID, go to http://wikipedia.org/wiki/Sender_ID.
sending limits	The sending quota (p. 245) and maximum send rate (p. 237) that are associated with every Amazon SES account.
sending quota	The maximum number of emails that you can send using Amazon SES in a 24-hour period.
service endpoint	See endpoint.
service health dashboard	A web page showing up-to-the-minute information about AWS service availability. The dashboard is located at http://status.aws.amazon.com.
SHA	Secure Hash Algorithm. SHA1 is an earlier version of the algorithm, which AWS has deprecated in favor of SHA256.

shared AMI	An Amazon Machine Image (p. 222) that a developer builds and makes available for others to use.
shutdown action	A predefined bootstrap action that launches a script that executes a series of commands in parallel before terminating the job flow.
signature	Refers to a *digital signature*, which is a mathematical way to confirm the authenticity of a digital message. AWS uses signatures to authenticate the requests you send to our web services. For more information, to http://aws.amazon.com/security.
SIGNATURE file	A file you copy to the root directory of your storage device. The file contains a job ID, manifest file, and a signature.
Simple Mail Transfer Protocol	See SMTP.
Single-AZ DB Instance	A standard (non-Multi-AZ) DB instance (p. 228) that is deployed in one Availability Zone (p. 224), without a standby replica in another Availability Zone. See Also Multi-AZ deployment.
single-valued attribute	An attribute with one value.
sloppy phrase search	A search for a phrase that specifies how close the terms must be to one another to be considered a match.
SMTP	Simple Mail Transfer Protocol. The standard that is used to exchange email messages between internet hosts for the purpose of routing and delivery.
snapshot	Amazon Elastic Block Store (p. 222) creates *snapshots* or backups of your volumes and stores them in Amazon S3. You can use these snapshots as the starting point for new Amazon EBS volumes or to protect your data for long-term durability.
soft bounce	A temporary email delivery failure such as "mailbox full."
software VPN	A software appliance-based VPN connection over the Internet.
sort enabled	An Amazon CloudSearch index field option that enables a field to be used to sort the search results.
source/destination checking	A security measure to verify that an EC2 instance is the origin of all traffic that it sends and the ultimate destination of all traffic that it receives, that is, that the instance is not relaying traffic. Source/destination checking is enabled by default. For instances that function as gateways, such as VPC NAT instances, source/destination checking must be disabled.
spam	Unsolicited bulk email.
spamtrap	An email address that is set up by an anti-spam (p. 246) entity, not for correspondence, but to monitor unsolicited email. This is also called a *honeypot*.
SPF	Sender Policy Framework. A standard for authenticating email. See Also http://www.openspf.org.
Spot Instance	A type of EC2 instance (p. 229) that you can bid on to take advantage of unused Amazon EC2 capacity.
Spot Price	The price for a Spot Instance (p. 246) at any given time. If your maximum price exceeds the current price and your restrictions are met, Amazon EC2 launches instances on your behalf.
stack	AWS CloudFormation: A collection of AWS resources you create and delete as a single unit.

	AWS OpsWorks: A set of instances you manage collectively, typically because they have a common purpose such as serving PHP applications. A stack serves as a container and handles tasks that apply to the group of instances as a whole, such as managing applications and cookbooks.
station	A place at an AWS facility where we transfer your AWS Import/Export data on to, or off of, your storage device.
statistic	One of five functions of the values submitted for a given sampling period (p. 244). These functions are "Maximum", "Minimum," "Sum," "Average," and "Sample-Count."
stem	The common root or substring shared by a set of related words.
stemming	The process of mapping related words to a common stem. This enables matching on variants of a word. For example, a search for "horse" could return matches for horses, horseback, and horsing, as well as horse. Amazon CloudSearch supports both dictionary based and algorithmic stemming.
step	A single function applied to the data in a job flow (p. 234). The sum of all steps comprises a job flow.
step type	The type of work done in a step. There are a limited number of step types, such as moving data from Amazon S3 to Amazon EC2 or from Amazon EC2 to Amazon S3.
sticky session	A feature of the load balancer that binds a user's session to a specific application instance so that all requests coming from the user during the session are sent to the same application instance. By contrast, a load balancer defaults to route each request independently to the application instance with the smallest load.
stopping	The process of filtering stop words from an index or search request.
stopword	A word that is not indexed and is automatically filtered out of search requests because it is either insignificant or so common that including it would result in too many matches to be useful. Stop words are language-specific.
streaming	Amazon EMR: A utility that comes with Hadoop that enables you to develop MapReduce executables in languages other than Java.
	CloudFront: The ability to use a media file in real time—as it is transmitted in a steady stream from a server.
streaming distribution	A special kind of distribution (p. 229) that serves streamed media files using a Real Time Messaging Protocol (RTMP) connection.
string-to-sign	Before you calculate an HMAC signature, you first assemble the required components in a canonical order. The pre-encrypted string is the string-to-sign.
structured query	Search criteria specified using the Amazon CloudSearch structured query language. You use the structured query language to construct compound queries that use advanced search options and combine multiple search criteria using Boolean operators.
subnet	A segment of the IP address range of a VPC (p. 250) that EC2 instances can be attached to. You can create subnets to group instances according to security and operational needs.
Subscription button	An HTML-coded button that enables an easy way to charge customers a recurring fee.

suggester	Specifies an Amazon CloudSearch index field you want to use to get autocomplete suggestions and options that can enable fuzzy matches and control how suggestions are sorted.
suggestions	Documents that contain a match for the partial search string in the field designated by the suggester. Amazon CloudSearch suggestions include the document IDs and field values for each matching document. To be a match, the string must match the contents of the field starting from the beginning of the field.
supported AMI	An Amazon Machine Image (p. 222) similar to a paid AMI (p. 239), except that the owner charges for additional software or a service that customers use with their own AMIs.
synchronous bounce	A type of bounce (p. 225) that occurs while the email servers of the sender (p. 245) and receiver (p. 242) are actively communicating.
synonym	A word that is the same or nearly the same as an indexed word and that should produce the same results when specified in a search request. For example, a search for "Rocky Four" or "Rocky 4" should return the fourth *Rocky* movie. This can be done by designating that `four` and `4` are synonyms for `IV`. Synonyms are language-specific.
system Qualifications	The set of Qualifications (p. 241) that represent a worker's (p. 251) history and reputation. The Mechanical Turk system assigns these Qualifications to each worker, and continuously updates the values as they use the system.

T

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

tag	Metadata (consisting of up to 10 key/value pairs) that you can define and assign to Amazon EC2 resources.
tagging	Also called *labeling*. A way to format return path (p. 244) email addresses so that you can specify a different return path for each recipient of a message. Tagging enables you to support VERP (p. 250). For example, if Andrew manages a mailing list, he can use the return paths andrew+recipient1@example.net and andrew+recipient2@example.net so that he can determine which email bounced.
task node	An EC2 instance (p. 229) that runs Hadoop map and reduce tasks, but does not store data. Task nodes are managed by the master node (p. 237), which assigns Hadoop tasks to nodes and monitors their status. While a job flow is running you can increase and decrease the number of task nodes. Because they don't store data and can be added and removed from a job flow, you can use task nodes to manage the EC2 instance capacity your job flow uses, increasing capacity to handle peak loads and decreasing it later. Task nodes only run a TaskTracker Hadoop daemon.
tebibyte	A contraction of tera binary byte, a tebibyte is 2^40 bytes or 1,099,511,627,776 bytes. A terabyte is 10^12 or 1,000,000,000,000 bytes.
template format version	The version of an AWS CloudFormation template design that determines the available features. If you omit the `AWSTemplateFormatVersion` section from your template, AWS CloudFormation assumes the most recent format version.

template validation	The process of confirming the use of JSON (p. 235) code in an AWS CloudFormation template. You can validate any AWS CloudFormation template using the `cfn-validate-template` command.
throttling	The means by which Amazon SES rejects your attempts to send email because you have exceeded your sending limits (p. 245).
time series data	Data provided as part of a metric. The time value is assumed to be when the value occurred. A metric is the fundamental concept for CloudWatch and represents a time-ordered set of data points. You publish metric data points into CloudWatch and later retrieve statistics about those data points as a time-series ordered data set.
time stamp	A date/time string in ISO 8601 format.
TLS	See Transport Layer Security.
tokenization	The process of splitting a stream of text into separate tokens on detectable boundaries such as whitespace and hyphens.
topic	A communication channel to send messages and subscribe to notifications. It provides an access point for publishers and subscribers to communicate with each other.
Transport Layer Security	A cryptographic protocol that provides security for communication over the Internet. Its predecessor is Secure Sockets Layer (SSL).
trusted signers	AWS accounts that the CloudFront distribution owner has given permission to create signed URLs for a distribution's content.
tuning	Selecting the number and type of AMIs (p. 222) to run a Hadoop job flow most efficiently.
tunnel	A route for transmission of private network traffic that uses the Internet to connect nodes in the private network. The tunnel uses encryption and secure protocols such as PPTP to prevent the traffic from being intercepted as it passes through public routing nodes.

U

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

unbounded	The number of potential occurrences is not limited by a set number. This value is often used when defining a data type that is a list (for example, `maxOccurs="unbounded"`), in Web Services Description Language (p. 251).
unit	Standard measurement for the values submitted to CloudWatch as metric data. Units include Seconds, Percent, Bytes, Bits, Count, Bytes/Second, Bits/Second, Count/Second, and None.
usage report	An AWS report giving details of your usage of a particular AWS service. You can generate and download usage reports from http://aws.amazon.com/usage-reports/.
user	A person or application under an account (p. 220) that needs to make API calls to AWS products. Each user has a unique name within the AWS account, and a set of security credentials not shared with other users. These credentials are separate

	from the AWS account's security credentials. Each user is associated with one and only one AWS account.
user token	A customer credential returned to your product during product activation (p. 221). The user token is a long, encoded string that AWS uses to identify the customer. Your product provides the customer's user token in each request for Amazon S3 the product makes on behalf of the customer. Every user token generated for a particular customer differs from the others because the creation time is one of the items making up the token value.

V

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

validation	See template validation.
value	Instances of attributes (p. 223) for an item, such as cells in a spreadsheet. An attribute might have multiple values.
value-add	The amount you charge each customer on top of the cost of the AWS services they used.
Variable Envelope Return Path	See VERP.
verification	The process of confirming that you own an email address or a domain so that you can send emails from or to it.
VERP	Variable Envelope Return Path. A way in which email sending applications can match bounced emails with the undeliverable address that caused the bounce by using a different return path (p. 244) for each recipient. VERP is typically used for mailing lists. With VERP, the recipient's email address is embedded in the address of the return path, which is where bounced emails are returned. This makes it possible to automate the processing of bounced emails without having to open the bounce messages, which may vary in content.
versioning	Every object in Amazon S3 has a key and a version ID. Objects with the same key, but different version IDs can be stored in the same bucket. Versioning is enabled at the bucket layer using PUT Bucket versioning.
virtualization	Allows multiple guest virtual machines (VM) to run on a host operating system. Guest VMs can run on one or more levels above the host hardware, depending on the type of virtualization. See Also PV virtualization, HVM virtualization.
virtual private cloud	See VPC.
virtual private gateway	See VPG.
visibility timeout	The period of time that a message is invisible to the rest of your application after an application component gets it from the queue. During the visibility timeout, the component that received the message usually processes it, and then deletes it from the queue. This prevents multiple components from processing the same message.
VPC	Virtual private cloud. An elastic network populated by infrastructure, platform, and application services that share common security and interconnection.

VPG	Virtual private gateway. The Amazon side of a VPN connection that maintains connectivity. The internal interfaces of the virtual private gateway connect to your VPC via the VPN attachment and the external interfaces connect to the VPN connection, which leads to the customer gateway.
VPN CloudHub	See AWS VPN CloudHub.
VPN connection	Although VPN connection is a general term, we specifically mean the IPsec connection between a VPC (p. 250) and some other network, such as a corporate data center, home network, or co-location facility.

W

Numbers and Symbols (p. 220) | A (p. 220) | B (p. 224) | C (p. 225) | D (p. 227) | E (p. 229) | F (p. 231) | G (p. 232) | H (p. 232) | I (p. 233) | J (p. 234) | K (p. 235) | L (p. 235) | M (p. 236) | N (p. 238) | O (p. 238) | P (p. 239) | Q (p. 241) | R (p. 242) | S (p. 244) | T (p. 248) | U (p. 249) | V (p. 250) | W (p. 251) | X, Y, Z (p. 251)

Web Services Description Language	A language used to describe the actions that a web service can perform, along with the syntax of action requests and responses. Your SOAP or other toolkit interprets a WSDL file to provide your application access to the actions provided by the web service. For most toolkits, your application calls a service action using routines and classes provided or generated by the toolkit.
worker	A person who performs the tasks specified by a Requester (p. 243) in a Human Intelligence Task (p. 233).

X, Y, Z

No entries