**Using Versioning**

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures.

In one bucket, for example, you can have two objects with the same key, but different version IDs, such as photo.gif (version 111111) and photo.gif (version 121212).



Versioning-enabled buckets enable you to recover objects from accidental deletion or overwrite. For example:

* If you delete an object, instead of removing it permanently, Amazon S3 inserts a delete marker, which becomes the current object version. You can always restore the previous version. For more information, see [Deleting Object Versions](http://docs.aws.amazon.com/AmazonS3/latest/dev/DeletingObjectVersions.html).
* If you overwrite an object, it results in a new object version in the bucket. You can always restore the previous version.

**Important**

If you have an object expiration lifecycle policy in your non-versioned bucket and you want to maintain the same permanent delete behavior when you enable versioning, you must add a noncurrent expiration policy. The noncurrent expiration lifecycle policy will manage the deletes of the noncurrent object versions in the version-enabled bucket. (A version-enabled bucket maintains one current and zero or more noncurrent object versions.) For more information, see [How Do I Create a Lifecycle Policy for an S3 Bucket?](http://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-lifecycle.html) in the *Amazon Simple Storage Service Console User Guide*.

Buckets can be in one of three states: unversioned (the default), versioning-enabled, or versioning-suspended.

**Important**

Once you version-enable a bucket, it can never return to an unversioned state. You can, however, suspend versioning on that bucket.

The versioning state applies to all (never some) of the objects in that bucket. The first time you enable a bucket for versioning, objects in it are thereafter always versioned and given a unique version ID. Note the following:

* Objects stored in your bucket before you set the versioning state have a version ID of null. When you enable versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles the objects in future requests. For more information, see [Managing Objects in a Versioning-Enabled Bucket](http://docs.aws.amazon.com/AmazonS3/latest/dev/manage-objects-versioned-bucket.html).
* The bucket owner (or any user with appropriate permissions) can suspend versioning to stop accruing object versions. When you suspend versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles objects in future requests. For more information, see [Managing Objects in a Versioning-Suspended Bucket](http://docs.aws.amazon.com/AmazonS3/latest/dev/VersionSuspendedBehavior.html).

# Deleting Object Versions

You can delete object versions whenever you want. In addition, you can also define lifecycle configuration rules for objects that have a well-defined lifecycle to request Amazon S3 to expire current object versions or permanently remove noncurrent object versions. When your bucket is version-enabled or versioning is suspended, the lifecycle configuration actions work as follows:

* The Expiration action applies to the current object version and instead of deleting the current object version, Amazon S3 retains the current version as a noncurrent version by adding a delete marker, which then becomes the current version.
* The NoncurrentVersionExpiration action applies to noncurrent object versions, and Amazon S3 permanently removes these object versions. You cannot recover permanently removed objects.

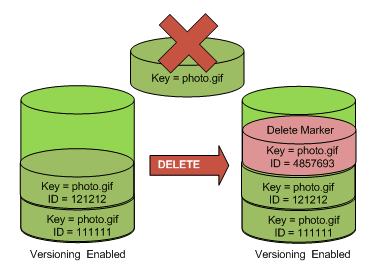
For more information, see [Object Lifecycle Management](http://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html).

A DELETE request has the following use cases:

* When versioning is enabled, a simple DELETE cannot permanently delete an object.

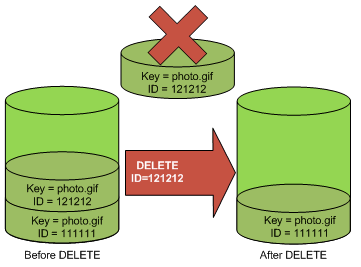
Instead, Amazon S3 inserts a delete marker in the bucket, and that marker becomes the current version of the object with a new ID. When you try to GET an object whose current version is a delete marker, Amazon S3 behaves as though the object has been deleted (even though it has not been erased) and returns a 404 error.

The following figure shows that a simple DELETE does not actually remove the specified object. Instead, Amazon S3 inserts a delete marker.



* To permanently delete versioned objects, you must use DELETE Object versionId.

The following figure shows that deleting a specified object version permanently removes that object.



## Using the Console

For instructions see, [How Do I See the Versions of an S3 Object?](http://docs.aws.amazon.com/AmazonS3/latest/user-guide/view-object-versions.html) in the Amazon Simple Storage Service Console User Guide.

## Using the AWS SDKs

For examples of uploading objects using the AWS SDKs for Java, .NET, and PHP, see [Deleting Objects](http://docs.aws.amazon.com/AmazonS3/latest/dev/DeletingObjects.html). The examples for uploading objects in nonversioned and versioning-enabled buckets are the same, although in the case of versioning-enabled buckets, Amazon S3 assigns a version number. Otherwise, the version number is null.

# Object Lifecycle Management

Lifecycle configuration enables you to specify the lifecycle management of objects in a bucket. The configuration is a set of one or more rules, where each rule defines an action for Amazon S3 to apply to a group of objects. These actions can be classified as follows:

* **Transition actions** – In which you define when objects transition to another [storage class](http://docs.aws.amazon.com/AmazonS3/latest/dev/storage-class-intro.html). For example, you may choose to transition objects to the STANDARD\_IA (IA, for infrequent access) storage class 30 days after creation, or archive objects to the GLACIER storage class one year after creation.

* **Expiration actions** – In which you specify when the objects expire. Then Amazon S3 deletes the expired objects on your behalf.

For more information about lifecycle rules, see [Lifecycle Configuration Elements](http://docs.aws.amazon.com/AmazonS3/latest/dev/intro-lifecycle-rules.html).

## When Should I Use Lifecycle Configuration for Objects?

You can define lifecycle configuration rules for objects that have a well-defined lifecycle. For example:

* If you are uploading periodic logs to your bucket, your application might need these logs for a week or a month after creation, and after that you might want to delete them.
* Some documents are frequently accessed for a limited period of time. After that, these documents are less frequently accessed. Over time, you might not need real-time access to these objects, but your organization or regulations might require you to archive them for a longer period and then optionally delete them later.
* You might also upload some types of data to Amazon S3 primarily for archival purposes, for example digital media archives, financial and healthcare records, raw genomics sequence data, long-term database backups, and data that must be retained for regulatory compliance.

Using lifecycle configuration rules, you can direct S3 to tier down the storage classes, archive, or delete the objects during their lifecycle.

## How Do I Configure a Lifecycle?

A lifecycle configuration, an XML file, comprises a set of rules with predefined actions that you want Amazon S3 to perform on objects during their lifetime.

Amazon S3 provides a set of API operations that you use to manage lifecycle configuration on a bucket. Amazon S3 stores the configuration as a lifecycle subresource that is attached to your bucket.

[PUT Bucket lifecycle](http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTlifecycle.html)

[GET Bucket lifecycle](http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketGETlifecycle.html)

[DELETE Bucket lifecycle](http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketDELETElifecycle.html)

You can also configure the lifecycle by using the Amazon S3 console or programmatically by using the AWS SDK wrapper libraries, and if you need to you can also make the REST API calls directly. For more information, see [Setting Lifecycle Configuration On a Bucket](http://docs.aws.amazon.com/AmazonS3/latest/dev/how-to-set-lifecycle-configuration-intro.html).

<http://docs.aws.amazon.com/AmazonS3/latest/dev/manage-lifecycle-using-java.html>

**Manage Object Lifecycle Using the AWS SDK for Java**

You can use the AWS SDK for Java to manage lifecycle configuration on a bucket. For more information about managing lifecycle configuration, see [Object Lifecycle Management](http://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html).

The example code in this topic does the following:

* Add lifecycle configuration with the following two rules:
  + A rule that applies to objects with the glacierobjects/ key name prefix. The rule specifies a transition action that directs Amazon S3 to transition these objects to the GLACIER storage class. Because the number of days specified is 0, the objects become eligible for archival immediately.
  + A rule that applies to objects having tags with tag key archive and value true. The rule specifies two transition actions, directing Amazon S3 to first transition objects to the STANDARD\_IA (IA, for infrequent access) storage class 30 days after creation, and then transition to the GLACIER storage class 365 days after creation. The rule also specifies expiration action directing Amazon S3 to delete these objects 3650 days after creation.
* Retrieves the lifecycle configuration.
* Updates the configuration by adding another rule that applies to objects with the YearlyDocuments/ key name prefix. The expiration action in this rule directs Amazon S3 to delete these objects 3650 days after creation.

**Note**

When you add a lifecycle configuration to a bucket, any existing lifecycle configuration is replaced. To update existing lifecycle configuration, you must first retrieve the existing lifecycle configuration, make changes and then add the revised lifecycle configuration to the bucket.

**Example Java Code Example**

The following Java code example provides a complete code listing that adds, updates, and deletes a lifecycle configuration to a bucket. You need to update the code and provide your bucket name to which the code can add the example lifecycle configuration.

For instructions on how to create and test a working sample, see [Testing the Java Code Examples](http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingTheMPDotJavaAPI.html#TestingJavaSamples).

import java.io.IOException;

import java.util.Arrays;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;

import com.amazonaws.services.s3.AmazonS3Client;

import com.amazonaws.services.s3.model.AmazonS3Exception;

import com.amazonaws.services.s3.model.BucketLifecycleConfiguration;

import com.amazonaws.services.s3.model.BucketLifecycleConfiguration.Transition;

import com.amazonaws.services.s3.model.StorageClass;

import com.amazonaws.services.s3.model.Tag;

import com.amazonaws.services.s3.model.lifecycle.LifecycleAndOperator;

import com.amazonaws.services.s3.model.lifecycle.LifecycleFilter;

import com.amazonaws.services.s3.model.lifecycle.LifecyclePrefixPredicate;

import com.amazonaws.services.s3.model.lifecycle.LifecycleTagPredicate;

public class LifecycleConfiguration {

public static String bucketName = "*\*\*\* Provide bucket name \*\*\**";

public static AmazonS3Client s3Client;

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

s3Client = new AmazonS3Client(new ProfileCredentialsProvider());

try {

BucketLifecycleConfiguration.Rule rule1 =

new BucketLifecycleConfiguration.Rule()

.withId("Archive immediately rule")

.withFilter(new LifecycleFilter(

new LifecyclePrefixPredicate("glacierobjects/")))

.addTransition(new Transition()

.withDays(0)

.withStorageClass(StorageClass.Glacier))

.withStatus(BucketLifecycleConfiguration.ENABLED.toString());

BucketLifecycleConfiguration.Rule rule2 =

new BucketLifecycleConfiguration.Rule()

.withId("Archive and then delete rule")

.withFilter(new LifecycleFilter(

new LifecycleTagPredicate(new Tag("archive", "true"))))

.addTransition(new Transition()

.withDays(30)

.withStorageClass(StorageClass.StandardInfrequentAccess))

.addTransition(new Transition()

.withDays(365)

.withStorageClass(StorageClass.Glacier))

.withExpirationInDays(3650)

.withStatus(BucketLifecycleConfiguration.ENABLED.toString());

BucketLifecycleConfiguration configuration =

new BucketLifecycleConfiguration()

.withRules(Arrays.asList(rule1, rule2));

// Save configuration.

s3Client.setBucketLifecycleConfiguration(bucketName, configuration);

// Retrieve configuration.

configuration = s3Client.getBucketLifecycleConfiguration(bucketName);

// Add a new rule.

configuration.getRules().add(

new BucketLifecycleConfiguration.Rule()

.withId("NewRule")

.withFilter(new LifecycleFilter(

new LifecycleAndOperator(Arrays.asList(

new LifecyclePrefixPredicate("YearlyDocuments/"),

new LifecycleTagPredicate(new Tag("expire\_after", "ten\_years"))))))

.withExpirationInDays(3650)

.withStatus(BucketLifecycleConfiguration.

ENABLED.toString())

);

// Save configuration.

s3Client.setBucketLifecycleConfiguration(bucketName, configuration);

// Retrieve configuration.

configuration = s3Client.getBucketLifecycleConfiguration(bucketName);

// Verify there are now three rules.

configuration = s3Client.getBucketLifecycleConfiguration(bucketName);

System.out.format("Expected # of rules = 3; found: %s\n",

configuration.getRules().size());

System.out.println("Deleting lifecycle configuration. Next, we verify deletion.");

// Delete configuration.

s3Client.deleteBucketLifecycleConfiguration(bucketName);

// Retrieve nonexistent configuration.

configuration = s3Client.getBucketLifecycleConfiguration(bucketName);

String s = (configuration == null) ? "No configuration found." : "Configuration found.";

System.out.println(s);

} catch (AmazonS3Exception amazonS3Exception) {

System.out.format("An Amazon S3 error occurred. Exception: %s", amazonS3Exception.toString());

} catch (Exception ex) {

System.out.format("Exception: %s", ex.toString());

}

}

}

<http://docs.aws.amazon.com/AmazonS3/latest/dev/how-to-set-lifecycle-configuration-intro.html>

<http://docs.aws.amazon.com/AmazonS3/latest/dev/lifecycle-configuration-examples.html>

<http://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-lifecycle.html>

<http://docs.aws.amazon.com/AmazonS3/latest/dev/intro-lifecycle-rules.html>

<http://docs.aws.amazon.com/AmazonS3/latest/dev/manage-lifecycle-using-java.html>