

Amazon Simple Storage Service

Developer Guide

API Version 2006-03-01



Amazon Simple Storage Service: Developer Guide

Copyright © 2008 Amazon Web Services LLC or its affiliates. All rights reserved.

Table of Contents

What's New	1
Welcome to Amazon S3	3
Introduction to Amazon S3	6
Core Concepts	7
Components of Amazon S3	7
Operations	8
Amazon S3 Application Programming Interfaces (API)	8
Amazon S3 Data Consistency Model	9
Paying for Amazon S3	10
Using Amazon S3	11
Working with Amazon S3 Components	11
Working with Amazon S3 Buckets	11
Bucket Restrictions and Limitations	12
Bucket Configuration Options	13
Buckets and Access Control	14
Billing and Reporting of Buckets	14
Working with Amazon S3 Objects	14
Keys	15
Listing Keys	15
Common List Request Parameters	16
Common List Response Elements	16
Iterating Through Multi-Page Results	18
Listing Keys Hierarchically using Prefix and Delimiter	19
Metadata	19
Getting Objects	20
Standard Downloads	20
Chunked and Resumable Downloads	21
Authentication and Access Control	21
Authentication	22
Access Control Lists	25
Query String Authentication	30
Request Routing	31
Request Redirection and the REST API	31
DNS Considerations	36
Performance Optimization	36
TCP Window Scaling	36
TCP Selective Acknowledgement	37
Using Amazon DevPay with Amazon S3	37
Amazon S3 Customer Data Isolation	37
Amazon DevPay Token Mechanism	38
Amazon S3 and Amazon DevPay Authentication	39
Amazon S3 Bucket Limitation	39
Amazon S3 and Amazon DevPay Process	40
Additional Information	40
Working with Errors	40
Amazon S3 Error Best Practices	40
Error Response	41
Error Code	41
Error Message	42
Further Details	42
List of Error Codes	42
Server Access Logging	46
Server Access Logging Configuration API	47

Delivery of Server Access Logs	49
Server Access Log Format	50
Setting Up Server Access Logging	53
Using the REST API	57
Common REST API Elements	58
The REST Error Response	59
Authenticating REST Requests	60
Setting Access Policy with REST	70
Virtual Hosting of Buckets	71
Request Redirection and the REST API	74
Browser-Based Uploads Using POST	77
Introduction	77
HTML Forms	78
Examples	86
POST with Adobe Flash	94
Operations on the Service	95
GET Operation	95
Operations on Buckets	96
PUT Bucket	97
GET Bucket	98
GET Bucket Location	100
DELETE Bucket	100
POST Object	101
Operations on Objects	107
PUT Object	107
GET Object	109
HEAD Object	114
DELETE Object	114
Using the SOAP API	116
Common SOAP API Elements	116
The SOAP Error Response	117
Authenticating SOAP Requests	117
Setting Access Policy with SOAP	118
Operations on the Service	119
ListAllMyBuckets	119
Operations on Buckets	120
CreateBucket	121
DeleteBucket	121
ListBucket	122
GetBucketAccessControlPolicy	124
SetBucketAccessControlPolicy	124
GetBucketLoggingStatus	125
SetBucketLoggingStatus	126
Operations on Objects	127
PutObjectInline	127
PutObject	129
GetObject	131
GetObjectExtended	136
DeleteObject	136
GetObjectAccessControlPolicy	137
SetObjectAccessControlPolicy	138
Using BitTorrent™ with Amazon S3	140
How You are Charged for BitTorrent Delivery	140
Using BitTorrent to Retrieve Objects Stored in Amazon S3	141
Publishing Content Using Amazon S3 and BitTorrent	141
Glossary	143
Document Conventions	144
Index	147

What's New

This What's New is associated with the 2006-03-01 release of Amazon S3. This guide was last updated on April 09, 2008.

The following table describes the important changes since the last release of the Amazon S3 Developer Guide.

Change	Description	Release Date
Logging Changes	Amazon S3 now enables you to automatically grant access to logs within a bucket to users other than the bucket owner. For more information, see Setting Up Server Access Logging .	7 April 2008
TCP Window Scaling	Amazon S3 now supports TCP window scaling and TCP selective acknowledgement which enables you to optimize network performance. For more information, see Performance Optimization .	3 March 2008
Chunked and Resumable Downloads	The guide was updated to describe how to perform chunked and resumable downloads. For more information, see Chunked and Resumable Downloads .	11 January 2008
HTTP POST changes	The redirect field was changed to success_action_redirect and the success_action_status field was added. For more information, see Browser-Based Uploads Using POST .	31 December 2007
DevPay	Amazon DevPay enables you to charge customers for using your Amazon S3 product through Amazon's authentication and billing infrastructure. You can charge any amount for your product including usage charges (storage, transactions, and bandwidth), monthly fixed charges, and a one-time charge. For more information, see Using Amazon DevPay with Amazon S3 .	18 December 2007
HTTP POST	Amazon S3 now supports browser-based uploads using POST, which allows your users to upload content directly to Amazon S3. For more information, see Browser-Based Uploads Using POST .	17 December 2007
Restructuring and Various Edits	The introductory sections of the document were restructured and numerous edits were made based on customer input from	17 December 2007

Change	Description	Release Date
	the Feedback link and forums.	
Location Constraints	Amazon S3 now supports location constraints, which allow you to specify where to store data. For more information, see Location Selection .	15 October 2007
Support for Redirects	If DNS information for a bucket is not propagated throughout the Internet, clients will receive a 307 redirect. If you attempt use a path-style request to access an object within a bucket that was created using <code><CreateBucketConfiguration></code> , you will receive a permanent 301 redirect. For more information on redirects, so you can optimize your code, see Location Selection .	15 October 2007
Bucket Location	Amazon S3 supports a new operation for getting the location of a bucket. For more information, see GET Bucket Location .	15 October 2007
New Authentication Section	The authentication section was rewritten to clarify questions that appeared in the forums. For more information, see Authentication and Access Control .	10 September 2007
Feedback	You can now provide feedback comments on any topic in the HTML version of this guide. To provide feedback, simply click the Feedback link at the top of the page.	10 September 2007
Minor Edits	Minor edits were made throughout the document to clarify issues that appeared in the forums and to improve overall document quality.	10 September 2007
Amazon DevPay	Amazon DevPay is a new Amazon service that enables you to charge customers for use of your Amazon S3 product through the Amazon authentication and billing infrastructure. For more information, see Using Amazon DevPay with Amazon S3 .	10 September 2007
New Bucket Limit	In addition to the 100 bucket limit associated with your AWS account, each of your customers can have up to 100 buckets for each Amazon DevPay product that you sell. For more information, see Using Amazon DevPay with Amazon S3 .	10 September 2007

Welcome to Amazon S3

Topics

- [Audience](#)
- [How This Guide Is Organized](#)
- [Related Resources](#)

Thank you for your interest in Amazon S3.

This section describes who should read this guide, how the guide is organized, and other resources related to Amazon S3.

Amazon S3 will occasionally be referred to within this guide as simply "S3"; all copyrights and legal protections still apply.

We hope you find the service to be easy-to-use, reliable, and inexpensive. If you want to provide feedback to the Amazon S3 development team, please post a message to the Amazon S3 Developer Forum.

Audience

This guide describes the Amazon S3 interfaces and functionality in detail and is intended for developers who are building application and services that need to store and retrieve any amount of data, at any time, from anywhere on the web.

Required Knowledge and Skills

Use of this guide assumes you are familiar with the following:

- XML (See [W3 Schools XML Tutorial](#))
- Basic understanding of web services (See [W3 Schools Web Services Tutorial](#))
- A programming language for consuming a web service and any related tools

You should also have read the *Amazon S3 Getting Started Guide*.

How This Guide Is Organized

This guide is organized into several major sections described in the table below.

Information	Relevant Sections
General information about Amazon S3	<ul style="list-style-type: none">Introduction to Amazon S3
Conceptual information about Amazon S3	<ul style="list-style-type: none">Core Concepts
Information about using Amazon S3	<ul style="list-style-type: none">Using Amazon S3
API Information	<ul style="list-style-type: none">Using the REST APIUsing the SOAP API
Information about BitTorrent	<ul style="list-style-type: none">Using BitTorrent™ with Amazon S3
Typographic and symbol conventions	<ul style="list-style-type: none">Document Conventions

Each section is written to stand on its own, so you should be able to look up the information you need and go back to work. However, you can also read through the major sections sequentially to get in-depth knowledge about the Amazon S3.

Related Resources

The table below lists related resources that you'll find useful as you work with this service.

Resource	Description
<i>Amazon S3 Getting Started Guide</i>	The Getting Started Guide provides a quick tutorial of the service based on a simple use case. Examples and instructions for Java, Perl, PHP, C#, Python, and Ruby are included.
<i>Amazon S3 Developer Guide</i>	The Developer Guide (which you are reading) provides a detailed discussion of the service. It includes an overview, programming reference, and API reference.
<i>Amazon S3 Release Notes</i>	The Release Notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
AWS Developer Resource Center	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.

Amazon Simple Storage Service Developer Guide

Related Resources

Resource	Description
Amazon S3 product information	The primary web page for information about Amazon S3.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
E-mail address for questions related to your AWS account: <webservices@amazon.com>	This e-mail address is <i>only</i> for account questions. For technical questions, use the Discussion Forums.

Introduction to Amazon S3

Topics

- [Overview of Amazon S3](#)
- [Service Features](#)

This introduction to Amazon S3 is intended to give you a detailed summary of this web service. After reading this section, you should have a good idea of what it offers and how it can fit in with your business.

Overview of Amazon S3

Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers.

Amazon S3 has a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of websites. The service aims to maximize benefits of scale and to pass those benefits to developers.

Service Features

Amazon S3 is intentionally built with a minimal feature set that focuses on simplicity and robustness. The following are some of features of the Amazon S3 service:

- Write, read, and delete objects from 1 byte to 5 gigabytes in size with accompanying metadata. There is no fixed limit on the number of objects you can store.
- A straightforward flat object store model, where each object is stored and retrieved using a unique developer-assigned key
- Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public and rights can be granted to specific users.
- Standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.

Core Concepts

Topics

- [Components of Amazon S3](#)
- [Operations](#)
- [Amazon S3 Application Programming Interfaces \(API\)](#)
- [Amazon S3 Data Consistency Model](#)
- [Paying for Amazon S3](#)

This chapter describes core concepts you should understand before using Amazon S3.



Note

If you have not read the Getting Started Guide, we recommend you review it first: It contains a tutorial-style overview of Amazon S3 concepts and functionality and a walkthrough of sample code.

Components of Amazon S3

This section describes the components of Amazon S3:

- [Buckets](#)
- [Objects](#)
- [Keys](#)



Note

If you already read the Amazon S3 Getting Started Guide (available in the [Amazon S3 Resource Center](#)), this section will not contain any new information.

Buckets

A bucket is simply a container for objects stored in Amazon S3. Every object is contained within a bucket. For example, if the object named `photos/puppy.jpg` is stored in the `johnsmith` bucket, then it is addressable using the URL `http://johnsmith.s3.amazonaws.com/photos/puppy.jpg`

Buckets serve several purposes: they organize the Amazon S3 namespace at the highest level, they identify the account responsible for storage and data transfer charges, they play a role in access control, and they serve as the unit of aggregation for usage reporting.

For more information about buckets, see [Working with Amazon S3 Buckets](#).

Objects

Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The data portion is opaque to Amazon S3. The metadata is a set of name-value pairs that describe the object. These include some default metadata such as the date last modified, and standard HTTP metadata such as Content-Type. The developer can also specify custom metadata at the time the Object is stored.

Keys

A key is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. Since a bucket and key together uniquely identify each object, Amazon S3 can be thought of as a basic data map between "bucket + key" and the object itself. Every object in Amazon S3 can be uniquely addressed through the combination of the Service endpoint, bucket name, and key, as in `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.

Operations

Amazon S3 offers APIs in REST and SOAP. The most common operations you'll execute through the API include the following:

- *Create a Bucket*: Create and name your own bucket in which to store your objects.
- *Write an Object*: Store data by creating or overwriting an object. When you write an object, you specify a unique key in the namespace of your bucket. This is also a good time to specify any access control you want on the object.
- *Read an Object*: Read data back. You can choose to download the data via HTTP or BitTorrent.
- *Deleting an Object*: Delete some of your data.
- *Listing Keys*: List the keys contained in one of your buckets. You can filter the key list based on a prefix.

Details on this and all other functionality are described in detail later in this guide.

Amazon S3 Application Programming Interfaces (API)

The Amazon S3 architecture is designed to be programming language-neutral, using our supported interfaces to store and retrieve objects.

Amazon S3 currently provides a REST and a SOAP interface. They are very similar, but there are some differences. For example, in the REST interface, metadata is returned in HTTP headers. Because we

only support HTTP requests of up to 4k (not including the body), the amount of metadata you can supply is restricted.

REST Interface

The REST API is an HTTP interface to Amazon S3. Using REST, you use standard HTTP requests to create, fetch, and delete buckets and objects.

You can use any toolkit that supports HTTP to use the REST API. You can even use a browser to fetch objects, as long as they are anonymously readable.

The REST API uses the standard HTTP headers and status codes, so that standard browsers and toolkits work as expected. In some areas, we have added functionality to HTTP (for example, we added headers to support access control). In these cases, we have done our best to add the new functionality in a way that matched the style of standard HTTP usage.

SOAP Interface

The SOAP API provides a SOAP 1.1 interface using document literal encoding. The most common way to use SOAP is to download the WSDL (available at <http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl>), use a SOAP toolkit such as Apache Axis or Microsoft .NET to create bindings, and then write code that uses the bindings to call Amazon S3.

Amazon S3 Data Consistency Model

Updates to a single key are atomic. For example, if you PUT to an existing key, a subsequent read might return the old data or the updated data, but it will never write corrupted or partial data.

Amazon S3 achieves high availability by replicating data across multiple servers within Amazon's data centers. After a "success" is returned, your data is safely stored. However, information about the changes might not be replicated across Amazon S3 and you may observe the following behaviors:

- A process writes a new object to Amazon S3 and immediately attempts to read it. Until the change is fully propagated, Amazon S3 may report "key does not exist".
- A process writes a new object to Amazon S3 and immediately lists keys within its bucket. Until the change is fully propagated, the object may not appear in the list.
- A process replaces an existing object and immediately attempts to read it. Until the change is fully propagated, Amazon S3 may return the prior data.
- A process deletes an existing object and immediately attempts to read it. Until the deletion is fully propagated, Amazon S3 may return the deleted data.
- A process deletes an existing object and immediately lists keys within its bucket. Until the deletion is fully propagated, Amazon S3 may list the deleted object.



Note

Amazon S3 does not currently support object locking. If two puts are simultaneously made to the same key, the put with the latest time stamp wins. If this is an issue, you will need to build an object-locking mechanism into your application.

Updates are key-based; there is no way to make atomic updates across keys. For example, you cannot make the update of one key dependent on the update of another key unless you design this functionality into your application.

Paying for Amazon S3

Pricing for Amazon S3 is designed so that you don't have to plan for the storage requirements of your application. Most storage providers force you to purchase a pre-determined amount of storage and network transfer capacity: If you exceed that capacity, your service is shut off or you are charged high overage fees. If you do not exceed that capacity, you pay as though you used it all.

Amazon S3 charges you only for what you actually use, with no hidden fees and no overage charges. This gives developers a variable-cost service that can grow with their business while enjoying the cost advantages of Amazon's infrastructure.

Before storing anything in Amazon S3, you need to register with the service and provide a payment instrument that will be charged at the end of each month. There are no set-up fees to begin using the service. At the end of the month, your payment instrument is automatically charged for that month's usage.

For information about paying for Amazon S3 storage, visit the [AWS Resource Center](#).

Using Amazon S3

This section discusses Amazon S3 concepts that apply regardless of the API style you choose. The following topics are included:

- [Working with Amazon S3 Components](#)
- [Authentication and Access Control](#)
- [Request Routing](#)
- [Performance Optimization](#)
- [Using Amazon DevPay with Amazon S3](#)
- [Working with Errors](#)
- [Server Access Logging](#)

Working with Amazon S3 Components

The following topics describe buckets and objects:

- [Working with Amazon S3 Buckets](#)
- [Working with Amazon S3 Objects](#)



Note

The [Authentication and Access Control](#) section describes access control in detail.

Working with Amazon S3 Buckets

Topics

- [Bucket Restrictions and Limitations](#)
- [Bucket Configuration Options](#)
- [Buckets and Access Control](#)
- [Billing and Reporting of Buckets](#)

Every object stored in Amazon S3 is contained in a bucket. Buckets partition the namespace of objects stored in Amazon S3 at the top level. Within a bucket, you can use any names for your objects, but bucket names must be unique across all of Amazon S3.

Buckets are similar to Internet domain names. Just as Amazon is the only owner of the domain name Amazon.com, only one person or organization can own a bucket within Amazon S3. Once you create a uniquely named bucket in Amazon S3, you can organize and name the objects within the bucket in any way you like and the bucket will remain yours for as long as you like and as long as you have the Amazon S3 account.

The similarities between buckets and domain names is not a coincidence#there is a direct mapping between Amazon S3 buckets and subdomains of s3.amazonaws.com. Objects stored in Amazon S3 are addressable using the REST API under the domain *bucketname.s3.amazonaws.com*. For example, if the object *homepage.html* is stored in the Amazon S3 bucket *mybucket* its address would be `http://mybucket.s3.amazonaws.com/homepage.html`. For more information, see [Virtual Hosting of Buckets](#).

Bucket Restrictions and Limitations

A bucket is owned by the AWS account (identified by AWS Access Key ID) that created it. Each AWS account can own up to 100 buckets at a time. Bucket ownership is not transferable. However, if a bucket is empty, it can be deleted and its name can be reused.



Note

If you are using Amazon DevPay, each of your customers can have up to 100 buckets for each Amazon DevPay product they use. For more information, see [Using Amazon DevPay with Amazon S3](#).

Buckets have the following restrictions:

- Bucket names can only contain lowercase letters, numbers, periods (.), underscores (_), and dashes (-).
- Bucket names must start with a number or letter.
- Bucket names must be between 3 and 255 characters long.
- Bucket names cannot be in an IP address style (e.g., "192.168.5.4").

To conform with DNS requirements, we recommend following these additional guidelines when creating buckets:

- Bucket names should not contain underscores (_).
- Bucket names should be between 3 and 63 characters long.
- Bucket names should not end with a dash.
- Dashes cannot appear next to periods. For example, "my-.bucket.com" and "my.-bucket" are invalid.



Note

Buckets with names containing uppercase characters are not accessible using the virtual hosted-style request (e.g., `http://yourbucket.s3.amazonaws.com/yourobject`)

If you create a bucket using `<CreateBucketConfiguration>`, you must follow the additional guidelines.

If you create a bucket using `<CreateBucketConfiguration>`, applications that access

your bucket must be able to handle 307 redirects. For more information, see [Request Redirection and the REST API](#).

When using virtual hosted-style buckets with SSL, the SSL wildcard certificate only matches buckets that do not contain periods. To work around this, use HTTP or write your own certificate verification logic.

There is no limit to the number of objects that can be stored in a bucket and no variation in performance when using many buckets or just a few. You can store all of your objects in a single bucket or organize them across several buckets.

Buckets cannot be nested, meaning buckets cannot be created within buckets.

The high availability engineering of Amazon S3 is focused on get, put, list, and delete operations. Because bucket operations work against a centralized, global resource space, it is not appropriate to make bucket create or delete calls on the high availability code path of your application. It is better to create or delete buckets in a separate initialization or setup routine that you run less often.



Note

If your application automatically creates buckets, choose a bucket naming scheme that is unlikely to cause naming conflicts. Additionally, make sure your application has logic to choose a different bucket name if a bucket name is already taken.

Bucket Configuration Options

When creating buckets, you can take advantage of additional Amazon S3 features by attaching the `<CreateBucketConfiguration>` XML body to a PUT Bucket request. Currently, you can select a location constraint (see [Location Selection](#)).

Buckets created with `<CreateBucketConfiguration>` are subject to additional restrictions:

- You cannot make a request to a bucket created with `<CreateBucketConfiguration>` using a path-style request; you must use the virtual hosted-style request. For more information, see [Virtual Hosting of Buckets](#).
- You must follow additional bucket naming restrictions. For more information, see [Bucket Restrictions and Limitations](#).

Location Selection

You can now choose a location constraint that will affect where objects are stored within Amazon S3. You can currently specify a Europe (EU) location constraint.

Choosing a location is simple; just specify a location constraint when creating a new bucket and all objects placed within that bucket are automatically stored in the same location.



Note

If you do not specify a location constraint, Amazon S3 automatically selects a location which will be billed at the standard Amazon S3 rates.

Pricing varies based on the specified location constraint. For more information, refer to the [Amazon S3 portal page](#).

The SOAP API does not support requests that use `CreateBucketConfiguration`.

Bucket Access

To access Amazon S3 buckets and objects that were created using `CreateBucketConfiguration`, you must use the virtual hosted-style request. For example:

`http://yourbucket.s3.amazonaws.com/yourobject`

You cannot use the path-style request:

`http://s3.amazonaws.com/yourbucket/yourobject`

If you use the path-style request, you receive a permanent redirect.

Redirection

Amazon supports two types of redirects: temporary and permanent.

Temporary redirects automatically redirect users that do not have DNS information for the requested bucket. This occurs because DNS changes take time to propagate through the Internet. For example, if a user creates a bucket with a location constraint and immediately stores an object in the bucket, information about the bucket might not be distributed throughout the Internet. Because the bucket is a subdomain of `s3.amazonaws.com`, Amazon S3 redirects it to the correct Amazon S3 location.

Permanent redirects redirect users from the path-style request to the virtual hosted-style request format for buckets created using `<CreateBucketConfiguration>`. Users will be provided with the correct URL, but will not be forwarded to the correct location.

Buckets and Access Control

Each bucket has an associated access control policy. This policy governs the creation, deletion and enumeration of objects within the bucket. For more information, see [Authentication and Access Control](#).

Billing and Reporting of Buckets

Fees for object storage and network data transfer are always billed to the owner of the bucket that contains the object.

The reporting tools available at the Amazon Web Services developer portal organize your Amazon S3 usage reports by bucket.

Working with Amazon S3 Objects

Topics

- [Keys](#)
- [Metadata](#)
- [Getting Objects](#)

Amazon S3 is designed to store objects. Objects are stored in buckets and consist of a value, a key, metadata, and an access control policy.

The object value is the content that you are storing. The object value can be any sequence of bytes, but must be smaller than five gigabytes. There is no fixed limit to the number of objects you can store in Amazon S3.

The key is the handle that you assign to an object that allows you retrieve it later.

Metadata is a set of key-value pairs with which you can store information regarding the object.

The access control policy controls access to the object.

Keys

The key is the handle that you assign to an object that allows you retrieve it later. A key is a sequence of Unicode characters whose UTF-8 encoding is at most 1024 bytes long. Each object in a bucket must have a unique key.

Keys can be listed by prefix. By choosing a common prefix for the names of related keys and marking these keys with a special character that delimits hierarchy, you can use the list operation to select and browse keys hierarchically. This is similar to how files are stored in directories within a filesystem. For more information, see [Listing Keys](#).

Keys often have a suffix that describes the type of data in the object. For example, ".jpg" indicates that an object is an image. Although Amazon S3 supports key suffixes, they are not required.

Listing Keys

Amazon S3 exposes a list operation that lets you enumerate the keys contained in a bucket. Keys are selected for listing by bucket and prefix. For example, consider a bucket named 'dictionary' that contains a key for every English word. You might make a call to list all the keys in that bucket that start with the letter 'q'. List results are always returned in lexicographic (alphabetical) order.

For API independent information about composing a list request, see [Common List Request Parameters](#).

Both the SOAP and REST list operations return an XML document that contains the names of matching keys and information about the object identified by each key. This common XML response document is documented in detail. For more information, see [Common List Response Elements](#).

You can iterate through large collections of keys by making multiple, paginated, list requests. For example, an initial list request against the dictionary bucket might only retrieve information about the keys 'quack' through 'quartermaster.' But a subsequent request would retrieve 'quarters' through 'quince', and so on.

For instructions on how to correctly handle large list result sets, see [Iterating Through Multi-Page Results](#).

Groups of keys that share a prefix terminated by a special delimiter can be rolled-up by that common prefix for the purposes of listing. This allows applications to organize and browse their keys hierarchically, much like how you would organize your files into directories in a filesystem. For example, to extend the dictionary bucket to contain more than just English words, you might form keys by prefixing each word with its language and a delimiter, like "French/logiciel". Using this naming scheme and the hierarchical listing feature, you could retrieve a list of only French words. You could also browse the top-level list of available languages without having to iterate through all the lexicographically intervening keys.

For more information on this aspect of listing, see [Listing Keys Hierarchically using Prefix and Delimiter](#).

List Implementation Efficiency

List performance is not substantially affected by the total number of keys in your bucket, nor by the presence or absence of the prefix, marker, maxkeys, or delimiter arguments.

Common List Request Parameters

Both SOAP and REST list requests accept the following parameters:

- *Prefix* Restricts the response to only contain results that begin with the specified prefix. If you omit this optional argument, the value of Prefix for your query will be the empty string. In other words, the results will not be restricted by prefix.
- *Marker* This optional parameter enables pagination of large result sets. Marker specifies where in the result set to resume listing. It restricts the response to only contain results that occur alphabetically after the value of marker. To retrieve the next page of results, use the last key from the current page of results as the marker in your next request. See also NextMarker, below. If Marker is omitted, the first page of results is returned.
- *Delimiter* If this optional, Unicode string parameter is included with your request, then keys that contain the same string between the prefix and the first occurrence of the delimiter will be rolled up into a single result element in the CommonPrefixes collection. These rolled-up keys are not returned elsewhere in the response.

For example, with Prefix="USA/" and Delimiter="/", the matching keys "USA/Oregon/Salem" and "USA/Oregon/Portland" would be summarized in the response as a single "USA/Oregon" element in the CommonPrefixes collection. If an otherwise matching key does not contain the delimiter after the prefix, it appears in the Contents collection.

Each element in the CommonPrefixes collection counts as one against the MaxKeys limit. The rolled-up keys represented by each CommonPrefixes element do not.

If the Delimiter parameter is not present in your request, keys in the result set will not be rolled-up and neither the CommonPrefixes collection nor the NextMarker element will be present in the response.

- *MaxKeys* This optional argument limits the number of results returned in response to your query. Amazon S3 will return no more than this number of results, but possibly less. Even if MaxKeys is not specified, Amazon S3 will limit the number of results in the response. Check the IsTruncated flag to see if your results are incomplete. If so, use the Marker parameter to request the next page of results.

For the purpose of counting MaxKeys, a 'result' is either a key in the 'Contents' collection, or a delimited prefix in the 'CommonPrefixes' collection. So for delimiter requests, MaxKeys limits the total number of list results, not just the number of keys.

While the SOAP and REST list parameters are substantially the same, the parameter names and the mechanics of submitting the request are different. A SOAP list request is an XML document, with the parameters as elements, while a REST list request is a GET on the bucket resource, with parameters in the query-string. See the API-specific sections for details:

- [ListBucket](#)
- [GET Bucket](#)

Access Control

The list operation requires READ permission on the bucket in question. Permission to list is conferred for any value of Prefix, Marker, Delimiter and MaxKeys.

Common List Response Elements

The SOAP and REST XML list response share the same structure and element names.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Name>johnsmith</Name>
  <Prefix>photos/2006/</Prefix>
  <Marker/>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>>false</IsTruncated>
  <Contents>
    <Key>photos/2006/index.html</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"celacdafcc879d7eee54cf4e97334078"</ETag>
    <Size>1234</Size>
    <Owner>
      <ID>214153b66967d86f031c7249d1d9a80249109428335cd08f1cdc487b4566cb1b</ID>
      <DisplayName>John Smith</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <CommonPrefixes>
    <Prefix>photos/2006/January/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

ListBucketResult is the root element of the list response document. To make the list response self-describing, ListBucketResult echoes back the list request parameters that generated it. ListBucketResult also contains the following elements:

- *IsTruncated*

A flag that indicates whether or not all results of your query were returned in this response. If your results were truncated, you can make a follow-up paginated request using the Marker parameter to retrieve the rest of the results.

- *NextMarker*

A convenience element, useful when paginating with delimiters. The value of NextMarker, if present, is the largest (alphabetically) of all key names and all CommonPrefixes prefixes in the response. If the

If the `IsTruncated` flag is set, request the next page of results by setting `Marker` to `NextMarker`. This element is only present in the response if the `Delimiter` parameter was sent with the request.

The `Contents` Element (of type `ListEntry`) contains information about each key that is part of the list results.

- *Key*

The object's key.

- *LastModified*

The time that the object was placed into Amazon S3.

- *ETag*

The object's entity tag is an opaque string used to quickly check an object for changes. With high probability, the object data associated with a key is unchanged if and only if the entity tag is unchanged. Entity tags are useful in conditional gets.

- *Size*

The number of bytes of object data stored under this key. Size does not include metadata or the size of the key.

- *Owner*

This element represents the identity of the principal who created the object. It is only present if you have permission to view it. See the 'Access Control' discussion, below.

- *StorageClass*

Always has the value `STANDARD`

The `CommonPrefixes` element may be present when you make a list request with the `delimiter` parameter. Each element in this collection represents a group of keys that share a common prefix terminated by the specified delimiter. To expand the list of keys under this prefix, make a new list request formed by substituting the value of the `CommonPrefixes/Prefix` response element for the `Prefix` request parameter.

- *Prefix*

The shared common prefix.

Access Control

The `Owner` element is only present in a given `ListEntry` element if you have `READ_ACP` permission on the object in question, or if you own the containing bucket. Otherwise, it is omitted.

Iterating Through Multi-Page Results

As buckets can contain a virtually unlimited number of keys, the complete results of a list query can be extremely large. To manage large result sets, Amazon S3 uses pagination to split them into multiple responses. The following pseudo-code procedure demonstrates how to iteratively fetch an exhaustive list of results, given a prefix, marker and delimiter:

Example

```
function exhaustiveList(bucket, prefix, marker, delimiter) :
    do {
        result = AmazonS3.list(bucket, prefix, marker, delimiter);
        // ... work with incremental list results ...

        marker = max(result.Contents.Keys, result.CommonPrefixes.Prefixes)
        // or more conveniently, when delimiter != null
        // marker = result.NextMarker;
    }
    while (result.IsTruncated);
```

Listing Keys Hierarchically using Prefix and Delimiter

The Prefix and Delimiter parameters limit the kind of results returned by a list operation. Prefix limits results to only those keys that begin with the specified prefix, and Delimiter causes list to roll-up all keys that share a common prefix into a single summary list result.

The purpose of the prefix and delimiter parameters is to allow you to organize, and then browse, your keys hierarchically. To do this, first pick a delimiter for your bucket, say '/', that doesn't occur in any of your anticipated key names. Next, construct your key names by concatenating all containing levels of the hierarchy, separating each level with the delimiter.

For example, if you were storing information about cities, you might naturally organize them by continent, then by country, then by province or state. Since these names don't usually contain punctuation, you might select '/' as the delimiter. In that case, you would name your keys like so:

- Europe/France/Aquitaine/Bordeaux
- North America/Canada/Quebec/Montreal
- North America/USA/California/San Francisco
- North America/USA/Washington/Seattle
- ... and so on.

If you stored data for every city in the world in this manner, it would become awkward to manage a flat key namespace. But, by using the Prefix and Delimiter parameters with the list operation, you can list using the hierarchy you've built into your data. For example, to list all the cities in California, set Delimiter='/' and Prefix='/North America/USA/California/'. To list all the provinces in Canada for which you have data, set Delimiter='/' and Prefix='North America/Canada/'

A list request with a delimiter lets you browse your hierarchy at just one level, skipping over and summarizing the (possibly millions of) keys nested at deeper levels.

Metadata

Each Amazon S3 object has a set of key-value pairs with which it is associated. There are two kinds of metadata: system metadata, and user metadata.

System metadata is used and is sometimes processed by Amazon S3. System metadata behavior depends on which API (REST or SOAP) you are using.

User metadata entries are specified by you. Amazon S3 does not interpret this metadata#it simply stores it and passes it back when you ask for it. Metadata keys and values can be any length, but must conform to US-ASCII when using REST and UTF-8 when using SOAP or browser-based uploads through POST.



Note

For more information about metadata encodings, go to sections 2 and 4.2 of <http://www.ietf.org/rfc/rfc2616.txt>.

Metadata Size

For both REST and SOAP requests to Amazon S3, user metadata size is limited to 2k bytes for the total length of all values and keys.

Metadata Interoperability

In REST, user metadata keys must begin with "x-amz-meta-" to distinguish them as custom HTTP headers. When this metadata is retrieved via SOAP, the x-amz-meta- prefix is removed. Similarly, metadata stored via SOAP will have x-amz-meta- added as a prefix when it is retrieved via REST, except when the metadata fits an HTTP standard header (e.g., "Content-Type" metadata).

When metadata is retrieved through the REST API, Amazon S3 combines headers that have the same name (ignoring case) into a comma-delimited list. If some metadata contains unprintable characters, it is not returned. Instead, the "x-amz-missing-meta" header is returned with a value of the number of the unprintable metadata entries.

Getting Objects

You get objects from Amazon S3 using the GET operation. This operation returns the object directly from Amazon S3.

Standard Downloads

The following is an example of a REST GET request:

```
GET /Nelson HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

which returns the following:

```
HTTP/1.1 200 OK
x-amz-id-2: j5ULAWpFbJQJpukUsZ4tfXVOjVZExLtEyNTvY5feC+hHIegsn5p578JLTVpkFrpL
x-amz-request-id: BE39A20848A0D52B
Date: Wed, 01 Mar 2006 12:00:00 GMT
x-amz-meta-family: Muntz
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
```



```
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"  
Content-Type: text/plain  
Content-Length: 5  
Connection: close  
Server: AmazonS3
```

HA-HA

Chunked and Resumable Downloads

To provide GET flexibility, Amazon S3 supports chunked and resumable downloads.

This allows you to download part of an object stored in Amazon S3 so you can break large downloads into smaller chunks or design your applications to recover from failed downloads.

Select from the following:

- For information about using resumable downloads with the REST API, see [Chunked and Resumable Downloads](#).
- For information about using resumable downloads with the SOAP API, see [Chunked and Resumable Downloads](#).

Authentication and Access Control

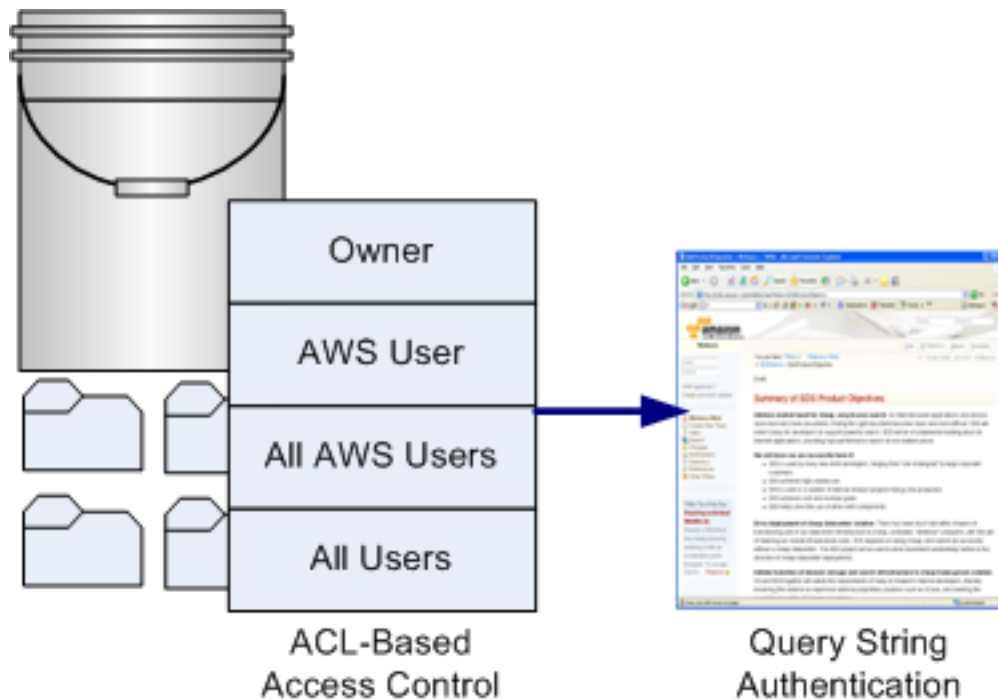
Topics

- [Authentication](#)
- [Access Control Lists](#)
- [Query String Authentication](#)

Authentication is the process of verifying the identity of a user or service trying to access an Amazon Web Services (AWS) product. Access Control defines who can access objects and buckets within Amazon S3 and the type of access (e.g., READ, WRITE, and so on). Authentication combined with access control prevents unauthorized users from accessing your data, modifying your data, deleting your data, or using your AWS account for services that cost you money.

Every interaction with Amazon S3 is authenticated or anonymous. When you sign up for an AWS account, you are provided with an AWS Access Key ID and a Secret Access Key. When you perform a request with Amazon S3, you assemble the request, perform a hash on the request using your Secret Access Key, attach the Signature (hash) to the request, and forward it to Amazon S3. Amazon S3 verifies the Signature is a valid hash of the request and, if authenticated, processes the request.

To allow selected users to access objects or buckets in your Amazon S3 account, you can use access control lists (ACLs) or query string authentication.



ACLs allow you grant access to specific AWS users, all AWS users, or any user through anonymous access. When granting access to a specific AWS user, the user must have an Amazon account and must be signed up for AWS and Amazon S3. This will enable the user to access any allowed buckets or objects using his AWS Access Key ID and Secret Access Key. When you grant access to all AWS users, any AWS user will be able to access allowed buckets or objects using an AWS Access Key ID and Secret Access Key. When you grant anonymous access, any user will be able to access allowed buckets or objects by omitting the AWS Access Key ID and Signature from a request.

Any user that is granted access to an object or bucket can construct an HTTP URL that can be used to access that object or bucket through the query string authentication mechanism. This HTTP URL can be distributed to any user with a web client or embedded in a web page.



Note

All HTTP queries have an expiration parameter that allows you to set how long the query will be valid. For example, you can configure a web page graphic to expire after a very long period of time or a software download to only last for 24 hours.

Authentication

When you create an AWS account, AWS assigns your AWS access key identifiers, a pair of related credentials:

- Access Key ID (a 20-character, alphanumeric string). For example: 022QF06E7MXBSH9DHM02
- Secret Access Key (a 40-character string). For example:
kWcrIUX5JEDGM/LtmEENI/aVmYvHNif5zB+d9+ct



Important

Your Secret Access Key is a secret and should be known only by you and AWS. It is important to keep it confidential to protect your account. Never include it in your requests

to AWS and never e-mail it to anyone. Do not share it outside your organization, even if an inquiry appears to come from AWS or Amazon.com. No one who legitimately represents Amazon will ever ask you for your Secret Access Key.

The Access Key ID uniquely identifies an AWS account. You include it in AWS service requests to identify yourself as the sender of the request.

To prove that you are the owner of the account making the request, you must include a signature. For all requests, you calculate the signature with your Secret Access Key. AWS uses the Access Key ID in the request to look up your Secret Access Key and then calculates a signature with the key. If the calculated signature matches the signature you sent, the request is considered authentic. Otherwise, the request fails authentication and is not processed.

Viewing Your Credentials

Your Access Key ID and Secret Access Key are displayed when you create your AWS account. They are not e-mailed to you. If you need to see them again, you can view them at any time from your AWS account.

To view your AWS access identifiers

1. Go to the Amazon Web Services web site at <http://aws.amazon.com>.
2. Point to **Your Web Services Account to display a list of options**.
3. Click **View Access Key Identifiers and log in to your AWS account**.

Your Access Key ID and Secret Access Key are displayed on the resulting AWS Access Identifiers page.

Using HMAC-SHA1 Signatures

When accessing Amazon S3 using REST and SOAP, you must provide the following items so the request can be authenticated:

Request Elements

- **AWS Access Key Id**—your AWS account is identified by your Access Key ID, which AWS uses to look up your Secret Access Key.
- **Signature**—each request must contain a valid request signature, or the request is rejected. A request signature is calculated using your Secret Access Key, which is a shared secret known only to you and AWS.
- **Time stamp**—each request must contain the date and time the request was created, represented as a string in UTC. The format of the value of this parameter is API-specific.
- **Date**—each request must contain the time stamp of the request. Depending on the API you're using, you can provide an expiration date and time for the request instead of or in addition to the time stamp. See the authentication topic for the particular API to determine what the API requires.

Below are the general steps for authenticating requests to AWS. It is assumed you have already created an AWS account and received an Access Key ID and Secret Access Key.

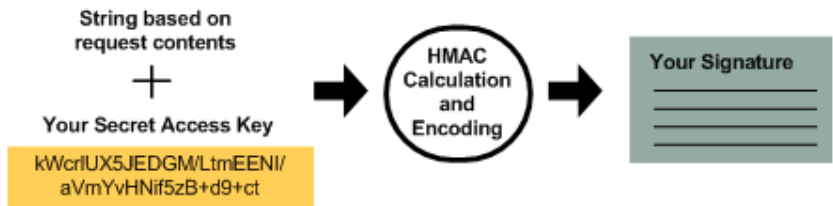
You perform the first three steps.

You

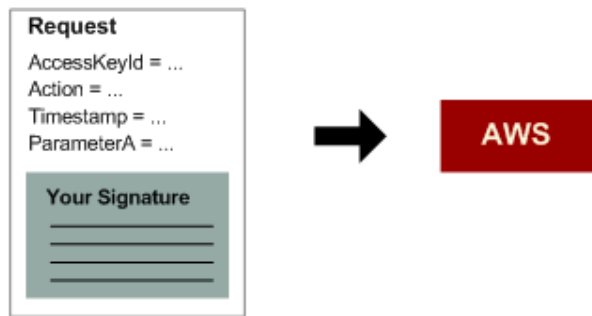
1 Create a request:

```
Request
AccessKeyId = ...
Action = ...
Timestamp = ...
ParameterA = ...
```

2 Create an HMAC-SHA1 signature:

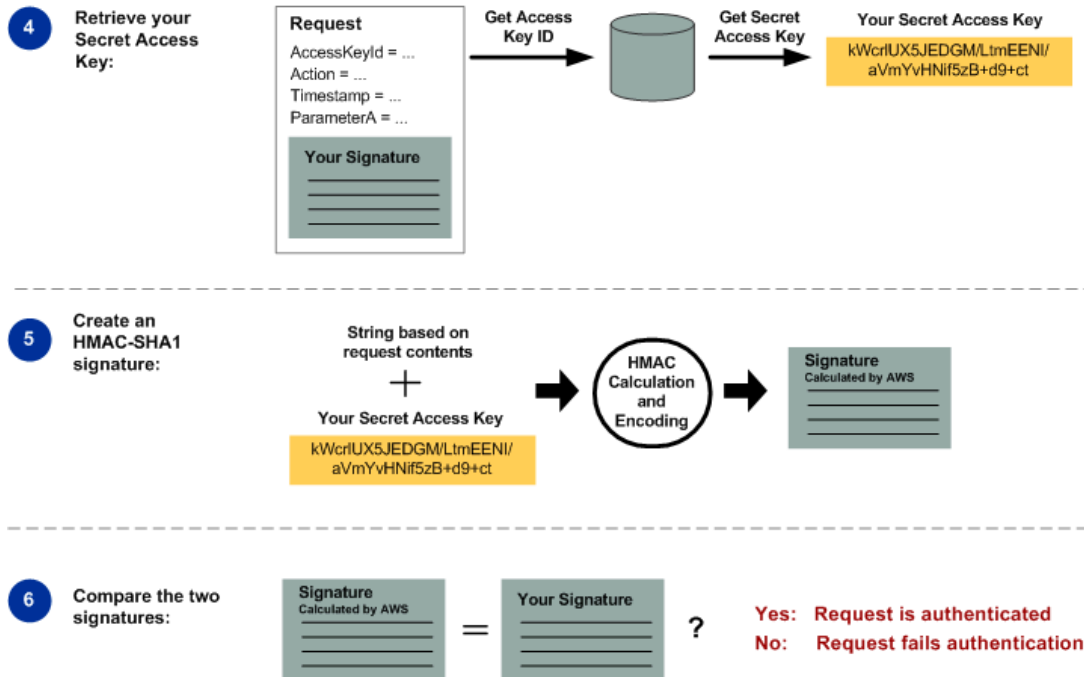


3 Send the request and signature to AWS:



1. Construct a request to AWS.
2. Calculate a keyed-hash message authentication code (HMAC) signature using your Secret Access Key
3. Include the signature and your Access Key ID in the request, and then send the request to AWS. AWS performs the next three steps.

AWS



4. AWS uses the Access Key ID to look up your Secret Access Key.
5. AWS generates a signature from the request data and the Secret Access Key using the same algorithm you used to calculate the signature you sent in the request.
6. If the signature generated by AWS matches the one you sent in the request, the request is considered authentic. If the comparison fails, the request is discarded, and AWS returns an error response.

Detailed Authentication Information

For detailed information about REST and SOAP authentication, see [Authenticating REST Requests](#) and [Authenticating SOAP Requests](#).

Using Base64 Encoding

HMAC request signatures must be Base64 encoded. Base64 encoding converts the signature into a simple ASCII string that can be attached to the request.

For examples of Base64 encoding, refer to the Amazon S3 code samples.

Access Control Lists

Topics

- [Grantees](#)
- [Permissions](#)
- [Using ACLs](#)

Each bucket and object in Amazon S3 has an ACL that defines its access control policy. When a request is made, Amazon S3 authenticates the request using its standard authentication procedure and then checks the ACL to verify sender was granted access to the bucket or object. If the sender is approved, the request proceeds. Otherwise, Amazon S3 returns an error.

An ACL is a list of grants. A grant consists of one grantee and one permission. ACLs only grant permissions; they do not deny them.



Note

Bucket and object ACLs are completely independent; an object does not inherit the ACL from its bucket. For example, if you create a bucket and grant write access to another user, you will not be able to access the user's objects unless the user explicitly grants access. This also applies if you grant anonymous write access to a bucket. Only the user "anonymous" will be able to access objects the user created unless permission is explicitly granted to the bucket owner.



Important

We highly recommend that you do not grant the anonymous group write access to your buckets as you will have no control over the objects others can store and their associated charges. For more information, see [Grantees](#) and [Permissions](#).

Grantees

There are five types of grantees that can access a bucket or object within Amazon S3. These include:

- Owner
- User by Email
- User by Canonical Representation
- AWS User Group
- Anonymous Group

Owner

Every bucket and object in Amazon S3 has an owner, the user that created the bucket or object. The owner of a bucket or object cannot be changed. However, if the object is overwritten by another user (deleted and rewritten), the new object will have a new owner.



Note

Even the owner is subject to the ACL. For example, if an owner does not have READ access to an object, the owner cannot read that object. However, the owner of an object always has write access to the access control policy (WRITE_ACP) and can change the ACL to read the object.

User by Email

You can grant access to buckets and objects within your Amazon S3 account to anyone with an Amazon Web Services account. Any users that you grant access will be able to access buckets and objects using their AWS Access Key IDs and Secret Access Keys.

The following is the XML format for granting access to a user through an Amazon customer email address:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail">

  <EmailAddress>chriscustomer@email.com</EmailAddress>

</Grantee>
```

Email grants are internally converted to the CanonicalUser representation when you create the ACL. If the grantee changes his or her email address, it will not affect the existing Amazon S3 permissions.

Adding a grantee by email address only works if exactly one Amazon account corresponds to the specified email address. If multiple Amazon accounts are associated with the email address, an AmbiguousGrantByEmail error message is returned. This is rare but usually occurs if a user created an Amazon account in the past, forgot the password, and created another Amazon account using the same email address. If this occurs, the user should contact Amazon.com customer service to have the accounts merged or you should grant user access specifying the CanonicalUser representation.

User by Canonical Representation

You can grant access to buckets and objects within your Amazon S3 account to anyone with an Amazon Web Services account. Any users that you grant access will be able to access buckets and objects using their AWS Access Key IDs and Secret Access Keys.



Note

To locate the CanonicalUser ID for a user, the user must perform the ListAllMyBuckets operation in his or her Amazon S3 account and copy the ID from the Owner XML object.

The following is the XML format for granting access to a user through an Amazon customer CanonicalUser ID:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">

  <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>

  <DisplayName>chriscustomer</DisplayName>

</Grantee>
```

The ID string specifies the CanonicalUser ID and must exactly match the ID of the user that you are adding. The DisplayName element is read-only. If you specify a DisplayName, it will be ignored and replaced with the name stored by Amazon.

AWS User Group

You can grant access to buckets or objects to anyone with an Amazon AWS account. Although this inherently insecure as any AWS user who is aware of the bucket or object will be able to access it, you may find this authentication method useful.

All AWS users can be specified as a grantee using the following example XML representation:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="Group">

  <URI>http://acs.amazonaws.com/groups/global/AuthenticatedUsers</URI>
```

```
</Grantee>
```

AllUsers Group

You can grant anonymous access to any Amazon S3 object or bucket. Any user will be able to access the object by omitting the AWS Key ID and Signature from a request.

AllUsers can be specified as a grantee using the following example XML representation:

```
<Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="Group">
  <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
</Grantee>
```

Permissions

The permission in a grant describes the type of access to be granted to the respective grantee. The following permissions are supported by Amazon S3:

Elements

- **READ**—when applied to a bucket, grants permission to list the bucket. When applied to an object, this grants permission to read the object data and/or metadata.
- **WRITE**—when applied to a bucket, grants permission to create, overwrite, and delete any object in the bucket. This permission is not supported for objects.
- **READ_ACP**—grants permission to read the ACL for the applicable bucket or object. The owner of a bucket or object always has this permission implicitly.
- **WRITE_ACP**—gives permission to overwrite the ACP for the applicable bucket or object. The owner of a bucket or object always has this permission implicitly. Granting this permission is equivalent to granting **FULL_CONTROL** because the grant recipient can make any changes to the ACP.
- **FULL_CONTROL**—provides **READ**, **WRITE**, **READ_ACP**, and **WRITE_ACP** permissions. It does not convey additional rights and is provide only for convenience.

Using ACLs

An ACL can contain up to 100 grants. If no ACL is provided when a bucket is created or an object written, a default ACL is created. The default ACL consists of a single grant that gives the owner (i.e., the creator) the **FULL_CONTROL** permission. If you overwrite an existing object, the ACL for the existing object is overwritten and will to default to **FULL_CONTROL** for the owner if no ACL is specified.

You can change the ACL of a resource without changing the resource itself. However, like Amazon S3 objects, there is no way to modify an existing ACL—you can only overwrite it with a new version. Therefore, to modify an ACL, read the ACL from Amazon S3, modify it locally, and write the entire updated ACL back to Amazon S3.



Note

The method of reading and writing ACLs differs depending on which API you are using. Please see the API-specific documentation for details.

Regardless of which API you are using, the XML representation of an ACL stored in Amazon S3 (and returned when the ACL is read) is the same. In the example ACL below, the owner has the default FULL_CONTROL, the "Frank" and "Jose" users both have WRITE and READ_ACP permissions, and all users have permission to READ:

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
<ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
      <DisplayName>chriscustomer</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
<ID>79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be</ID>
    <DisplayName>Frank</DisplayName>
  </Grantee>
    <Permission>WRITE</Permission>
  </Grant>
  <Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
<ID>79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be</ID>
    <DisplayName>Frank</DisplayName>
  </Grantee>
    <Permission>READ_ACP</Permission>
  </Grant>
  <Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
<ID>e019164ebb0724ff67188e243eae9ccbebdde523717cc312255d9a82498e394a</ID>
```

```
<DisplayName>Jose</DisplayName>
</Grantee>
<Permission>WRITE</Permission>
</Grant>
<Grant>
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
<ID>e019164ebb0724ff67188e243eae9ccbcbdde523717cc312255d9a82498e394a</ID>
  <DisplayName>Jose</DisplayName>
  </Grantee>
  <Permission>READ_ACP</Permission>
</Grant>
<Grant>
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="Group">
  <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
</AccessControlList>
</AccessControlPolicy>
```



Note

When you write an ACL to Amazon S3 that AmazonCustomerByEmail grantees, they will be converted to the CanonicalUser type prior to committing the ACL.

Query String Authentication

Query string authentication is useful for giving HTTP or browser access to resources that would normally require authentication.

When using query string authentication, you create a query, specify an expiration time for the query, sign it with your signature, place the data in an HTTP request, and distribute the request to a user or embed the request in a web page.

Query string authentication requests require an expiration date. You can specify any future expiration time in epoch or UNIX time (number of seconds since January 1, 1970). For example, a query URL will be similar to the following:

```
ht
tp://quotes.s3.amazonaws.com/nelson?AWSAccessKeyId=44CF9590006BF252F707&Expires=1177363698&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv4%3D
```

Request Routing

Topics

- [Request Redirection and the REST API](#)
- [DNS Considerations](#)

Programs that make requests against buckets created using the <CreateBucketConfiguration> API must support redirects. Additionally, some clients that do not respect DNS TTLs may encounter issues.

This section describes routing and DNS issues to consider when designing your service or application for use with Amazon S3.

Request Redirection and the REST API

Overview

Amazon S3 uses the Domain Name System (DNS) to route requests to facilities that can process them. This system works very effectively. However, temporary routing errors can occur.

If a request arrives at the wrong Amazon S3 location, Amazon S3 responds with a temporary redirect that tells the requestor to resend the request to a new endpoint.

If a request is incorrectly formed, Amazon S3 uses permanent redirects to provide direction on how to perform the request correctly.



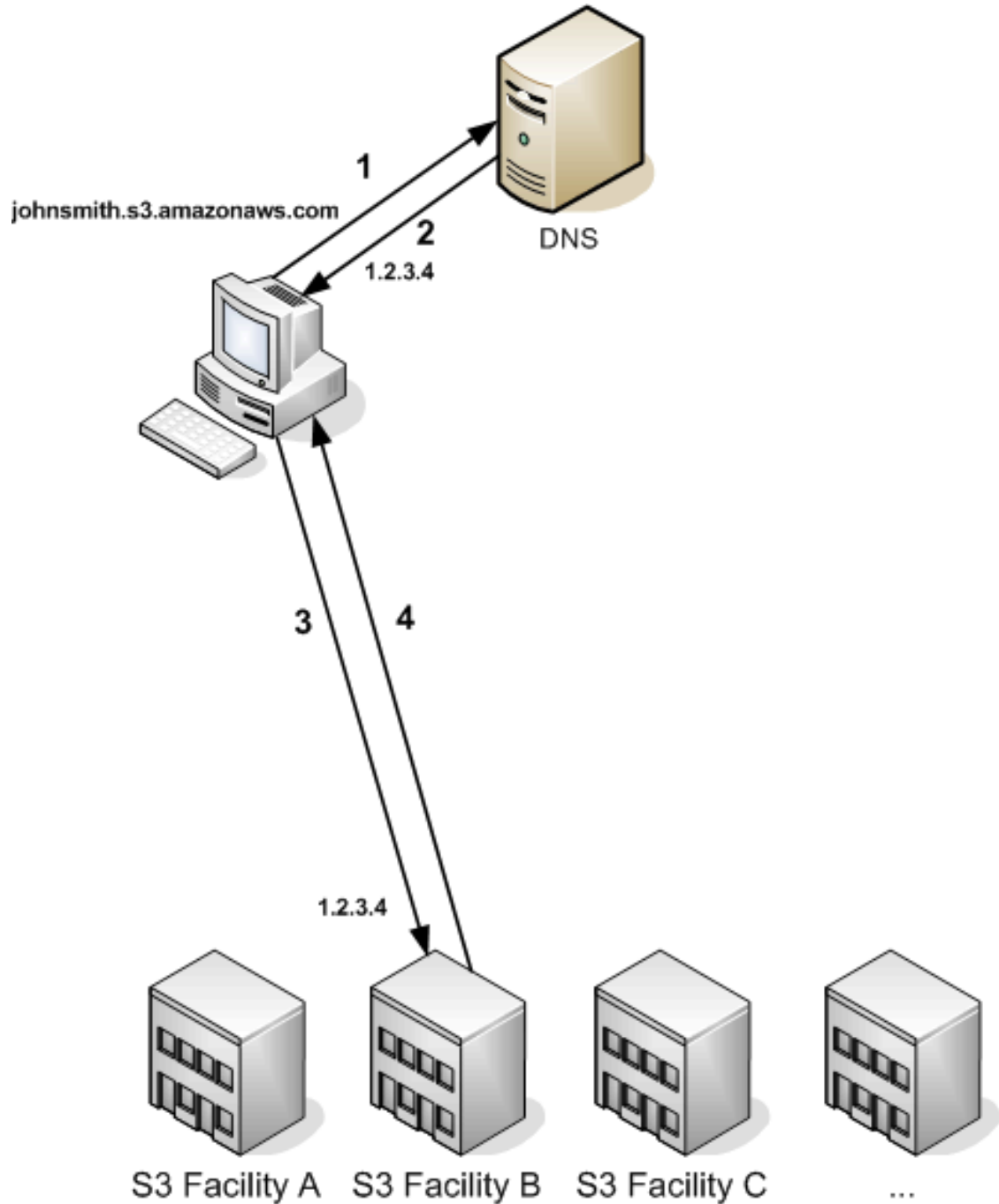
Important

Every Amazon S3 program must be designed to handle redirect responses. The only exception is for programs that work exclusively with buckets that were created without <CreateBucketConfiguration>. For more information on location constraints, see [the section called “Location Selection”](#).

DNS Routing

DNS routing routes requests to appropriate Amazon S3 facilities.

The following graphic shows an example of DNS routing.



Step	Action
1	The client makes a DNS request to get an object stored on Amazon S3.
2	The client receives one or more IP addresses for facilities that can process the request.
3	The client makes a request to Amazon S3 Facility B.
4	Facility B returns a copy of the object.

Temporary Request Redirection

A temporary redirect is a type of error response that signals to the requestor that he should resend his request to a different endpoint.

Due to the distributed nature of Amazon S3, requests can be temporarily routed to the wrong facility. This is most likely to occur immediately after buckets are created or deleted. For example, if you create a new bucket and immediately make a request to the bucket, you will receive a temporary redirect. After information about the bucket propagates through DNS, redirects will be rare.

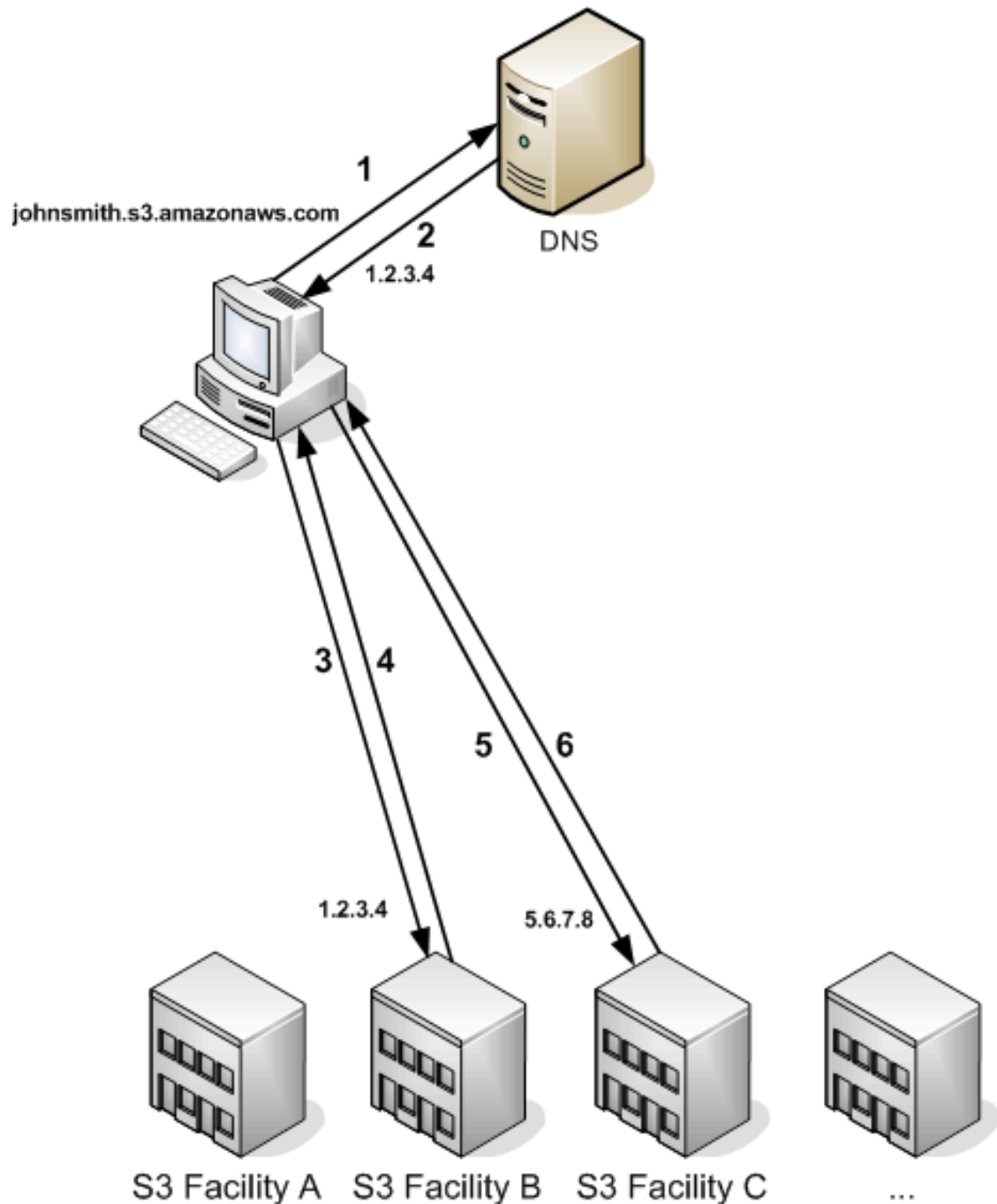
Temporary redirects contain a URI to the correct facility which you can use to immediately resend the request.



Important

Do not reuse an endpoint provided by a previous redirect response. It might appear to work (even for long periods of time), but might provide unpredictable results and will eventually fail without notice.

The following graphic shows an example of a temporary redirect.



Step	Action
1	The client makes a DNS request to get an object stored on Amazon S3.
2	The client receives one or more IP addresses for facilities that can process the request.
3	The client makes a request to Amazon S3 Facility B.
4	Facility B returns a redirect indicating the object is available from Location C.

Step	Action
5	The client resends the request to Facility C.
6	Facility C returns a copy of the object.

Permanent Request Redirection

A permanent redirect indicates that your request addressed a resource inappropriately. For example, permanent redirects occur if you use a path-style request to access a bucket that was created using `<CreateBucketConfiguration>`.

To help you find these errors during development, this type of redirect does not contain a Location HTTP header that allows you to automatically follow the request to the correct location. Consult the resulting XML error document for help using the correct Amazon S3 endpoint.

Examples

This is an example of a redirect issued by the Amazon S3 REST API.

```
HTTP/1.1 307 Temporary Redirect
Location: http://johnsmith.s3-gz4b4pa9sq.amazonaws.com/photos/puppy.jpg?rk=e2c69a31
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Fri, 12 Oct 2007 01:12:56 GMT
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the specified temporary endpoint.
  Continue to use the original request endpoint for future requests.</Message>
  <Endpoint>johnsmith.s3-gz4b4pa9sq.amazonaws.com</Endpoint>
</Error>
```

This is an example of a redirect issued by the Amazon S3 SOAP API.

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.TemporaryRedirect</Faultcode>
    <Faultstring>Please re-send this request to the specified temporary endpoint.
  </Faultstring>
  </soapenv:Fault>
</soapenv:Body>
```

```
Continue to use the original request endpoint for future re
quests.</Faultstring>

<Detail>

  <Bucket>images</Bucket>

  <Endpoint>s3-gztb4pa9sq.amazonaws.com</Endpoint>

</Detail>

</soapenv:Fault>

</soapenv:Body>
```

DNS Considerations

One of the design requirements of Amazon S3 is extremely high availability. One of the ways we meet this requirement is by updating the IP addresses associated with the Amazon S3 endpoint in DNS as needed. These changes are automatically reflected in short-lived clients, but not in some long-lived clients. Long-lived clients will need to take special action to re-resolve the Amazon S3 endpoint periodically to benefit from these changes. Please see following for specific information for various virtual machines (VMs).

- *Java*: Sun's JVM caches DNS lookups forever by default; see the "InetAddress Caching" section of [the InetAddress documentation](#) for information on how to change this behavior.
- *PHP*: The persistent PHP VM that runs in the most popular deployment configurations caches DNS lookups until the VM is restarted. See [the getHostByName PHP docs](#).

Performance Optimization

Topics

- [TCP Window Scaling](#)
- [TCP Selective Acknowledgement](#)

Amazon S3 provides new features that support high performance networking. These include TCP window scaling and selective acknowledgements.



Note

For more information on high performance tuning, go to <http://-www.psc.edu/-networking/-projects/-tcptune/>.

TCP Window Scaling

TCP window scaling allows you to improve network throughput performance between your operating system and application layer and Amazon S3 by supporting window sizes larger than 64 KB. At the start of the TCP session, a client advertises its supported receive window WSCALE factor, and Amazon S3 responds with its supported receive window WSCALE factor for the upstream direction.

Although TCP window scaling can improve performance, it can be challenging to set correctly. Make

sure to adjust settings at both the application and kernel level. For more information about TCP window scaling, refer to your operating system's documentation and go to [RFC 1323](#).

TCP Selective Acknowledgement

TCP selective acknowledgement is designed to increase recovery time after a large number of packet losses. TCP selective acknowledgement is supported by most newer operating systems, but might have to be enabled. For more information about TCP selective acknowledgements, refer to the documentation that accompanied your operating system and go to [RFC 2018](#).

Using Amazon DevPay with Amazon S3

Topics

- [Amazon S3 Customer Data Isolation](#)
- [Amazon DevPay Token Mechanism](#)
- [Amazon S3 and Amazon DevPay Authentication](#)
- [Amazon S3 Bucket Limitation](#)
- [Amazon S3 and Amazon DevPay Process](#)
- [Additional Information](#)

Amazon DevPay enables you to charge customers for using your Amazon S3 product through Amazon's authentication and billing infrastructure. You can charge any amount for your product including usage charges (storage, transactions, and bandwidth), monthly fixed charges, and a one-time charge.

Once a month, Amazon bills your customers for you. AWS then deducts the Amazon DevPay fees and pays you the difference. AWS then separately charges you for the Amazon S3 usage costs incurred by your customers.

If your customers do not pay their bills, AWS turns off access to Amazon S3 (and your product). AWS handles all payment processing.

Amazon S3 Customer Data Isolation

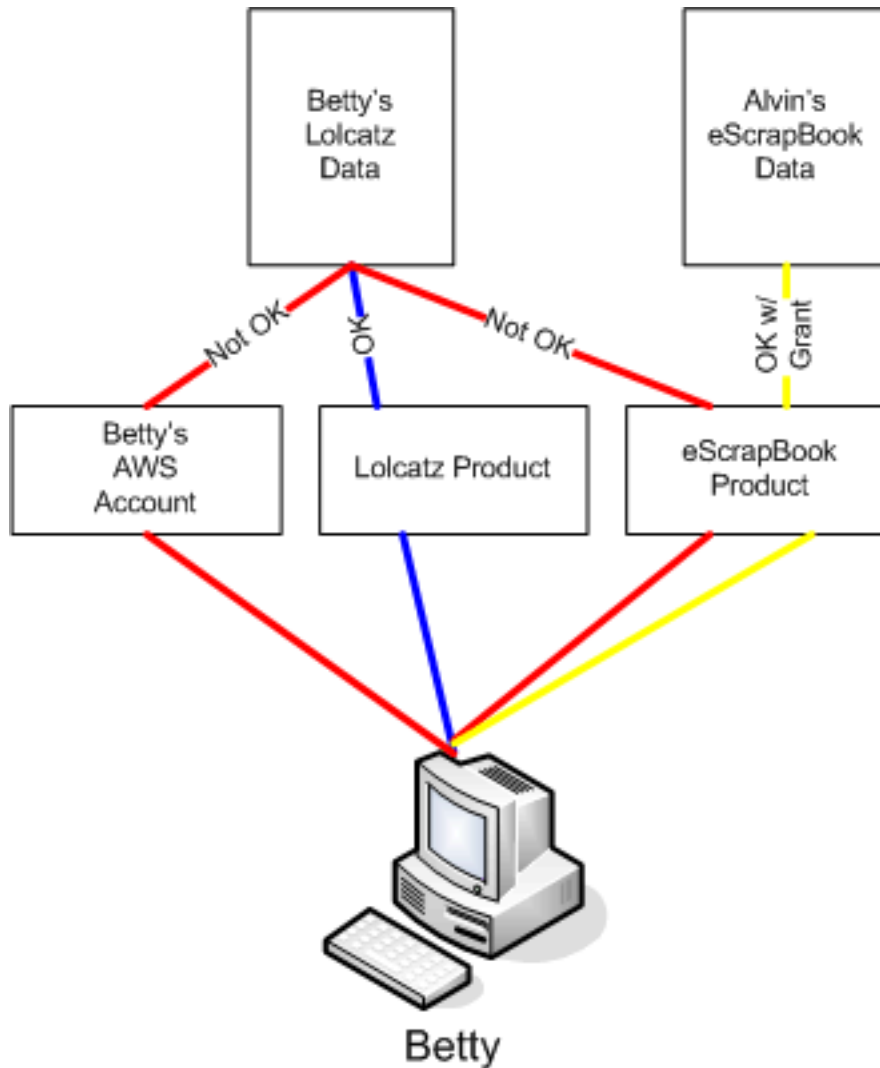
Amazon DevPay requests store and access data on behalf of the users of your product. The resources created by your application are owned by your users; unless you modify the ACL, you cannot read or modify the user's data.

Data stored by your product is isolated from other Amazon DevPay products and general Amazon S3 access. Customers that *store* data in Amazon S3 through your product can only *access* that data through your product. The data cannot be accessed through other Amazon DevPay products or through a personal AWS account.

Two users of a product can only access each other's data if your application explicitly grants access through the ACL.

Example

The following graphic shows examples of allowed, disallowed, and conditional (discretionary) data access:



For example:

- Betty can access Lolcatz data through the Lolcatz product. If she attempts to access her Lolcatz data through another product or a personal AWS account, her requests will be denied.
- Betty can access Alvin's eScrapBook data through the eScrapBook product if access is explicitly granted.

Amazon DevPay Token Mechanism

To enable you to make requests on behalf of your customers and ensure that your customers are billed for use of your application, your application must send two tokens with each request: the product token and the user token.

The product token identifies your product; you must have one product token for each Amazon DevPay product that you provide. The user token identifies a user in relationship to your product; you must have a user token for each user/product combination. For example, if you provide two products and a user subscribes to each, you must obtain a separate user token for each product.

For information on obtaining product and user tokens, refer to the *Amazon DevPay Developer Guide*.

Amazon S3 and Amazon DevPay Authentication

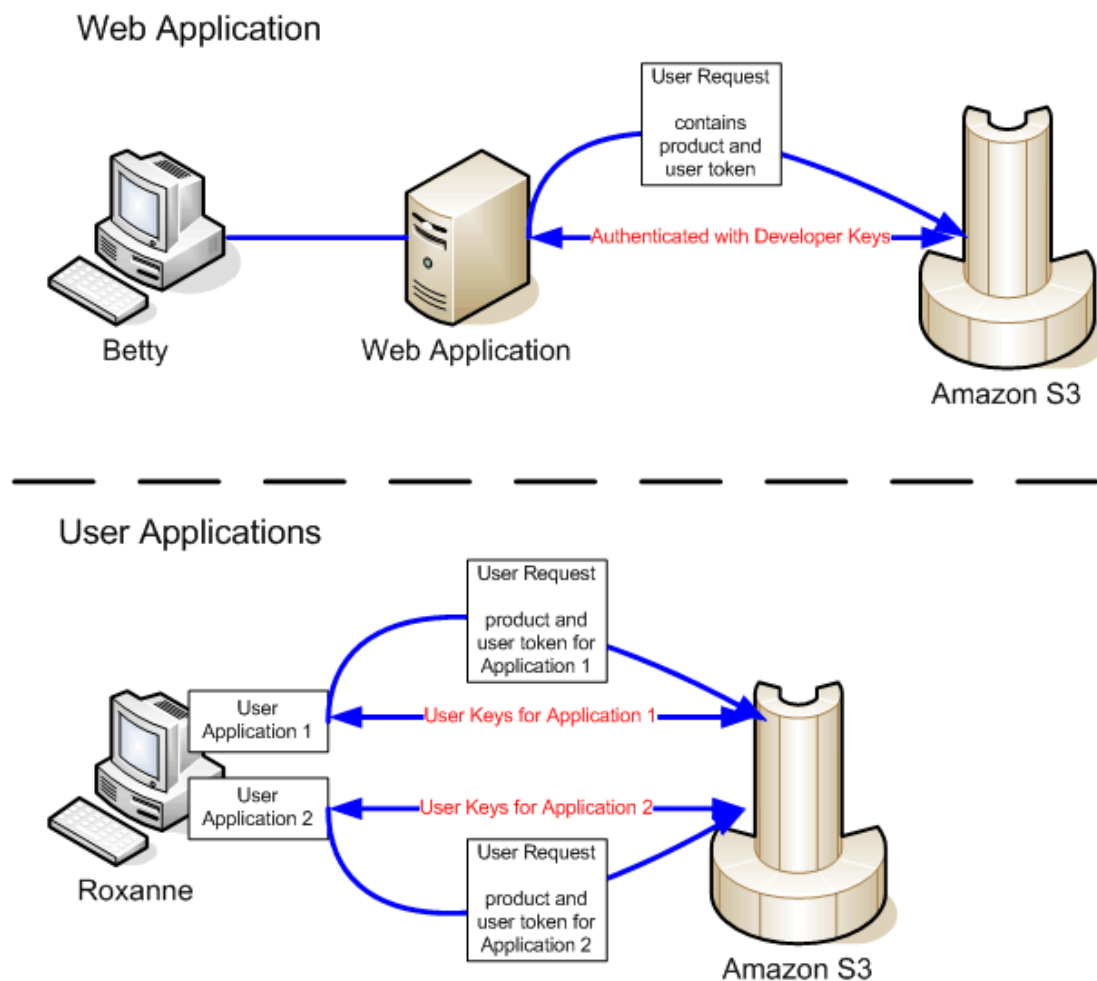
Although the token mechanism uniquely identifies a customer and product, it does not provide authentication.

Normally, your applications communicate directly with Amazon S3 using your Access Key ID and Secret Access Key. For Amazon DevPay, Amazon S3 authentication works a little differently.

If your Amazon DevPay product is a web application, you securely store the Secret Access Key on your servers and use the user token to specify the customer for which requests are being made.

However, if your Amazon S3 application is installed on your customers' computers, your application must obtain an Access Key ID and a Secret Access Key for each installation and must use those credentials when communicating with Amazon S3.

The following image shows the differences between authentication for web applications and user applications.



Amazon S3 Bucket Limitation

Each of your customers can have up to 100 buckets for each Amazon DevPay product that you sell. For example, if a customer uses three of your products, the customer can have up to 300 buckets (100 * 3) plus any buckets outside of your Amazon DevPay products (i.e., buckets in Amazon DevPay products from other developers and the customer's personal AWS account).

Amazon S3 and Amazon DevPay Process

The following provides a high-level overview of the Amazon DevPay process:

Launch Process

1	A customer signs up for your product through Amazon.
2	The customer receives an activation key.
3	The customer enters the activation key into your application.
4	Your application communicates with Amazon and obtains the user's token. If your application is installed on the user's computer, it also obtains an Access Key ID and Secret Access Key on behalf of the customer.
5	Your application provides the customer's token and the application product token when making Amazon S3 requests on behalf of the customer. If your application is installed on the customer's computer, it authenticates with the customer's credentials.
6	Amazon uses the customer's token and your product token to determine who to bill for the Amazon S3 usage.
7	Once a month, Amazon processes usage data and bills your customers according to the terms you defined.
8	AWS deducts Amazon DevPay fees and pays you the difference. AWS then separately charges you for the Amazon S3 usage costs incurred by your customers.

Additional Information

For information about using, setting up, and integrating with Amazon DevPay, refer to the *Amazon DevPay Developer Guide*.

Working with Errors

Topics

- [Amazon S3 Error Best Practices](#)
- [Error Response](#)

This section describes best practices for managing Amazon S3 errors and the format of an Amazon S3 error response.

Amazon S3 Error Best Practices

When designing an application for use with Amazon S3, it is important to handle Amazon S3 errors appropriately. This section describes issues to consider when designing your application.

Retry InternalErrors

Internal errors are errors that occur within the Amazon S3 environment.

Requests that receive an `InternalServerError` response may or may not have been processed. For example, if a `PUT` request returns `InternalServerError`, a subsequent `GET` may retrieve the old value or the updated value.

If Amazon S3 returns an `InternalServerError` response, retry the request.

Tune Application for Repeated SlowDown errors

As with any distributed system, S3 has protection mechanisms which detect intentional or unintentional resource over-consumption and react accordingly. `SlowDown` errors can occur when a high request rate triggers one of these mechanisms. Reducing your request rate will decrease or eliminate errors of this type. Generally speaking, most users will not experience these errors regularly; however, if you would like more information or are experiencing high or unexpected `SlowDown` errors, contact us via e-mail at webservices@amazon.com to discuss how to optimize your use of S3 and prevent these types of errors in your application.

Isolate Errors

Amazon S3 provides a set of error codes that are used by both the SOAP and REST API. The SOAP API returns standard Amazon S3 error codes. The REST API is designed to look like a standard HTTP server and interact with existing HTTP clients (e.g., browsers, HTTP client libraries, proxies, caches, and so on). To ensure the HTTP clients handle errors properly, we map each Amazon S3 error to an HTTP status code.

HTTP status codes are less expressive than Amazon S3 error codes and contain less information about the error. For example, the `NoSuchKey` and `NoSuchBucket` Amazon S3 errors both map to the HTTP 404 `Not Found` status code.

Although the HTTP status codes contain less information about the error, clients that understand HTTP, but not the Amazon S3 API, will usually handle the error correctly.

Therefore, when handling errors or reporting Amazon S3 errors to end users, use the Amazon S3 error code instead of the HTTP status code as it contains the most information about the error. Additionally, when debugging your application, you should also consult the human readable `<Details>` element of the XML error response.

Error Response

When an Amazon S3 request is in error, the client receives an error response. The exact format of the error response is API specific: For example, the REST error response differs from the SOAP error response. However, all error responses have the following common elements:

- [Error Code](#)
- [Error Message](#)
- [Further Details](#)
- [List of Error Codes](#)

Error Code

The error code is a string that uniquely identifies an error condition. It is meant to be read and understood by programs that detect and handle errors by type. Many error codes are common across SOAP and REST APIs, but some are API-specific. For example, `NoSuchKey` is universal, but `UnexpectedContent` can occur only in response to an invalid REST request. In all cases, SOAP fault codes carry a prefix as indicated in the table of error codes, so that a `NoSuchKey` error is actually returned in SOAP as `Client.NoSuchKey`.

Error Message

The error message contains a generic description of the error condition in English. It is intended for a human audience. Simple programs display the message directly to the end user if they encounter an error condition they don't know how or don't care to handle. Sophisticated programs with more exhaustive error handling and proper internationalization are more likely to ignore the error message.

Further Details

Many error responses contain additional structured data meant to be read and understood by a developer diagnosing programming errors. For example, if you send a Content-MD5 header with a REST PUT request that doesn't match the digest calculated on the server, you receive a BadDigest error. The error response also includes as detail elements the digest we calculated, and the digest you told us to expect. During development, you can use this information to diagnose the error. In production, a well-behaved program might include this information in its error log.

List of Error Codes

The following table lists the Amazon S3 Error Codes.

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
AccessDenied	Access Denied	403 Forbidden	Client
AccountProblem	There is a problem with your AWS account that prevents the operation from completing successfully. Please contact customer service at webservices@amazon.com .	403 Forbidden	Client
AmbiguousGrantByEmailAddress	The e-mail address you provided is associated with more than one account.	400 Bad Request	Client
BadDigest	The Content-MD5 you specified did not match what we received.	400 Bad Request	Client
BucketAlreadyExists	The requested bucket name is not available. The bucket namespace is shared by all users of the system. Please select a different name and try again.	409 Conflict	Client
BucketAlreadyOwnedByYou	Your previous request to create the named bucket succeeded and you already own it.	409 Conflict	Client
BucketNotEmpty	The bucket you tried to delete is not empty.	409 Conflict	Client
CredentialsNotSupported	This request does not support credentials.	400 Bad Request	Client
CrossLocationLoggingProhibited	Cross location logging not allowed. Buckets in one geographic location	403 Forbidden	Client

Amazon Simple Storage Service Developer Guide
Error Response

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
	cannot log information to a bucket in another location.		
EntityTooSmall	Your proposed upload is smaller than the minimum allowed object size.	400 Bad Request	Client
EntityTooLarge	Your proposed upload exceeds the maximum allowed object size.	400 Bad Request	Client
ExpiredToken	The provided token has expired.	400 Bad Request	Client
IncompleteBody	You did not provide the number of bytes specified by the Content-Length HTTP header	400 Bad Request	Client
IncorrectNumberOfFilesInPostRequest	POST requires exactly one file upload per request.	400 Bad Request	Client
InlineDataTooLarge	Inline data exceeds the maximum allowed size.	400 Bad Request	Client
InternalServerError	We encountered an internal error. Please try again.	500 Internal Server Error	Server
InvalidAccessKeyId	The AWS Access Key Id you provided does not exist in our records.	403 Forbidden	Client
InvalidAddressingHeader	You must specify the Anonymous role.	N/A	Client
InvalidArgument	Invalid Argument	400 Bad Request	Client
InvalidBucketName	The specified bucket is not valid.	400 Bad Request	Client
InvalidDigest	The Content-MD5 you specified was an invalid.	400 Bad Request	Client
InvalidLocationConstraint	The specified location constraint is not valid.	400 Bad Request	Client
InvalidPayer	All access to this object has been disabled.	403 Forbidden	Client
InvalidPolicyDocument	The content of the form does not meet the conditions specified in the policy document.	400 Bad Request	Client
InvalidRange	The requested range cannot be satisfied.	416 Requested Range Not	Client

Amazon Simple Storage Service Developer Guide
Error Response

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
		Satisfiable	
InvalidSecurity	The provided security credentials are not valid.	403 Forbidden	Client
InvalidSOAPRequest	The SOAP request body is invalid.	400 Bad Request	Client
InvalidStorageClass	The storage class you specified is not valid.	400 Bad Request	Client
InvalidTargetBucketForLogging	The target bucket for logging does not exist, is not owned by you, or does not have the appropriate grants for the log-delivery group.	400 Bad Request	Client
InvalidToken	The provided token is malformed or otherwise invalid.	400 Bad Request	Client
InvalidURI	Couldn't parse the specified URI.	400 Bad Request	Client
KeyTooLong	Your key is too long.	400 Bad Request	Client
MalformedACLError	The XML you provided was not well-formed or did not validate against our published schema.	400 Bad Request	Client
MalformedXML	The XML you provided was not well-formed or did not validate against our published schema.	400 Bad Request	Client
MaxMessageLengthExceeded	Your request was too big.	400 Bad Request	Client
MaxPostPreDataLengthExceededError	Your POST request fields preceding the upload file were too large.	400 Bad Request	Client
MetadataTooLarge	Your metadata headers exceed the maximum allowed metadata size.	400 Bad Request	Client
MethodNotAllowed	The specified method is not allowed against this resource.	405 Method Not Allowed	Client
MissingAttachment	A SOAP attachment was expected, but none were found.	N/A	Client
MissingContentLength	You must provide the Content-Length HTTP header.	411 Length Required	Client
MissingSecurityElement	The SOAP 1.1 request is missing a security element.	400 Bad Request	Client

Amazon Simple Storage Service Developer Guide
Error Response

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
MissingSecurityHeader	Your request was missing a required header.	400 Bad Request	Client
NoLoggingStatusForKey	There is no such thing as a logging status sub-resource for a key.	400 Bad Request	Client
NoSuchBucket	The specified bucket does not exist.	404 Not Found	Client
NoSuchKey	The specified key does not exist.	404 Not Found	Client
NotImplemented	A header you provided implies functionality that is not implemented.	501 Not Implemented	Server
NotSignedUp	Your account is not signed up for the Amazon S3 service. You must sign up before you can use Amazon S3. You can sign up at the following URL: http://aws.amazon.com/s3	403 Forbidden	Client
OperationAborted	A conflicting conditional operation is currently in progress against this resource. Please try again.	409 Conflict	Client
PermanentRedirect	The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint.	301 Moved Permanently	Client
PreconditionFailed	At least one of the pre-conditions you specified did not hold.	412 Precondition Failed	Client
Redirect	Temporary redirect.	307 Moved Temporarily	Client
RequestIsNotMultiPartContent	Bucket POST must be of the enclosure-type multipart/form-data.	400 Bad Request	Client
RequestTimeout	Your socket connection to the server was not read from or written to within the timeout period.	400 Bad Request	Client
RequestTimeTooSkewed	The difference between the request time and the server's time is too large.	403 Forbidden	Client
RequestTorrentOfBucketError	Requesting the torrent file of a bucket is not permitted.	400 Bad Request	Client
SignatureDoesNotMatch	The request signature we calculated does not match the signature you provided. Check your AWS Secret	403 Forbidden	Client

Error Code	Description	HTTP Status Code	SOAP Fault Code Prefix
	Access Key and signing method. For more information, see Authenticating REST Requests and Authenticating SOAP Requests for details.		
SlowDown	Please reduce your request rate.	503 Service Unavailable	Server
TemporaryRedirect	You are being redirected to the bucket while DNS updates.	307 Moved Temporarily	Client
TokenRefreshRequired	The provided token must be refreshed.	400 Bad Request	Client
TooManyBuckets	You have attempted to create more buckets than allowed.	400 Bad Request	Client
UnexpectedContent	This request does not support content.	400 Bad Request	Client
UnresolvableGrantByEmailAddress	The e-mail address you provided does not match any account on record.	400 Bad Request	Client
UserKeyMustBeSpecified	The bucket POST must contain the specified field name. If it is specified, please check the order of the fields.	400 Bad Request	Client

Server Access Logging

Topics

- [Server Access Logging Configuration API](#)
- [Delivery of Server Access Logs](#)
- [Server Access Log Format](#)
- [Setting Up Server Access Logging](#)



Important

This section describes Beta functionality that is subject to change in future releases. Please provide feedback on this functionality in the [Amazon S3 Developer Forum](#).

An Amazon S3 bucket can be configured to create access log records for the requests made against it. An access log record contains details about the request such as the request type, the resource with which

the request worked, and the time and date that the request was processed. Server access logs are useful for many applications, because they give bucket owners insight into the nature of requests made by clients not under their control.

By default, server access logs are not collected for a bucket. Learn how to enable server access logging by consulting the [Logging Configuration API](#) documentation.

Once logging is enabled for a bucket, available log records are periodically aggregated into log files and delivered to you via an Amazon S3 bucket of your choosing. For a detailed description of this process, see [Delivery of Server Access Logs](#).

For information on how to interpret the contents of log files, see [Server Access Log Format](#).

To walk through the process of enabling logging for your bucket, see [Setting Up Server Access Logging](#).



Note

There is no extra charge for enabling the server access logging feature on an Amazon S3 bucket, however any log files the system delivers to you will accrue the usual charges for storage (you can delete the log files at any time). No data transfer charges will be assessed for log file delivery, but access to the delivered log files is charged for data transfer in the usual way.

Server Access Logging Configuration API



Important

This section describes Beta functionality that is subject to change in future releases. Please provide feedback on this functionality in the [Amazon S3 Developer Forum](#).

Each Amazon S3 bucket has an associated XML sub-resource that you can read and write in order to inspect or change the logging status for that bucket. The XML schema for the bucket logging status resource is common across SOAP and REST.

The `BucketLoggingStatus` element has the following structure:

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>mylogs</TargetBucket>
    <TargetPrefix>access_log-</TargetPrefix>
    <TargetGrants>
      <Grant>
        <Grantee xm
lns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail">
          <EmailAddress>email_address</EmailAddress>
        </Grantee>
        <Permission>permission</Permission>
      </Grant>
    </TargetGrants>
  </LoggingEnabled>
</BucketLoggingStatus>
```

- *LoggingEnabled*

The presence of this element indicates that server access logging is enabled for the bucket. The absence of this element (and all nested elements) indicates that logging is disabled for the bucket.

- *TargetBucket*

This element specifies the bucket where server access logs will be delivered. You can have your logs delivered to any bucket that you own, including the same bucket that is being logged. You can also configure multiple buckets to deliver their logs to the same target bucket. In this case you should choose a different `TargetPrefix` for each source bucket so that the delivered log files can be distinguished by key.



Note

The source and the target buckets must be in the same location. For more information about bucket location constraints, see [Location Selection](#)

- *TargetPrefix*

This element lets you specify a prefix for the keys that the delivered log files will be stored under. For information on how the key name for log files is constructed, see [Delivery of Server Access Logs](#).

- *TargetGrants*

The bucket owner is automatically granted `FULL_CONTROL` to all logs delivered to the bucket. This optional element enables you grant access to others (see [Access Control Lists](#)). Any specified `TargetGrants` are added to the default ACL.

To enable server access logging, Set or PUT a `BucketLoggingStatus` with a nested `LoggingEnabled` element. To disable server access logging, Set or PUT an empty `BucketLoggingStatus` element.

In REST, the address of the `BucketLoggingStatus` resource for a bucket 'mybucket' is `http://s3.amazonaws.com/mybucket?logging`. The PUT and GET methods are valid for this resource. For example, the following request fetches the `BucketLoggingStatus` resource for mybucket:

Example

```
GET ?logging HTTP/1.1
Host: mybucket.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS YOUR_AWS_ACCESS_KEY_ID:YOUR_SIGNATURE_HERE

HTTP/1.1 200 OK
Date: Wed, 01 Mar 2006 12:00:00 GMT
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>mybucketlogs</TargetBucket>
    <TargetPrefix>mybucket-access_log-</TargetPrefix>
    <TargetGrants>
```

```
<Grant>
  <Grantee xm
lns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail">
    <EmailAddress>user@company.com</EmailAddress>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
</TargetGrants>

</LoggingEnabled>
</BucketLoggingStatus>
```

In SOAP, you can work with BucketLoggingStatus resource using the [SetBucketLoggingStatus](#) and [GetBucketLoggingStatus](#) operations.

Amazon S3 checks the validity of the proposed BucketLoggingStatus when you try to Set or PUT to it. If the TargetBucket does not exist, is not owned by you, or does not have the appropriate grants, you will receive the InvalidTargetBucketForLogging error. If your proposed BucketLoggingStatus document is not well-formed XML or does not match our published schema, you will receive the MalformedXML error.

BucketLoggingStatus Changes Take Effect Over Time

Changes to the logging status for a bucket are visible in the configuration API immediately, but they take time to actually affect the delivery of log files. For example, if you enable logging for a bucket, some requests made in the following hour may be logged, while others may not. Or, if you change the target bucket for logging from bucket A to bucket B, some logs for the next hour might continue to be delivered to bucket A, while others might be delivered to the new target bucket B. In all cases, the new settings will eventually take effect without any further action on your part.

Delivery of Server Access Logs



Important

This section describes Beta functionality that is subject to change in future releases. Please provide feedback on this functionality in the [Amazon S3 Developer Forum](#).

Server access logs are delivered by writing them to the bucket of your choice. This 'target bucket' may or may not be the same as the bucket being logged. The owner of the bucket being logged must match the owner of the target bucket, otherwise no logs will be delivered.



Note

The source and the target buckets must be in the same location. For more information about bucket location constraints, see [Location Selection](#).

When a log file is delivered to the target bucket, it is stored under a key of the form:

TargetPrefixYYYY-mm-DD-HH-MM-SS-UniqueString

where YYYY, mm, DD, HH, MM and SS are the digits of the year, month, day, hour, minute, and

seconds (respectively) when the log file was delivered.

A log file delivered at time 't' can contain records written at any point before time 't'. There is no way to know whether all log records for a certain time interval have been delivered or not.

The `TargetPrefix` component of the key is a string provided by the bucket owner using the logging configuration API. For more information, see [Server Access Logging Configuration API](#).

The `UniqueString` component of the key carries no meaning and should be ignored by log processing software.

The system does not delete old log files. If you do not want server logs to accumulate, you must delete them yourself. To do so, use the `List` operation with the `prefix` parameter to locate old logs to delete. For more information, see [Listing Keys](#).

Access Control Interaction

Log files will be written to the target bucket under the identity of a member of the `http://acs.amazonaws.com/groups/s3/LogDelivery` group. These writes are subject to the usual access control restrictions. Therefore, logs will not be delivered unless the access control policy of the target bucket grants the log delivery group `WRITE` access. To ensure log files are delivered correctly, the log delivery group must also have `READ_ACP` permission on the target bucket. Consult the [Authentication and Access Control](#) section of the documentation for information about access control lists and groups, or see the [Setting Up Server Access Logging](#) for an example of how to correctly configure your target bucket's access control policy.

Log files created in the target bucket have an access control list entry that consists of a `FULL_CONTROL` grant to the bucket owner and grants to any users specified through the `TargetGrants` element.

Best Effort Server Log Delivery

The server access logging feature is designed for best effort. You can expect that most requests against a bucket that is properly configured for logging will result in a delivered log record, and that most log records will be delivered within a few hours of the time that they were recorded.

However, the server logging feature is offered on a best-effort basis. The completeness and timeliness of server logging is not guaranteed. The log record for a particular request may be delivered long after the request was actually processed, or it may never be delivered at all. The purpose of server logs is to give the bucket owner an idea of the nature of traffic against his or her bucket. It is not meant to be a complete accounting of all requests.

Usage Report Consistency

It follows from the best-effort nature of the server logging feature that the usage reports available at the AWS portal may include usage that does not correspond to any request in a delivered server log.

Server Access Log Format



Important

This section describes Beta functionality that is subject to change in future releases. Please provide feedback on this functionality in the [Amazon S3 Developer Forum](#).

The log files consist of a sequence of new-line delimited log records. Log records appear in no particular order. Each log record represents one request and consists of the following space delimited fields:

Amazon Simple Storage Service Developer Guide
Server Access Log Format

Field Name	Example Entry	Notes
Bucket Owner	314159b66967d86f031c7249d1d9a8024 9109428335cd0ef1cdc487b4566cb1b	The canonical user id of the owner of the source bucket.
Bucket	mybucket	The name of the bucket that the request was processed against. If the system receives a malformed request and cannot determine the bucket, the request will not appear in any server access log.
Time	[04/Aug/2006:22:34:02 +0000]	The time at which the request was received. The format, using <code>strftime()</code> terminology, is [%d/%B/%Y:%H:%M:%S %z]
Remote IP	72.21.206.5	The apparent Internet address of the requestor. Intermediate proxies and firewalls may obscure the actual address of the machine making the request.
Requestor	314159b66967d86f031c7249d1d9a80 249109428335cd0ef1cdc487b4566cb1b	The canonical user id of the requestor, or the string "Anonymous" for unauthenticated requests. This identifier is the same one used for access control purposes.
Request ID	3E57427F33A59F07	The request ID is a string generated by Amazon S3 to uniquely identify each request.
Operation	SOAP.CreateBucket or REST.PUT.OBJECT	Either <code>SOAP.operation</code> or <code>REST.HTTP_method.resource_type</code>
Key	/photos/2006/08/puppy.jpg	The 'key' part of the request, URL encoded, or '-' if the operation does not take a key parameter.
Request-URI	"GET /mybucket/photos/2006/08/ puppy.jpg?x-foo=bar"	The Request-URI part of the HTTP request message.
HTTP status	200	The numeric HTTP status code of the response.
Error Code	NoSuchBucket	The Amazon S3 Error Code , or '-' if no error occurred.
Bytes Sent	2662992	The number of response bytes sent, excluding HTTP protocol overhead,

Field Name	Example Entry	Notes
		or '-' if zero.
Object Size	3462992	The total size of the object in question.
Total Time	70	The number of milliseconds the request was in flight from the server's perspective. This value is measured from the time your request is received to the time that the last byte of the response is sent. Measurements made from the client's perspective may be longer due to network latency.
Turn-Around Time	10	The number of milliseconds that Amazon S3 spent processing your request. This value is measured from the time the last byte of your request was received until the time the first byte of the response was sent.
Referer	"http://www.amazon.com/webservices"	The value of the HTTP Referer header, if present. HTTP user-agents (e.g. browsers) typically set this header to the URL of the linking or embedding page when making a request.
User-Agent	"curl/7.15.1"	The value of the HTTP User-Agent header.

Any field may be set to '-' to indicate that the data was unknown or unavailable, or that the field was not applicable to this request.

Custom Access Log Information

You can include custom information to be stored in the access log record for a request by adding a custom query-string parameter to the URL for the request. Amazon S3 will ignore query-string parameters that begin with "x-", but will include those parameters in the access log record for the request, as part of the `Request-URI` field of the log record. For example, a GET request for "s3.amazonaws.com/mybucket/photos/2006/08/puppy.jpg?x-user=mrbar" will work the same as the same request for "s3.amazonaws.com/mybucket/photos/2006/08/puppy.jpg", except that the "x-user=mrbar" string will be included in the `Request-URI` field for the associated log record. This functionality is available in the REST interface only.

Extensible Server Access Log Format

From time to time, we may extend the access log record format by adding new fields to the end of each line. Code that parses server access logs must be written to handle trailing fields that it does not understand.

Setting Up Server Access Logging



Important

This section describes Beta functionality that is subject to change in future releases. Please provide feedback on this functionality in the [Amazon S3 Developer Forum](#).

The Amazon S3 server access logging feature lets you generate access log files for buckets that you own. These log files are delivered to you by writing them into a (possibly different) bucket that you own. Once delivered, the access logs are ordinary objects that you can read, list or delete at your convenience.

These instructions assume that you want to enable server access logging on one of your pre-existing buckets, and that you want to have those logs delivered into a new bucket you will create just for logging. We suppose that the bucket you want to log access to is called 'mybucket' and the new bucket you will create to hold your access logs is called 'mylogs'. This makes 'mybucket' the source bucket for logging and 'mylogs' the target bucket for logging. Whenever you see 'mybucket' or 'mylogs' in the example, replace them with the name of your bucket that you want to log, and the bucket you want to store your access logs, respectively.

This tutorial makes use of the [s3curl.pl sample program](#) to work with the Amazon S3 REST API. Make sure you use the most recent version of s3curl, as it has been updated to support this tutorial. After invoking s3curl, always check for a 200 OK HTTP response. If you get some other response code, refer to the XML error response which likely contains information about what went wrong.

Preparing the Target Bucket

First, decide if you want your logs delivered to an existing bucket, or if you want to create a new bucket just for access log files. The following command creates a new target bucket for logging. Notice the canned ACL argument that grants the system permission to write log files to this bucket.



Note

The source and the target buckets must be in the same location. For more information about bucket location constraints, see [Location Selection](#)

Example

```
$ ./s3curl.pl --id YOUR_AWS_ACCESS_KEY_ID --key YOUR_AWS_SECRET_ACCESS_KEY -  
-acl log-delivery-write --put /dev/null -- -s -v http://s3.amazonaws.com/my  
logs
```

If you just created a new bucket for logging, skip to the next section. Otherwise, to have your access logs files delivered to an existing bucket, you must modify the access control policy of that bucket by hand. Fetch the ?acl sub-resource of the target bucket and save it to a local file:

Example

```
$ ./s3curl.pl --id YOUR_AWS_ACCESS_KEY_ID --key YOUR_AWS_SECRET_ACCESS_KEY --  
-s -v 'http://s3.amazonaws.com/mylogs?acl' > mylogs.acl
```

Now open the local copy of the logging resource in your favorite text editor and insert a new `<Grant>` element to the `<AccessControlList>` section that gives the log delivery group `WRITE` and `READ_ACP` permission to your bucket.

Example

```
<Grant>  
  
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:type="Group">  
  
    <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>  
  
  </Grantee>  
  
  <Permission>WRITE</Permission>  
  
</Grant>  
  
<Grant>  
  
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:type="Group">  
  
    <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>  
  
  </Grantee>  
  
  <Permission>READ_ACP</Permission>  
  
</Grant>
```

Finally, apply the modified access control policy by writing it back to Amazon S3.

Example

```
$ ./s3curl.pl --id YOUR_AWS_ACCESS_KEY_ID --key YOUR_AWS_SECRET_ACCESS_KEY -  
-put mylogs.acl -- -s -v 'http://s3.amazonaws.com/mylogs?acl'
```

Enabling Server Access Logging on the Source Bucket

Now that the target bucket can accept log files, we'll update the `?logging` sub-resource of the source bucket to turn on server access logging. Remember that you must be the bucket owner to read or write this resource.

Fetch the `?logging` sub-resource for modification using the command:

Example

```
$ ./s3curl.pl --id YOUR_AWS_ACCESS_KEY_ID --key YOUR_AWS_SECRET_ACCESS_KEY --  
-s -v 'http://s3.amazonaws.com/mybucket?logging' > mybucket.logging
```

Open `mybucket.logging` in your favorite text editor and uncomment the `<LoggingSettings>` section. Replace the contents of the `<TargetBucket>` and `<TargetPrefix>` with `'mylogs'` and `'mybucket-access_log-'` respectively.

Additionally, to grant users access to log files within the bucket, you can specify one or more users in the `<TargetGrants>` section. You can specify users through their email address (`EmailAddress`) or canonical user ID (`CanonicalUser`). Permissions include `READ`, `WRITE`, and `FULL_CONTROL`. The result should be similar to the following:

Example

```
<?xml version="1.0" encoding="UTF-8"?>  
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">  
  <LoggingEnabled>  
    <TargetBucket>mylogs</TargetBucket>  
    <TargetPrefix>mybucket-access_log-</TargetPrefix>  
    <TargetGrants>  
      <Grant>  
        <Grantee xmlns:  
  xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:type="AmazonCustomerByEmail">  
          <EmailAddress>user@company.com</EmailAddress>  
        </Grantee>  
        <Permission>READ</Permission>  
      </Grant>  
    </TargetGrants>  
  </LoggingEnabled>  
</BucketLoggingStatus>
```



Note

For general information about authentication, see [Authentication and Access Control](#).

Now apply your modifications by writing the document back to the `?logging` sub-resource in Amazon S3.

Example

```
$ ./s3curl.pl --id YOUR_AWS_ACCESS_KEY_ID --key YOUR_AWS_SECRET_ACCESS_KEY -  
-put mybucket.logging -- -s -v 'http://s3.amazonaws.com/mybucket?logging'
```

You can confirm your changes by fetching the `?logging` sub-resource and comparing it to what you just wrote.

Server access logging should now be enabled. Make a few requests against the source bucket now, and your access logs should begin to be delivered to the target bucket within the next few hours.

Disabling Server Logging for a Bucket

Fetch, modify and apply the `?logging` sub resource in the same way as described above, except this time use your text editor to **REMOVE** the `<EnableLogging>` element.

Note that it takes some time for changes to take effect, so you may see log files delivered for a while after disabling logging.

Using the REST API

Topics

- [Common REST API Elements](#)
- [The REST Error Response](#)
- [Authenticating REST Requests](#)
- [Setting Access Policy with REST](#)
- [Virtual Hosting of Buckets](#)
- [Request Redirection and the REST API](#)
- [Browser-Based Uploads Using POST](#)
- [Operations on the Service](#)
- [Operations on Buckets](#)
- [Operations on Objects](#)

This section contains information specific to the Amazon S3 REST API.

The examples in this guide use the newer virtual hosted-style method for accessing buckets instead of the path-style. Although the path-style is still supported for legacy applications, we recommend using the virtual-hosted style where applicable. For more information, see [Working with Amazon S3 Buckets](#)

The following is an example of a virtual hosted-style request to delete the puppy.jpg file from the mybucket bucket:

```
DELETE /puppy.jpg HTTP/1.1
User-Agent: dotnet
Host: mybucket.s3.amazonaws.com
Date: Tue, 15 Jan 2008 21:20:27 +0000
x-amz-date: Tue, 15 Jan 2008 21:20:27 +0000
Authorization: AWS OPN5J17HBGZHT7JJ3X82:k3nL7gH3+PadhTEVn5EXAMPLE
```

The following is an example of a path-style version of the same request:

```
DELETE /mybucket/puppy.jpg HTTP/1.1
User-Agent: dotnet
Host: s3.amazonaws.com
Date: Tue, 15 Jan 2008 21:20:27 +0000
x-amz-date: Tue, 15 Jan 2008 21:20:27 +0000
Authorization: AWS OPN5J17HBGZHT7JJ3X82:k3nL7gH3+PadhTEVn5EXAMPLE
```

Common REST API Elements

Amazon S3 REST Operations are HTTP requests, as defined by RFC 2616 (<http://www.ietf.org/rfc/rfc2616.txt>). This section describes how Amazon S3 uses HTTP and the parts of HTTP requests and responses that Amazon S3 REST operations have in common. Detailed descriptions of individual operations are provided later in this guide.

A typical REST operation consists of a sending a single HTTP request to Amazon S3, followed by waiting for an HTTP response. Like any HTTP request, a request to Amazon S3 contains a request method, a URI, request headers, and sometimes a query string and request body. The response contains a status code, response headers, and sometimes a response body.

Example

The following example of a request shows how to get an object named "Nelson" from the "quotes" bucket:

Sample Request

```
GET /Nelson HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: qBmKRcEWBBhH6XAqsKU/eg24V3jf/kWKN9dJiplL/FpbYr9FDy7wWFurfdQOEMcY
x-amz-request-id: F2A8CCCA26B4B26D
Date: Wed, 01 Mar 2006 12:00:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
Content-Type: text/plain
Content-Length: 5
Connection: close
Server: AmazonS3
```


ha-ha

Common Request Headers

Amazon S3 REST requests include headers which contain basic information about the request. The table below describes common headers for Amazon S3 REST requests.

Amazon Simple Storage Service Developer Guide

The REST Error Response

Header Name	Description	Required
Authorization	The information required for request authentication. See Constructing the Authentication Header for details about the format.	Yes
Content-Length	Length of the message (without the headers) according to RFC 2616. Condition: Required for PUTs and operations that load XML, such as logging and ACLs.	Conditional
Content-Type	The content type of the resource. Example: <code>text/plain</code>	Optional
Date	The current date and time according to the requestor. Example: <code>Wed, 01 Mar 2006 12:00:00 GMT</code>	Yes
Host	Normally, the value of Host is <code>s3.amazonaws.com</code> . A Host header with a value other than <code>s3.amazonaws.com</code> selects the bucket for the request as described in Virtual Hosting of Buckets . Condition: Required for HTTP 1.1 (most toolkits add this header automatically); optional for HTTP/1.0 requests.	Conditional
x-amz-security-token	The security tokens for operations that use Amazon DevPay. Each request that uses Amazon DevPay requires two <code>x-amz-security-token</code> headers: one for the product token and one for the user token. Condition: Required for requests that use Amazon DevPay.  Note When Amazon S3 receives an authenticated request, it compares the computed signature with the provided signature. Improperly formatted multi-value headers used to calculate a signature can cause authentication issues. To ensure the signature is calculated properly, follow the instructions in the Constructing the CanonicalizedResource Element section.	Conditional

The REST Error Response

If a REST request results in an error, the HTTP reply has:

- an XML error document as the response body
- Content-Type: application/xml
- an appropriate 3xx, 4xx, or 5xx HTTP status code

Example

An example REST Error Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

For more information about Amazon S3 errors, see [Working with Errors](#).

Response Headers

The following response headers are returned by all operations:

- `x-amz-request-id`: This is a unique id assigned to each request by the system. In the unlikely event that you have problems with Amazon S3, Amazon can use this to help troubleshoot the problem.
- `x-amz-id-2`: This is a special token that will help us to troubleshoot problems.

Authenticating REST Requests

Authentication is the process of proving your identity to the system. Identity is an important factor in Amazon S3 access control decisions. Requests are allowed or denied in part based on the identity of the requester. For example, the right to create buckets is reserved for registered developers and (by default) the right to create objects in a bucket is reserved for the owner of the bucket in question. As a developer, you'll be making requests that invoke these privileges so you'll need to prove your identity to the system by authenticating your requests. This section shows you how.



Note

The content in this section does not apply to HTTP POST. For more information, see [Browser-Based Uploads Using POST](#).

The Amazon S3 REST API uses a custom HTTP scheme based on a keyed-HMAC (Hash Message Authentication Code) for authentication. To authenticate a request, you first concatenate selected elements of the request to form a string. You then use your AWS Secret Access Key to calculate the HMAC of that string. Informally, we call this process "signing the request," and we call the output of the HMAC algorithm the "signature" because it simulates the security properties of a real signature. Finally, you add this signature as a parameter of the request, using the syntax described below.

When the system receives an authenticated request, it fetches the AWS Secret Access Key that you claim to have, and uses it in the same way to compute a "signature" for the message it received. It then

compares the signature it calculated against the signature presented by the requester. If the two signatures match, then the system concludes that the requester must have access to the AWS Secret Access Key, and therefore acts with the authority of the principal to whom the key was issued. If the two signatures do not match, the request is dropped and the system responds with an error message.

Example An Example Authenticated Amazon S3 REST Request

```
GET /photos/puppy.jpg HTTP/1.1
Host: johnsmith.s3.amazonaws.com
Date: Mon, 26 Mar 2007 19:37:58 +0000
Authorization: AWS 0PN5J17HBGZHT7JJ3X82:frJIUN8DYpKDtOLCwo//y1lqDzg=
```

Constructing the Authentication Header

The Amazon S3 REST API uses the standard HTTP `Authorization` header to pass authentication information. (The name of the standard header is unfortunate, since it carries authentication information, not authorization). Under the Amazon S3 authentication scheme, the `Authorization` header has the following form:

```
Authorization: AWS AWSAccessKeyId:Signature
```

Developers are issued an AWS Access Key Id and AWS Secret Access Key when they register. For request authentication, the *AWSAccessKeyId* element identifies the secret key that was used to compute the signature, and (indirectly) the developer making the request.

The *Signature* element is the RFC 2104 HMAC-SHA1 of selected elements from the request, and so the *Signature* part of the `Authorization` header will vary from request to request. If the request signature calculated by the system matches the *Signature* included with the request, then the requester will have demonstrated possession to the AWS Secret Access Key. The request will then be processed under the identity, and with the authority, of the developer to whom the key was issued.

The following pseudo-grammar illustrates the construction of the `Authorization` request header. (`\n` means the Unicode code point U+000A)

```
Authorization = "AWS" + " " + AWSAccessKeyId + ":" + Signature
;

Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of( String
ToSign ) ) );

StringToSign = HTTP-Verb + "\n" +
Content-MD5 + "\n" +
```

```
Content-Type + "\n" +
Date + "\n" +
CanonicalizedAmzHeaders +
CanonicalizedResource;

CanonicalizedResource = [ "/" + Bucket ] +
                        <HTTP-Request-URI, from the protocol
name up to the query string> +
                        [ sub-resource, if present. For ex
ample "?acl", "?location", "?logging", or "?torrent" ];

CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1 is an algorithm defined by ["RFC 2104 - Keyed-Hashing for Message Authentication"](#). The algorithm takes as input two byte-strings: a key and a message. For Amazon S3 Request authentication, use your AWS Secret Access Key as the key, and the UTF-8 encoding of the *StringToSign* as the message. The output of HMAC-SHA1 is also a byte string, called the digest. The *Signature* request parameter is constructed by Base64 encoding this digest.

Request Canonicalization for Signing

Recall that when the system receives an authenticated request, it compares the computed request signature with the signature provided in the request. In order for the system-computed signature to match the developer-computed signature, the *StringToSign* for a request must be constructed by both parties in exactly the same way. We call the process of putting a request in an agreed-upon form for signing "canonicalization".

Constructing the CanonicalizedResource Element

CanonicalizedResource represents the Amazon S3 resource targeted by the request. Construct it for a REST request as follows:

- Start with the empty string (" ")
- If the request specifies a bucket using the HTTP Host header (virtual hosted-style), append the bucket name preceded by a "/" (e.g., "/bucketname"). For path-style requests and requests that don't address a bucket, do nothing. For more information on virtual hosted-style requests, see [Virtual Hosting of Buckets](#).
- Append the path part of the un-decoded HTTP Request-URI, up-to but not including the query string.
- If the request addresses a sub-resource, like `?location`, `?acl`, or `?torrent`, append the sub-resource including question mark.

Elements of the *CanonicalizedResource* that come from the HTTP Request-URI should be signed literally as they appear in the HTTP request, including URL-Encoding metacharacters.

The *CanonicalizedResource* may not be the same as the HTTP Request-URI. In particular, if your request uses the HTTP *Host* header to specify a bucket (as explained here: [Virtual Hosting of Buckets](#)),

the bucket will not appear in the HTTP Request-URI, however, the *CanonicalizedResource* continues to include the bucket. Query string parameters other than sub-resource flags (e.g., "?acl", "?location", "?logging", or "?torrent") will also appear in the Request-URI but are not included in *CanonicalizedResource*. See below for examples.

Constructing the CanonicalizedAmzHeaders Element

To construct the CanonicalizedAmzHeaders part of *StringToSign*, select all HTTP request headers that start with 'x-amz-' (using a case-insensitive comparison) and do the following:

- Convert each HTTP header name to lower-case. For example, 'X-Amz-Date' becomes 'x-amz-date'.
- Sort the collection of headers lexicographically by header name
- Combine header fields with the same name into one "header-name:comma-separated-value-list" pair as prescribed by RFC 2616, section 4.2, without any white-space between values. For example, the two metadata headers 'x-amz-meta-username: fred' and 'x-amz-meta-username: barney' would be combined into the single header 'x-amz-meta-username: fred,barney'
- "Un-fold" long headers that span multiple lines (as allowed by RFC 2616, section 4.2) by replacing the folding white-space (including new-line) by a single space.
- Trim any white-space around the colon in the header. For example, the header 'x-amz-meta-username: fred,barney' would become 'x-amz-meta-username:fred,barney'

Finally, append a new-line (U+000A) to each canonicalized header in the resulting list. Construct the CanonicalizedResource element by concatenating all headers in this list into a single string.

Positional versus Named HTTP Header StringToSign Elements

The first few header elements of *StringToSign* (Content-Type, Date, and Content-MD5) are positional in nature. *StringToSign* does not include the names of these headers, only their values from the request. In contrast, the 'x-amz-' elements are named; Both the header names and the header values appear in *StringToSign*.

If a positional header called for in the definition of *StringToSign* is not present in your request, (Content-Type or Content-MD5, for example, are optional for PUT requests, and meaningless for GET requests), substitute the empty string ("") in for that position.

Time Stamp Requirement

A valid time-stamp (using either the HTTP Date header or an x-amz-date alternative) is mandatory for authenticated requests. Furthermore, the client time-stamp included with an authenticated request must be within 15 minutes of the Amazon S3 system time when the request is received. If not, the request will fail with the *RequestTimeTooSkewed* error status code. The intention of these restrictions is to limit the possibility that intercepted requests could be replayed by an adversary. For stronger protection against eavesdropping, use the HTTPS transport for authenticated requests.

Some HTTP client libraries do not expose the ability to set the Date header for a request. If you have trouble including the value of the 'Date' header in the canonicalized headers, you can set the time-stamp for the request using an 'x-amz-date' header instead. The value of the x-amz-date header must be in one of the RFC 2616 formats (<http://www.ietf.org/rfc/rfc2616.txt>). When an x-amz-date header is present in a request, the system will ignore any Date header when computing the request signature. Therefore, if you include the x-amz-date header, use the empty string for the Date when constructing the *StringToSign*. See the next section for an example.

Examples

The following examples use the following (non-working) credentials:

Parameter	Value
AWSAccessKeyId	0PN5J17HBGZHT7JJ3X82
AWSSecretAccessKey	uV3F3YluFJax1cknvbcGwgjvx4QpvB+1eU8dUj2o

In the example *StringToSigns*, formatting is not significant and \n means the Unicode code point U+000A.

Example Object GET from the 'johnsmith' bucket.

Request	StringToSign
<pre>GET /photos/puppy.jpg HTTP/1.1 Host: johnsmith.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:36:42 +0000 Authorization: AWS 0PN5J17HBGZHT7JJ3X82: xXjDGYUmKxnwqr5KXNPGldn5LbA=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:36:42 +0000\n /johnsmith/photos/puppy.jpg</pre>

Note that the CanonicalizedResource includes the bucket name, but the HTTP Request-URI does not (it is specified by the Host header)

Example Object PUT to the 'johnsmith' bucket.

Request	StringToSign
<pre>PUT /photos/puppy.jpg HTTP/1.1 Content-Type: image/jpeg Content-Length: 94328 Host: johnsmith.s3.amazonaws.com Date: Tue, 27 Mar 2007 21:15:45 +0000 Authorization: AWS 0PN5J17HBGZHT7JJ3X82: hcicpDDvL9SsO6AkvxqmIWkmOuQ=</pre>	<pre>PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /johnsmith/photos/puppy.jpg</pre>

Amazon Simple Storage Service Developer Guide

Examples

Request	StringToSign

Note the Content-Type header in the request and in the StringToSign. Also note that the Content-MD5 is left blank in the StringToSign since it is not present in the request.

Example List of the 'johnsmith' bucket

Request	StringToSign
<pre>GET / ?prefix=photos&max-keys=50&marker=puppy HTTP/1.1 User-Agent: Mozilla/5.0 Host: johnsmith.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:42:41 +0000 Authorization: AWS 0PN5J17HBGZHT7JJ3X82:jsRt/rhG+Vtp88Hr YL706QhE4w4=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:42:41 +0000\n /johnsmith/</pre>

Note the trailing slash on the CanonicalizedResource, and the absence of query string parameters.

Example Fetch the access control policy sub-resource for the 'johnsmith' bucket

Request	StringToSign
<pre>GET /?acl HTTP/1.1 Host: johnsmith.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:44:46 +0000 Authorization: AWS 0PN5J17HBGZHT7JJ3X82:thdUi9VAkzhkniLj 96JIrOPGi0g=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:44:46 +0000\n /johnsmith/?acl</pre>

Notice how the sub-resource query string parameter is included in the CanonicalizedResource.

Example Delete an object from the 'johnsmith' bucket using the path-style and Date alternative

Amazon Simple Storage Service Developer Guide
Examples

Request	StringToSign
<pre>DELETE /johnsmith/photos/puppy.jpg HTTP/1.1 User-Agent: dotnet Host: s3.amazonaws.com Date: Tue, 27 Mar 2007 21:20:27 +0000 x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000 Authorization: AWS OPN5J17HBGZHT7JJ3X82:k3nL7gH3+PadhTEV n5Ip83x1Yzk=</pre>	<pre>DELETE\n \n \n \n x-amz-date:Tue, 27 Mar 2007 21:20:26 +0000\n /johnsmith/photos/puppy.jpg</pre>

Note how we used the alternate 'x-amz-date' method of specifying the date (because our client library prevented us from setting the date, say). In this case the field for the actual 'Date' header is left blank in the StringToSign.

Example Upload an object to a CNAME style virtual hosted bucket, with metadata

Request	StringToSign
<pre>PUT /db-backup.dat.gz HTTP/1.1 User-Agent: curl/7.15.5 Host: static.johnsmith.net:8080 Date: Tue, 27 Mar 2007 21:06:08 +0000 x-amz-acl: public-read content-type: application/x-download Content-MD5: 4gJE4saaMU4BqNR0kLY+lw== X-Amz-Meta-ReviewedBy: joe@johnsmith.net X-Amz-Meta-ReviewedBy: jane@johnsmith.net X-Amz-Meta-FileChecksum: 0x02661779 X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment; fi lename=database.dat Content-Encoding: gzip Content-Length: 5913339</pre>	<pre>PUT\n 4gJE4saaMU4BqNR0kLY+lw==\n application/x-download\n Tue, 27 Mar 2007 21:06:08 +0000\n x-amz-acl:public-read\n x-amz-meta-checksumalgorithm:crc32\n x-amz-meta-filechecksum:0x02661779\n x- amz- meta-re viewedby:joe@johnsmith.net,jane@johns mith.net\n /stat ic.johnsmith.net/db-backup.dat.gz</pre>

Amazon Simple Storage Service Developer Guide Examples

Request	StringToSign
<pre>Authorization: AWS 0PN5J17HBGZHT7JJ3X82:C0F10tU8Y1b9KDTp ZqYkZPX91iI=</pre>	

Notice how the 'x-amz-' headers are sorted, white-space trimmed, converted to lowercase, and multiple headers with the same name have been joined using a comma to separate values.

Note how only the Content-Type and Content-MD5 HTTP entity headers appear in the *StringToSign*. The other Content-* entity headers do not.

Again, note that the *CanonicalizedResource* includes the bucket name, but the HTTP Request-URI does not (the bucket is specified by the Host header).

Example List All My Buckets

Request	StringToSign
<pre>GET / HTTP/1.1 Host: s3.amazonaws.com Date: Wed, 28 Mar 2007 01:29:59 +0000 Authorization: AWS 0PN5J17HBGZHT7JJ3X82:Db+gepJSUbZKwpx1 FR0DLtEYOZA=</pre>	<pre>GET\n \n \n Wed, 28 Mar 2007 01:29:59 +0000\n /</pre>

Example Unicode Keys

Request	StringToSign
<pre>GET /diction ary/fran%C3%A7ais/pr%c3%a9f%c3%a8re HTTP/1.1 Host: s3.amazonaws.com Date: Wed, 28 Mar 2007 01:49:49 +0000 Authorization: AWS 0PN5J17HBGZHT7JJ3X82:dxhSBHoI6eVSPcXJ qEghlUzZMnY=</pre>	<pre>GET\n \n \n Wed, 28 Mar 2007 01:49:49 +0000\n /diction ary/fran%C3%A7ais/pr%c3%a9f%c3%a8re</pre>

Request	StringToSign

Note how the elements in *StringToSign* that were derived from the Request-URI are taken literally, including URL-Encoding and capitalization.

Debugging REST Request Signing Problems

When REST request authentication fails, the system responds to the request with an XML error document. The information contained in this error document is meant to help developers diagnose the problem. In particular, the *StringToSign* element of the *SignatureDoesNotMatch* error document tells you exactly what request canonicalization the system is using.

Some toolkits may silently insert headers that you do not know about beforehand, such as adding the header *Content-Type* during a PUT. In most of these cases, the value of the inserted header remains constant, allowing you to discover the missing headers using tools such as *Ethereal* or *tcpmon*.

Query String Request Authentication Alternative

It is possible to authenticate certain types of requests by passing the required information as query-string parameters as an alternative to the *Authorization* HTTP header. This is useful for enabling direct third-party browser access to your private Amazon S3 data, without proxying the request. The idea is to construct a "pre-signed" request and encode it as a URL that an end-user's browser can retrieve. Query string request authentication allows the issuer to limit a pre-signed request to be valid only before a specified expiration time.

The practice of signing a request and giving it to a third-party for execution is suitable only for simple object GET requests.

Example An Example Query String Authenticated Amazon S3 REST Request

```
GET /pho
tos/
puppy.jpg?AWSAccessKeyId=0PN5J17HBGZHT7JJ3X82&Expires=1141889120&Signature=vj
byPxybdZaNmGa%2ByT272YEAiv4%3D HTTP/1.1
```

Host: johnsmith.s3.amazonaws.com

Date: Mon, 26 Mar 2007 19:37:58 +0000

The query string request authentication method doesn't require any special HTTP headers. Instead, the required authentication elements are specified as query string parameters:

Query String Parameter Name	Example Value	Description
<i>AWSAccessKeyId</i>	0PN5J17HBGZHT7JJ3X82	Your AWS Access Key Id. Specifies the AWS Secret Access Key used to sign the request, and (indirectly) the identity of the developer making the request.

Amazon Simple Storage Service Developer Guide

Query String Request Authentication Alternative

Query String Parameter Name	Example Value	Description
<i>Expires</i>	1141889120	The time when the signature expires, specified as the number of seconds since the epoch (00:00:00 UTC on January 1, 1970). A request received after this time (according to the server), will be rejected.
<i>Signature</i>	vjbyPxybdZaNmGa%2ByT272YEAiv4%3D	The URL encoding of the Base64 encoding of the HMAC-SHA1 of <i>StringToSign</i> , as defined below.

The query string request authentication method differs slightly from the ordinary method but only in the format of the *Signature* request parameter and the *StringToSign* element. The following pseudo-grammar illustrates the query string request authentication method:

```
Signature = URL-Encode( Base64( HMAC-SHA1( UTF-8-Encoding-Of(
StringToSign ) ) ) );
```

```
StringToSign = HTTP-VERB + "\n" +
              Content-MD5 + "\n" +
              Content-Type + "\n" +
              Expires + "\n" +
              CanonicalizedAmzHeaders +
              CanonicalizedResource;
```

Notice how the *Signature* is URL-Encoded to make it suitable for placement in the query-string. Also note that in *StringToSign*, the HTTP Date positional element has been replaced with *Expires*. The *CanonicalizedAmzHeaders* and *CanonicalizedResource* are the same as above.

Example Query String Request Authentication Example

Request	StringToSign
GET /photo/ tos/ puppy.jpg?AWSAccessKeyId=0PN5J17HBGZHT7JJ3X82& Signature=rucSbH0yNEcP9oM2XNlouVI3BH4%3D& Expires=1175139620 HTTP/1.1 Host: johnsmith.s3.amazonaws.com	GET\n \n \n 1175139620\n /johnsmith/photos/puppy.jpg

Request	StringToSign

We assume that when a browser makes the GET request, it won't provide a Content-MD5 or a Content-Type header, nor will it set any x-amz- headers, so those parts of the *StringToSign* are left blank.

Setting Access Policy with REST

There are two ways to set the access control policy with REST. You can set the access control policy (ACP) for an existing bucket or object by requesting a PUT to `/bucket?acl` or `/bucket/key?acl`. Or, at the time you are writing a bucket or object you can include an x-amz-acl header with your PUT request that stores a canned ACP with the written resource.

Setting the ACL on an Existing Bucket or Object

You can set the ACL on an existing bucket or object by doing an HTTP PUT to `/bucket?acl`, or `/bucket/key?acl`, where the body of the operation is the new ACL. To edit an existing ACL, fetch `/bucket?acl` or `/bucket/key?acl` to get the existing ACL, edit it locally, and then PUT the modified version back to `?acl`.

Example

The following example demonstrates how to set an existing object ACL so that only the owner has full access to the object. (The owner's canonical user grant information is first found by executing a GET on `/quotes/Neo?acl`):

```
PUT /Neo?acl HTTP/1.1
Content-Length: 214
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Content-Type: text/plain
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=

<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
<ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Canned Access Policies

Because of restrictions in what can be sent via http headers, Amazon S3 supports the concept of canned access policies for REST. A canned access policy can be included with the `x-amz-acl` header as part of a `PUT` operation to provide shorthand representation of a full access policy. When Amazon S3 sees the `x-amz-acl` header as part of a `PUT` operation, it will assign the respective access policy to the resource created as a result of the `PUT`. If no `x-amz-acl` header is included with a `PUT` request, then the bucket or object is written with the private access control policy (even if, in the case of an object, the object already exists with some other pre-existing access control policy).

The following canned ACLs are supported for REST:

- *private*: Owner gets `FULL_CONTROL`. No one else has any access rights. This is the default.
- *public-read*: Owner gets `FULL_CONTROL` and the anonymous principal is granted `READ` access. If this policy is used on an object, it can be read from a browser with no authentication.
- *public-read-write*: Owner gets `FULL_CONTROL`, the anonymous principal is granted `READ` and `WRITE` access. This is a useful policy to apply to a bucket, if you intend for any anonymous user to `PUT` objects into the bucket.
- *authenticated-read*: Owner gets `FULL_CONTROL`, and any principal authenticated as a registered Amazon S3 user is granted `READ` access.

Example

The following example demonstrates how to write data to an object and makes the object readable by anonymous principals:

Sample Request

```
PUT /Neo HTTP/1.1
x-amz-acl: public-read
Content-Length: 4
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Content-Type: text/plain
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=

woah
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdmOdAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEbla0wiIOIx1+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
Date: Wed, 01 Mar 2006 12:00:00 GMT
ETag: "aba878a8"
Content-Length: 0
Connection: close
Server: AmazonS3
```

Virtual Hosting of Buckets

Virtual Hosting, in general, is the practice of serving multiple web sites from a single web server. One way to differentiate sites is by using the apparent host name of the request instead of just the path name

part of the URI. An ordinary Amazon S3 REST request specifies a bucket using the first slash delimited component of the Request-URI path. Alternatively, using Amazon S3 Virtual Hosting, you can address a bucket in a REST API call using the HTTP `Host` header. In practice, Amazon S3's interpretation of `Host` means that most buckets are automatically accessible (for limited types of requests) at `http://bucketname.s3.amazonaws.com`. Furthermore, by naming your bucket after your registered domain name and by making that name a DNS alias for Amazon S3, you can completely customize the URL of your Amazon S3 resources, for example: `http://my.bucketname.com/`

Besides the attractiveness of customized URLs, a second benefit of virtual hosting is the ability to publish to the 'root directory' of your bucket's virtual server. This can be important because many existing applications search for files in this standard location. For example, `favicon.ico`, `robots.txt`, `crossdomain.xml`, are all expected to be found at the root.

Using the HTTP Host Header to Specify the Bucket

So long as your GET request does not use the SSL endpoint, you may specify the bucket for the request using the HTTP `Host` header. The `Host` header in a REST request is interpreted as follows:

- If the `Host` header is omitted or its value is 's3.amazonaws.com', the bucket for the request will be the first slash-delimited component of the Request-URI, and the key for the request will be the rest of the Request-URI. This is the ordinary method, as illustrated by the first and second example in the table below. Note that omitting the `Host` header is only legal for HTTP 1.0 requests.
- Otherwise, if the value of the `Host` header ends in '.s3.amazonaws.com', then the bucket name is the leading component of the `Host` header's value up to '.s3.amazonaws.com'. The key for the request is the Request-URI. This interpretation exposes buckets as sub-domains of s3.amazonaws.com, and is illustrated by the third and fourth example in the table below.
- Otherwise, the bucket for the request will be the lower-cased value of the `Host` header and the key for the request is the Request-URI. This interpretation is useful when you have registered the same DNS name as your bucket name, and have configured that name to be a CNAME alias for Amazon S3. The procedure for registering domain names and configuring DNS is outside the scope of this document, but the result is illustrated by the final example in the table below.

Examples

The following example illustrates these cases:

Example URL	Example Request	Bucket Name for Request	Key for Request	Notes
<code>http://s3.amazonaws.com/johnsmith/homepage.html</code>	<pre>GET /johnsmith/homepage.html HTTP/1.1 Host: s3.amazonaws.com</pre>	johnsmith	homepage.html	the ordinary method
	<pre>GET /johnsmith/homepage.html HTTP/1.0</pre>	johnsmith	homepage.html	HTTP 1.0 may omit the Host header
		johnsmith	homepage.html	All

Amazon Simple Storage Service Developer Guide
Custom Amazon S3 URLs using CNAMEs

Example URL	Example Request	Bucket Name for Request	Key for Request	Notes
http://johnsmith.s3.amazonaws.com/homepage.html	GET /homepage.html HTTP/1.1 Host: johnsmith.s3.amazonaws.com			Lower-case Amazon S3 buckets are automatically addressable by the sub-domain method.
	GET /homepage.html HTTP/1.1 Host: JohnSmith.s3.amazonaws.com	johnsmith	homepage.html	Note that the case of the Host header is insignificant. Upper-case bucket names are not accessible using this method.
http://www.johnsmith.net/homepage.html	GET /homepage.html HTTP/1.1 Host: www.johnsmith.net	www.johnsmith.net	homepage.html	To host a website in Amazon S3 using this method, you must configure your DNS name as a CNAME alias for <code>bucketname.s3.amazonaws.com</code> .

Custom Amazon S3 URLs using CNAMEs

Depending on your needs, you might not want "s3.amazonaws.com" to appear on your web site or service. For example, if you host your web site's images on Amazon S3, you may prefer a URL like this:

http://images.johnsmith.net/

to one that looks like this:

`http://johnsmith-images.s3.amazonaws.com/`

To associate a hostname with an Amazon S3 bucket using CNAMEs:

1. Select a hostname that belongs to a domain you control. This example uses the `images` subdomain of the `johnsmith.net` domain.
2. Create a bucket that matches the hostname. In this example, the hostname and bucket are named `images.johnsmith.net`.



Note

Your bucket name must exactly match the hostname.

3. Create a CNAME record that defines the hostname as an alias for the Amazon S3 bucket. For example:

```
images.johnsmith.net CNAME images.johnsmith.net.s3.amazonaws.com
```



Important

For request routing reasons, the CNAME record must be defined exactly as above. Otherwise, it might appear to operate correctly, but will eventually result in unpredictable behavior.



Note

The exact procedure for configuring DNS depends on your DNS server or DNS provider and is beyond scope of this document.

Limitations

Because DNS names are case insensitive, only lower-case buckets are addressable using the virtual hosting method.

Specifying the bucket for the request using the HTTP `Host` header is supported:

- For non-SSL requests.
- Using the REST API. You cannot specify the bucket in SOAP by using a different endpoint.

Backwards Compatibility

Early versions of Amazon S3 incorrectly ignored the HTTP `Host` header. Applications that depend on this undocumented behavior must be updated to set the `Host` header correctly. Because Amazon S3 determines the bucket name from `Host` when present, the most likely symptom of this problem is to receive an unexpected `NoSuchBucket` error result code.

Request Redirection and the REST API

For general information about Amazon S3 redirects, see [Request Redirection and the REST API](#).

Redirects and HTTP User-Agents

Programs that work against the Amazon S3 REST API can handle redirects either at the application layer or the HTTP layer. Many HTTP client libraries and user agents can be configured to correctly handle redirects automatically. However, many others have incorrect or incomplete redirect implementations.

Before relying on a library to fulfill the redirect requirement, test the following edge cases:

- Verify all HTTP request headers are correctly included in the redirected request (the second request after receiving a redirect) including HTTP standards such as Authorization and Date
- Verify non-GET redirects, such as PUT and DELETE, work correctly
- Verify large PUT requests follow redirects correctly
- Verify PUT requests follow redirects correctly if the 100-continue response takes a long time to arrive

HTTP user-agents that strictly conform to RFC2616 may require explicit confirmation before following a redirect when the HTTP request method is not GET or HEAD. It is generally safe to follow redirects generated by Amazon S3 automatically, as the system will only issue redirects to hosts within the amazonaws.com domain and the effect of the redirected request will be the same as the original request.

Redirects and 100-Continue

To simplify redirect handling, improve efficiencies, and avoid the costs associated with sending a redirected request body twice, configure your application to use 100-continues for PUT operations. When your application uses 100-continue, it does not send the request body until it receives an acknowledgement. If the message is rejected based on the headers, the body of the message is not sent. For more information about 100-continue, see [RFC 2616 Section 8.2.3](#)



Note

According to RFC 2616, when using `Expect: Continue` with an unknown HTTP server, you should not wait an indefinite period before sending the request body. This is because some HTTP servers do not recognize 100-continue. However, Amazon S3 does recognize if your request contains an `Expect: Continue` and will respond with a provisional 100-continue status or a final status code. Additionally, no redirect error will occur after receiving the provisional 100 continue go-ahead. This will help you avoid receiving a redirect response while you are still writing the request body.

Example

This section provides an example of client-server interaction using HTTP redirects and 100-continue.

The following is a sample PUT to the `quotes.s3.amazonaws.com` bucket:

```
PUT /nelson.txt HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000
Content-Length: 6
Expect: 100-continue
```

Amazon S3 returns the following:

```
HTTP/1.1 307 Temporary Redirect
Location: http://quotes.s3-4c25d83b.amazonaws.com/nelson.txt?rk=8d47490b
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Mon, 15 Oct 2007 22:18:46 GMT
Server: AmazonS3
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the
    specified temporary endpoint. Continue to use the
    original request endpoint for future requests.
  </Message>
  <Endpoint>quotes.s3-4c25d83b.amazonaws.com</Endpoint>
  <Bucket>quotes</Bucket>
</Error>
```

The client follows the redirect response and issues a new request to the `quotes.s3-4c25d83b.amazonaws.com` temporary endpoint:

```
PUT /nelson.txt?rk=8d47490b HTTP/1.1
Host: quotes.s3-4c25d83b.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000
Content-Length: 6
Expect: 100-continue
```

Amazon S3 returns a 100-continue indicating the client should proceed with sending the request body:

```
HTTP/1.1 100 Continue
```

The client sends the request body:

ha ha\n

Amazon S3 returns the final response:

```
HTTP/1.1 200 OK
Date: Mon, 15 Oct 2007 22:18:48 GMT
ETag: "a2c8d6b872054293afd41061e93bc289"
Content-Length: 0
Server: AmazonS3
```

Browser-Based Uploads Using POST

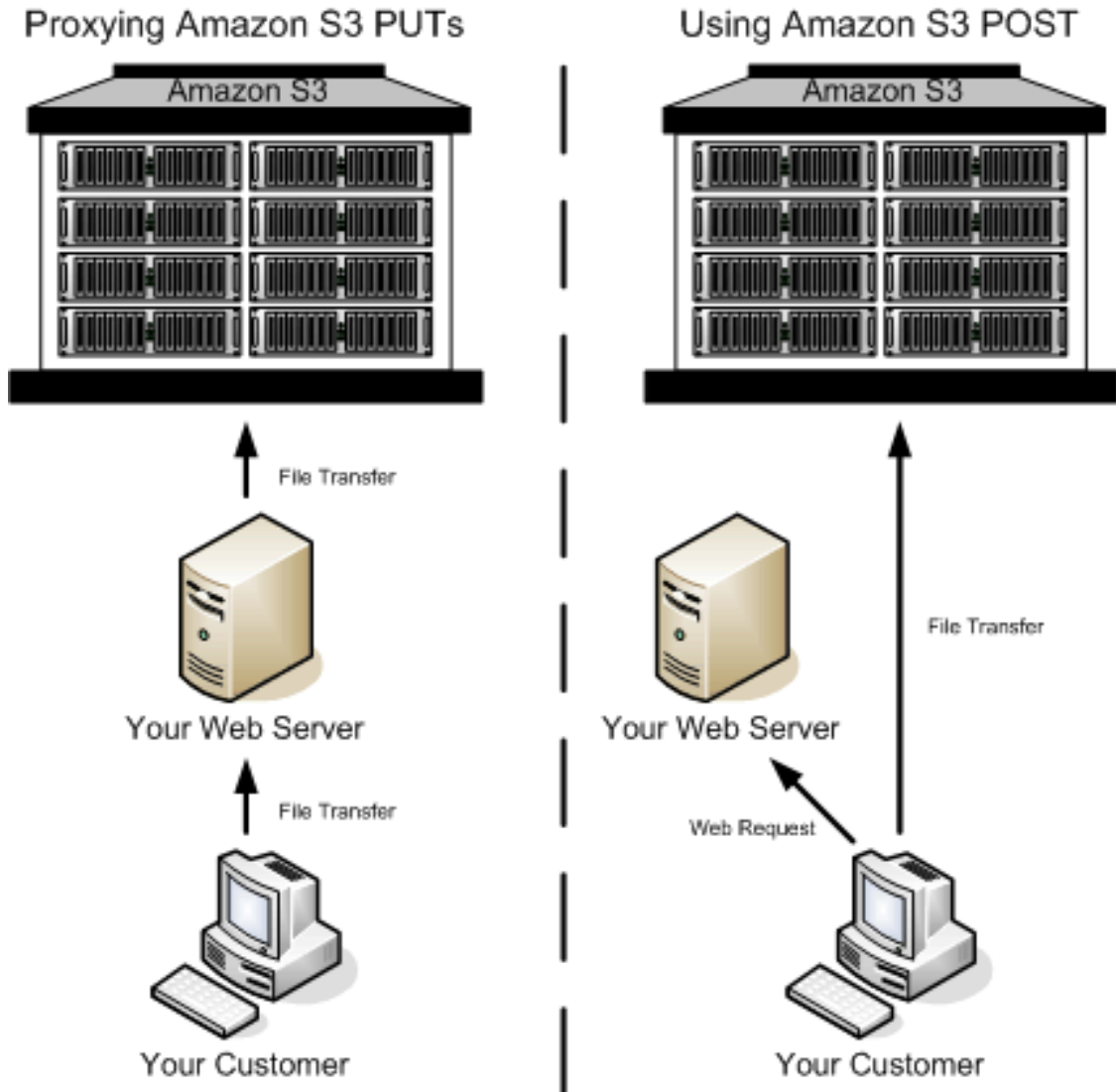
Topics

- [Introduction](#)
- [HTML Forms](#)
- [Examples](#)
- [POST with Adobe Flash](#)

Introduction

Amazon S3 supports POST, which allows your users to upload content directly to Amazon S3. POST is designed to simplify uploads, reduce upload latency, and save you money on applications where users upload data to store in Amazon S3.

The following image shows an upload using Amazon S3 POST.



Step	Description
1	The user opens a web browser and accesses your web page.
2	Your web page contains an HTTP form that contains all the information necessary for the user to upload content to Amazon S3.
3	The user uploads content directly to Amazon S3.



Note

Query string authentication is not supported for POST.

HTML Forms

Topics

- [HTML Form Encoding](#)
- [HTML Form Declaration](#)
- [HTML Form Fields](#)
- [Policy Construction](#)
- [Signature Construction](#)
- [Redirection](#)

When communicating with Amazon S3, you normally use the REST or SOAP APIs to perform put, get, delete, and other operations. With POST, users upload data directly to Amazon S3 through their browsers, which do not understand SOAP APIs or how to make a REST `PUT` request.

To allow users to upload content to Amazon S3 using their browsers, you use HTML forms. HTML Forms consist of a form declaration and form fields. The form declaration contains high level information about the request. The form fields contain detailed information about the request as well as the policy that is used to authenticate it and make sure that it meets conditions that you specify.



Note

The form data and boundaries (excluding the contents of the file) cannot exceed 20K.

This section describes how to use HTML forms.

HTML Form Encoding

The form and policy must be UTF-8 encoded. You can apply UTF-8 encoding to the form by specifying it in the HTML heading or as a request header.



Note

The HTML form declaration does not accept query string authentication parameters. For information about query string authentication, see [Query String Authentication](#).

The following is an example of UTF-8 encoding in the HTML heading:

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

The following is an example of UTF-8 encoding in a request header:

```
Content-Type: text/html; charset=UTF-8
```

HTML Form Declaration

The form declaration has three components: the action, the method, and the enclosure type. If any of these values is improperly set, the request fails.

The action specifies the URL that processes the request, which must be set to the URL of the bucket. For example, if the name of your bucket is "johnsmith", the URL is "http://johnsmith.s3.amazonaws.com/".



Note

The key name is specified in a form field.

The method must be POST.

The enclosure type (enctype) must be specified and must be set to multipart/form-data (Per [RFC 1867](#)) for both file uploads and text area uploads.

Example

This is an example of a form declaration for the bucket "johnsmith":

```
<form action="http://johnsmith.s3.amazonaws.com/" method="post" enctype="multipart/form-data">
```

HTML Form Fields

The following table describes a list of fields that can be used within a form.





Note

The variable `${filename}` is automatically replaced with the name of the file provided by the user and is recognized by all form fields. If the browser or client provides a full or partial path to the file, only the text following the last slash (/) or backslash (\) will be used (e.g., "C:\Program Files\directory\file.txt" will be interpreted as "file.txt"). If no file or filename is provided, the variable is replaced with an empty string.

Element Name	Description	Required?
AWSAccessKeyId	The AWS Access Key ID of the owner of the bucket who grants an Anonymous user access for a request that satisfies the set of constraints in the Policy.	Yes
acl	Specifies an Amazon S3 access control list. Options include private, public-read, public-read-write, authenticated-read. The default setting is private. If an invalid access control list is specified, an error is generated. For more information on ACLs, see Access	No

Amazon Simple Storage Service Developer Guide
HTML Forms

Element Name	Description	Required?
	Control Lists .	
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	REST-specific headers. For more information, see PUT Object .	No
key	The name of the uploaded key. To use the filename provided by the user, use the <code>\${filename}</code> variable. For example, if the user Betty uploads the file the file lolcatz.jpg and you specify <code>/user/betty/\${filename}</code> , the file is stored as <code>/user/betty/lolcatz.jpg</code> . For more information, see Keys .	Yes
policy	Security Policy describing what is permitted in the request. Requests without a security policy are considered anonymous and only work on publicly writable buckets. For more information, see Policy Construction	No
success_action_redirect, redirect	The URL to which the client is redirected upon successful upload. If <code>success_action_redirect</code> is not specified, Amazon S3 returns the empty document type specified in the <code>success_action_status</code> field. If Amazon S3 cannot interpret the URL, it acts as if the field is not present. If the upload fails, Amazon S3 displays an error and does not redirect the user to a URL. For more information, see Redirection .  Note The <code>redirect</code> field name is deprecated and support for the <code>redirect</code> field name will be removed in the future.	No
success_action_status	The status code returned to the client upon successful upload if <code>success_action_redirect</code> is not specified. Accepts the values 200, 201, or 204 (default). If the value is set to 200 or 204, Amazon S3 returns an empty document with a 200 or 204 status code. If the value is set to 201, Amazon S3 returns an XML document with a 201 status code. For information on the content of the XML document, see ???. If the value is not set or if it is set to an invalid value, Amazon S3 returns an empty document with a 204 status code.	No

Element Name	Description	Required?
	 <p>Note</p> <p>Some versions of the Adobe Flash player do not properly handle HTTP responses with an empty body. To support uploads through Adobe Flash, we recommend setting <i>success_action_status</i> to 201.</p>	
signature	<p>The HMAC signature constructed using the secret key of the provided AWSAccessKeyID.</p> <p>For more information, see Policy Construction and Authentication and Access Control.</p>	No
x-amz-security-token	<p>Amazon DevPay security token.</p> <p>Each request that uses Amazon DevPay requires two <code>x-amz-security-token</code> form fields: one for the product token and one for the user token.</p> <p>For more information, see Using Amazon DevPay with Amazon S3.</p>	No
Other field names prefixed with x-amz-meta-	<p>User-specified metadata.</p> <p>Amazon S3 does not validate or use this data.</p> <p>For more information, see PUT Object.</p>	No
file	<p>File or text content.</p> <p>The file or text content must be the last field in the form.</p> <p>You cannot upload more than one file at a time.</p>	Yes

Policy Construction

The policy is a UTF-8 and Base64 encoded JSON document that specifies conditions which the request must meet and is used to authenticate the content. Depending on how you design your policy documents, you can use them per-upload, per-user, for all uploads, or according to other designs that meet your needs.

The following is an example of a policy document:

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

The policy document contains the expiration and conditions.

Expiration

The expiration specifies the expiration date of the policy in ISO8601 GMT date format. For example, "2007-12-01T12:00:00.000Z" specifies that the policy is not valid after 12:00 GMT on 2007-12-01.

Conditions

The conditions in the policy document are used to validate the contents of the uploaded object. Each form field that you specify in the form (except `AWSAccessKeyId`, signature, file, policy, and field names that have an `x-ignore-` prefix) must be included in the list of conditions.



Note

If you have multiple fields with the same name, the values must be separated by commas. For example, if you have two fields named "x-amz-meta-tag" and the first one has a value of "Ninja" and second has a value of "Stallman", you would set the policy document to `Ninja,Stallman`.

All variables within the form are expanded prior to validating the policy. Therefore, all condition matching should be against the expanded fields. For example, if you set the key field to `user/betty/${filename}`, your policy might be ["starts-with", "\$key", "user/eric/"]. Do not enter ["starts-with", "\$key", "user/eric/\${filename}"]. For more information, see [Condition Matching](#).

Element Name	Description
<code>acl</code>	Specifies conditions the ACL must meet. Supports exact matching and <i>starts-with</i> .
<code>content-length-range</code>	Specifies the minimum and maximum allowable size for the uploaded content. Supports range matching.
<code>Cache-Control</code> , <code>Content-Type</code> , <code>Content-Disposition</code> , <code>Content-Encoding</code> , <code>Expires</code>	REST-specific headers. Supports exact matching and <i>starts-with</i> .
<code>key</code>	The name of the uploaded key. Supports exact matching and <i>starts-with</i> .
<code>success_action_redirect</code> , <code>redirect</code>	The URL to which the client is redirected upon successful upload. Supports exact matching and <i>starts-with</i> .
<code>success_action_status</code>	The status code returned to the client upon successful upload if <code>success_action_redirect</code> is not specified. Supports exact matching.
<code>x-amz-security-token</code>	Amazon DevPay security token. Each request that uses Amazon DevPay requires two <code>x-amz-security-token</code> form fields: one for the product token and one for the user token. As a result, the values must

Element Name	Description
	<p>be separated by commas. For example, if the user token is <code>eW91dHVizQ==</code> and the product token is <code>b0hnNVNKWVJIQTA=</code>, you set the policy entry to: { <code>"x-amz-security-token":</code> <code>"eW91dHVizQ==,b0hnNVNKWVJIQTA="</code> }.</p> <p>For more information about Amazon DevPay, see Using Amazon DevPay with Amazon S3.</p>
Other field names prefixed with <code>x-amz-meta-</code>	<p>User-specified metadata.</p> <p>Supports exact matching and <i>starts-with</i>.</p>



Note

If your toolkit adds additional fields (e.g., Flash adds filename), you must add them to the policy document. If you can control this functionality, prepend `x-ignore-` to the field so Amazon S3 ignores the feature and it won't affect future versions of this feature.

Condition Matching

The following table describes condition matching types. Although you must specify one condition for each form field that you specify in the form, you can create more complex matching criteria by specifying multiple conditions for a form field.

Condition	Description
Exact Matches	<p>Exact matches verify that fields match specific values. This example indicates that the ACL must be set to public-read:</p> <pre>{ "acl": "public-read" }</pre>
Starts With	<p>If the value must start with a certain value, use <i>starts-with</i>. This example indicates that the key must start with user/betty:</p> <pre>["starts-with", "\$key", "user/betty/"]</pre>
Matching Any Content	<p>To configure the policy to allow any content within a field, use <i>starts-with</i> with an empty value. This example allows any <code>success_action_redirect</code>:</p> <pre>["starts-with", "\$success_action_redirect", ""]</pre>
Specifying Ranges	<p>For fields that accept ranges, separate the upper and lower ranges with a comma. This example allows a file size from 1 to 10 megabytes:</p> <pre>["content-length-range", 1048579, 10485760]</pre>

Character Escaping

The following characters must be escaped within a policy document:

Escape Sequence	Description
\\	Backslash
\\$	Dollar symbol
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\uxxxx	All Unicode characters

Signature Construction

Step	Description
1	Encode the policy using UTF-8.
2	Encode those UTF-8 bytes using Base64.
3	Sign the policy with your Secret Access Key using HMAC SHA-1.
4	Encode the SHA-1 signature using Base64.

For information about constructing the policy, see [Policy Construction](#). For general information about authentication, see [Authentication and Access Control](#).

Redirection

General Redirection

On completion of the POST, the user is redirected to the location that you specified in the `success_action_redirect` field. If Amazon S3 cannot interpret the URL, it ignores the `success_action_redirect` field.

If `success_action_redirect` is not specified, Amazon S3 returns the empty document type specified in the `success_action_status` field.

If the POST fails, Amazon S3 displays an error and does not provide a redirect.

Pre-Upload Redirection

If your bucket was created using `<CreateBucketConfiguration>`, your end-users might require a redirect.

If this occurs, some browsers might handle the redirect incorrectly. This is relatively rare, but is most likely to occur right after a bucket is created.

Examples

Topics

- [File Upload](#)
- [Text Area Upload](#)

File Upload

This example shows the complete process for constructing a policy and form to upload a file attachment.

Constructing the Policy and Form

The following policy supports uploads to Amazon S3 for the johnsmith bucket:

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    { "bucket": "johnsmith" },
    [ "starts-with", "$key", "user/eric/" ],
    { "acl": "public-read" },
    { "success_action_redirect": "http://johnsmith.s3.amazonaws.com/successful_upload.html" },
    [ "starts-with", "$Content-Type", "image/" ],
    { "x-amz-meta-uuid": "14365123651274" },
    [ "starts-with", "$x-amz-meta-tag", "" ]
  ]
}
```

This policy requires the following:

- The upload must occur before 12:00 GMT on 2007-12-01
- The content must be uploaded to the johnsmith bucket
- The key must start with "/user/eric/"
- The ACL is set to public-read
- The success_action_redirect is set to http://johnsmith.s3.amazonaws.com/successful_upload.html
- The object is an image file
- The x-amz-meta-uuid tag must be set to 14365123651274
- The x-amz-meta-tag can contain any value

Amazon Simple Storage Service Developer Guide

Examples

The following is a Base64 encoded version of this policy:

```
eyAiZXhwaXJh
dGlvbiI6IClYMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAge
yJidWNrZXQiOiAi
am9obnNtaXRoIn0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCAidXNlci9lcmljLyJdLAogI
CAgeyJhY2wiOiAiCHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzcl9hY3Rpb25fcmlkaXJlY3QiOi
Ai
aHR0cDovL2pvaG5zbWl0aC5zMy5hbWF6b25hd3MuY29tL3NlY2Nlc3NmdWxkdXBsb2FkLmh0bWwif
SwKICAgIF
sic3RhcnczLXdpdGgiLCAiJENvbnRlbnQtVHlwZSIsICJpbWFnZS8iXSwKICAgIHsieC1hbXotbWV
OYS1ldWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LWFteilt
ZXRhLXRhZyIsICJlXQogIF0KfQo=
```

The secret key ID is uV3F3YluFJaxlcknvcGwgjvx4QpvB+leU8dUj2o, so the signature for the above Policy document is:

```
0RavWzkygo6QX9caELEqKi9kDbU=
```

The following form supports a POST to the johnsmith.net bucket using this policy:

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
    ...
    <form action="http://johnsmith.s3.amazonaws.com/" method="post" enc
type="multipart/form-data">
      Key to upload: <input type="input" name="key" value="user/eric/" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect"
value="http://johnsmith.s3.amazonaws.com/successful_upload.html" />
      Content-Type: <input type="input" name="Content-Type" value="image/jpeg"
/><br />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br /
>
      <input type="hidden" name="AWSAccessKeyId" value="15B4D3461F177624206A" /
>
      <input type="hidden" name="Policy" value="POLICY" />
      <input type="hidden" name="Signature" value="SIGNATURE" />
      File: <input type="file" name="file" /> <br />
```

```
<!-- The elements after this will be ignored -->
<input type="submit" name="submit" value="Upload to Amazon S3" />
</form>
...
</html>
```

Sample Request

This request assumes that the image uploaded is 117,108 bytes; the image data is not included.

```
POST / HTTP/1.1
Host: johnsmith.s3.amazonaws.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10)
Gecko/20071115 Firefox/2.0.0.10

Accept: text/
xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,im
age/png,*/*;q=0.5

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

Content-Type: multipart/form-data; bound
ary=-----9431149156168

Content-Length: 2661134

-----9431149156168
Content-Disposition: form-data; name="key"

user/eric/MyPicture.jpg
-----9431149156168
Content-Disposition: form-data; name="acl"

public-read
-----9431149156168
Content-Disposition: form-data; name="success_action_redirect"

http://johnsmith.s3.amazonaws.com/successful_upload.html
-----9431149156168
Content-Disposition: form-data; name="Content-Type"
```

Amazon Simple Storage Service Developer Guide Examples

```
image/jpeg
-----9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
-----9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

Some, Tag, For, Picture
-----9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

15B4D3461F177624206A
-----9431149156168
Content-Disposition: form-data; name="Policy"

eyJhZG90cDoyL2pvaG5zbWl0aC5zMy5hbWF6b25hd3MuY29tL3N1Y2Nlc3NmdWxzdXBsb2FkLmh0bWwifSwKICAgIFsic3RhcncRzLXdpdGgiLCAiJENvbnRlbnQtVHlwZSIsICJpbWFnZS8iXSwwKICAgIHSieClhbXotbWV0YS1ldWlkIjogIjE0MzYlMTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LWFteiltZXRhLXRhZyIsICJiXQogIF0KfQo=
-----9431149156168
Content-Disposition: form-data; name="Signature"

0RavWzkygo6QX9caELEqKi9kDbU=
-----9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

...file content...
-----9431149156168
Content-Disposition: form-data; name="submit"
```

Upload to Amazon S3

-----9431149156168--

Sample Response

The following is the sample response:

HTTP/1.1 303 Redirect

x-amz-request-id: 1AEE782442F35865

x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh

Content-Type: application/xml

Date: Wed, 14 Nov 2007 21:21:33 GMT

Connection: close

Location: ht

tp://johnsmith.s3.amazonaws.com/successful_upload.html?bucket=johnsmith&key=user/eric/MyPicture.jpg&etag="39d459dfbc0faabbb5e179358dfb94c3"

Server: AmazonS3

Text Area Upload

This example shows the complete process for constructing a policy and form to upload a text area. This is useful for submitting user-created content such as blog postings.

Constructing the Policy and Form

The following policy supports text area uploads to Amazon S3 for the johnsmith bucket:

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "johnsmith"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "ht
tp://johnsmith.s3.amazonaws.com/new_post.html"},
    ["eq", "$Content-Type", "text/html"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

This policy requires the following:

- The upload must occur before 12:00 GMT on 2007-12-01

Amazon Simple Storage Service Developer Guide

Examples

- The content must be uploaded to the johnsmith bucket
- The key must start with "user/eric/"
- The ACL is set to public-read
- The success_action_redirect is set to http://johnsmith.s3.amazonaws.com/new_post.html
- The object is HTML text
- The x-amz-meta-uuid tag must be set to 14365123651274
- The x-amz-meta-tag can contain any value

The following is a Base64 encoded version of this policy:

```
eyJhZG90cDovL2pvaG5zbWl0aC5zMy5hbWF6b25hd3MuY29tL251d19wb3N0Lmh0bWwifSwKICAgIFsiZ  
dGlvbiI6ICJyMDA3LTExYVpVDEYyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwoGICAgY  
yJidWNrZXQiOiAi  
am9obnNtaXRoIn0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCAidXNlci9lcmljLyJdLAogI  
CAGeyJhY2wiOiAicHVibG1jLXJlYWQifSwKICAgIHR5c3VjY2Vzc19hY3Rpb25fcmlkaXJlY3QiOi  
Ai  
aHR0cDovL2pvaG5zbWl0aC5zMy5hbWF6b25hd3MuY29tL251d19wb3N0Lmh0bWwifSwKICAgIFsiZ  
XEiLCAiJENvbnRlbnQtVHlwZSIs  
ICJ0ZXh0L2h0bWwiXSwwKICAgIHR5c3VjY2Vzc19hY3Rpb25fcmlkaXJlY3QiOiA  
iAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXh0L2h0bWwifSwKICAgIF0KfQo=
```

The secret key ID is uV3F3Y1uFJax1cknvbcGwgjvx4QpvB+leU8dUj2o, so the signature for the above Policy document is:

```
qA7FWXKq6VvU68lI9KdveT1cWgE=
```

The following form supports a POST to the johnsmith.net bucket using this policy:

```
<html>  
  
  <head>  
  
    ...  
  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />  
  
    ...  
  
  </head>  
  
  <body>  
  
    ...  
  
    <form action="http://johnsmith.s3.amazonaws.com/" method="post" enc  
type="multipart/form-data">  
  
      Key to upload: <input type="input" name="key" value="user/eric/" /><br />  
  
      <input type="hidden" name="acl" value="public-read" />  
  
      <input type="hidden" name="success_action_redirect"  
value="http://johnsmith.s3.amazonaws.com/new_post.html" />  
  
      <input type="hidden" name="Content-Type" value="text.html" />  
  
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />  
  
      Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
```

Amazon Simple Storage Service Developer Guide

Examples

```
>
> <input type="hidden" name="AWSAccessKeyId" value="15B4D3461F177624206A" /
>
  <input type="hidden" name="Policy" value="POLICY" />
  <input type="hidden" name="Signature" value="SIGNATURE" />
  Entry: <textarea name="file" cols="60" rows="10">
Your blog post goes here.
  </textarea><br />
  <!-- The elements after this will be ignored -->
  <input type="submit" name="submit" value="Upload to Amazon S3" />
</form>
...
</html>
```

Sample Request

This request assumes that the image uploaded is 117,108 bytes; the image data is not included.

```
POST / HTTP/1.1
Host: johnsmith.s3.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10)
Gecko/20071115 Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=-----178521717625888
Content-Length: 5299

-----178521717625888
Content-Disposition: form-data; name="key"

user/eric/NewEntry.html
-----178521717625888
```


Amazon Simple Storage Service Developer Guide

Examples

```
Content-Disposition: form-data; name="acl"

public-read
-----178521717625888
Content-Disposition: form-data; name="success_action_redirect"

http://johnsmith.s3.amazonaws.com/new_post.html
-----178521717625888
Content-Disposition: form-data; name="Content-Type"

text.html
-----178521717625888
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
-----178521717625888
Content-Disposition: form-data; name="x-amz-meta-tag"

Interesting Post
-----178521717625888
Content-Disposition: form-data; name="AWSAccessKeyId"

15B4D3461F177624206A
-----178521717625888
Content-Disposition: form-data; name="Policy"

eyJhZG90cDovL2pvaG5zbWl0aC5zMy5hbWF6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSIsICJ0ZXh0L2h0bWwiXSwKICAgIHsieC1hbXotbWV0YS1ldWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCgAgICBbInN0YXJ0cy13aXR0IiwgIiR4LWFteiltZXRhLXRhZyIsICJ0IiwgIF0KfQo=
-----178521717625888
Content-Disposition: form-data; name="Signature"

qA7FWXKq6VvU68lI9KdveTlcWgE=
```

```
-----178521717625888
Content-Disposition: form-data; name="file"

...content goes here...
-----178521717625888
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
-----178521717625888--
```

Sample Response

The following is the sample response:

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close

Location: ht
tp://johnsmith.s3.amazonaws.com/new_post.html?bucket=johnsmith&key=user/eric/
NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4

Server: AmazonS3
```

POST with Adobe Flash

Adobe Flash Player Security

By default, the Adobe Flash Player security model prohibits Adobe Flash Players from making network connections to servers outside the domain that serves the SWF file.

To override the default, you must upload a public-readable `crossdomain.xml` file to the bucket that will accept POST uploads. The following is a sample `crossdomain.xml` file:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
  <allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```



Note

We highly recommend reading more about the Adobe Flash security model on the [Adobe web site](#).

Adding the `crossdomain.xml` file to your bucket allows any Adobe Flash Player to connect to the `crossdomain.xml` file within your bucket. However, it does not grant access to the actual Amazon S3 bucket.

Other Adobe Flash Considerations

The FileReference API in Adobe Flash adds the `Filename` form field to the POST request. When building Adobe Flash applications that upload to Amazon S3 using the FileReference API, include the following condition in your policy:

```
[ 'starts-with', '$Filename', '' ]
```

Some versions of the Adobe Flash Player do not properly handle HTTP responses that have an empty body. To configure POST to return a response that does not have an empty body, set `success_action_status` to 201. When set, Amazon S3 returns an XML document with a 201 status code. For information on the content of the XML document, see [???](#). For information on form fields, see [HTML Form Fields](#).

Operations on the Service

Topics

- [GET Operation](#)

This section describes operations you can perform on the Amazon S3 service.

GET Operation

The GET operation on the Service endpoint (`s3.amazonaws.com`) returns a list of all of the buckets owned by the authenticated sender of the request.

Example

Sample Request

```
GET / HTTP/1.1
Host: s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult xmlns="http://doc.s3.amazonaws.com/2006-03-01">
```

```
<Owner>
  <ID>bcaf1ffd86f41caff1a493dc2ad8c2c281e37522a640e161ca5fb16fd081034f</ID>
  <DisplayName>webfile</DisplayName>
</Owner>
<Buckets>
  <Bucket>
    <Name>quotes</Name>
    <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
  </Bucket>
  <Bucket>
    <Name>samples</Name>
    <CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
  </Bucket>
</Buckets>
</ListAllMyBucketsResult>
```

Response Body

- *Owner*: This provides information that Amazon S3 uses to represent your identity for purposes of authentication and access control. ID is a unique and permanent identifier for the developer who made the request. DisplayName is a human-readable name representing the developer who made the request. It is not unique, and may change over time.
- *Name*: The name of a bucket. Note that if one of your buckets was recently deleted, the name of the deleted bucket may still be present in this list for a period of time.
- *CreationDate*: The time that the bucket was created.

Access Control

You must authenticate with a valid AWS Access Key ID that has been registered as a user of Amazon S3. Anonymous requests are never allowed to list buckets, and you cannot list buckets that you did not create.

Operations on Buckets

Topics

- [PUT Bucket](#)
- [GET Bucket](#)
- [GET Bucket Location](#)
- [DELETE Bucket](#)

- [POST Object](#)

This section describes operations you can perform on Amazon S3 buckets.

PUT Bucket

The PUT request operation with a bucket URI creates a new bucket. Depending on your latency and legal requirements, you can specify a location constraint that will affect where your data physically resides. You can currently specify a Europe (EU) location constraint.



Note

Not every string is an acceptable bucket name. For information on bucket naming restrictions, see [Working with Amazon S3 Buckets](#).

If you create a bucket using `<CreateBucketConfiguration>`, applications that access your bucket must be able to handle 307 redirects.

If you do not specify a location constraint, Amazon S3 automatically selects a location which will be billed at the standard Amazon S3 rates.

Example

Creates a bucket named "colorpictures" without a location constraint:

Sample Request

```
PUT / HTTP/1.1
Host: colorpictures.s3.amazonaws.com
Content-Length: 0
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPiFbiKa2bj0KMg95r/0zo3emzU4dzsD4rcKCHQUAdQkf3ShJTOOpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Mar 2006 12:00:00 GMT
Location: /colorpictures
Content-Length: 0
Connection: close
Server: AmazonS3
```

Creates a bucket named "colourpictures" using the Europe location constraint:

Sample Request

```
PUT / HTTP/1.1
Host: colourpictures.s3.amazonaws.com
Content-Length: 111
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=

<CreateBucketConfiguration>
  <LocationConstraint>EU</LocationConstraint>
</CreateBucketConfiguration>
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: YgIPiFbiKa2bj0KMg95r/0zo3emzU4dzsD4rcKCHQUAdQkf3ShJT0OpXUueF6QKo
x-amz-request-id: 236A8905248E5A01
Date: Wed, 01 Mar 2006 12:00:00 GMT
Location: /colourpictures
Content-Length: 0
Connection: close
Server: AmazonS3
```

Access Control

You must authenticate with a valid AWS Access Key ID that has been registered as a user of Amazon S3. Anonymous requests are never allowed to create buckets.

GET Bucket

A GET request operation using a bucket URI lists information about the objects in the bucket.

For a general introduction to the list operation, see the [Listing Keys](#) section.

Example

List up to 40 keys in the "quotes" bucket that have the prefix "N" and occur lexicographically after "Ned":

Sample Request

```
GET ?prefix=N&marker=Ned&max-keys=40 HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: gyB+3jRPnrkN98Za jxHXr3u7EFM67bNgSAXexeEHndCX/7GRnfTXxReKUQF28IFP
x-amz-request-id: 3B3C7C725673C630
Date: Wed, 01 Mar 2006 12:00:00 GMT
Content-Type: application/xml
Content-Length: 302
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Name>quotes</Name>
  <Prefix>N</Prefix>
  <Marker>Ned</Marker>
  <MaxKeys>40</MaxKeys>
  <IsTruncated>>false</IsTruncated>
  <Contents>
    <Key>Nelson</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <Size>5</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>bca1fffd86f41caff1a493dc2ad8c2c281e37522a640e161ca5fb16fd081034f</ID>
      <DisplayName>webfile</DisplayName>
    </Owner>
  </Contents>
  <Contents>
    <Key>Neo</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <Size>4</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>bca1fffd86f41caff1a493dc2ad8c2c281e37522a640e161ca5fb16fd081034f</ID>
      <DisplayName>webfile</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

Request Parameters

For comprehensive information about the list request parameters, see [Common List Request Parameters](#).

- *prefix*: Limits the response to keys which begin with the indicated prefix. You can use prefixes to separate a bucket into different sets of keys in a way similar to how a file system uses folders.
- *marker*: Indicates where in the bucket to begin listing. The list will only include keys that occur lexicographically after marker. This is convenient for pagination: To get the next page of results use the last key of the current page as the marker.
- *max-keys*: The maximum number of keys you'd like to see in the response body. The server may return fewer than this many keys, but will not return more.
- *delimiter*: Causes keys that contain the same string between the prefix and the first occurrence of the delimiter to be rolled up into a single result element in the CommonPrefixes collection. These rolled-up keys are not returned elsewhere in the response.

Response Body

For information about the list response, see [Common List Response Elements](#)

Access Control

To list the keys of a bucket you need to have READ access to the bucket.

GET Bucket Location

A GET location request operation using a bucket URI lists the location constraint of the bucket.

Example

List the bucket location constraint of the quotes bucket:

Sample Request

```
GET /?location HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Tue, 09 Oct 2007 20:26:04 +0000
Authorization: AWS 1ATXQ3HHA59CYF1CVS02:JUtd9kkJFjbKbkP9f6T/tAxozYY=
```

Sample Response

```
<?xml version="1.0" encoding="UTF-8"?>
<LocationConstraint xm
lns="http://s3.amazonaws.com/doc/2006-03-01/">EU</LocationConstraint>
```

Request Parameters

There are no request parameters for the GET location constraint request operation.

Access Control

To view the location constraint of a bucket, you must be the bucket owner.

DELETE Bucket

The DELETE request operation deletes the bucket named in the URI. All objects in the bucket must be deleted before the bucket itself can be deleted.

Example

Delete the bucket named "quotes"

Sample Request

```
DELETE / HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Sample Response

```
HTTP/1.1 204 No Content
x-amz-id-2: JuKZqmXuiwFeDQxhd7M8KtsKobSzWA1QEjLbTMTagkKdBX2z7I1/jGhDeJ3j6s80
x-amz-request-id: 32FE2CEB32F5EE25
Date: Wed, 01 Mar 2006 12:00:00 GMT
Connection: close
Server: AmazonS3
```

Access Control

Only the owner of a bucket is allowed to delete it, regardless of the bucket's access control policy.

POST Object

The POST request operation adds an object to a bucket using forms.



Note

For information about forms and constructing requests using POST, see [Browser-Based Uploads Using POST](#).

The response indicates that the object is successfully stored. Amazon S3 never stores partial objects: if you receive a successful response, the entire object was stored.

If the object already exists in the bucket, the new object overwrites the existing object.

Amazon S3 orders all of the requests that it receives. If two requests are sent nearly simultaneously, we might receive them in a different order than they were sent. The last request received is the one that is stored in Amazon S3, even though both requests receive a success response. This occurs because Amazon S3 is a distributed system and it might take a few seconds for one part of the system to realize that another part has received an object update. In this release of Amazon S3, you cannot lock an object for writing. If you require this functionality, you should provide it at the application layer.

If you specify a location constraint when creating a bucket, all objects added to the bucket are stored in the bucket's location. For example, if you specify a Europe (EU) location constraint for a bucket, all of that bucket's objects are stored in Europe. For more information on location constraints, see [Location Selection](#).

Example

This request assumes that the image uploaded is 117,108 bytes; the image data is not included.

```
POST / HTTP/1.1
Host: johnsmith.s3.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10)
Gecko/20071115 Firefox/2.0.0.10
```

Amazon Simple Storage Service Developer Guide
POST Object

Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

Content-Type: multipart/form-data; boundary=-----9431149156168

Content-Length: 2661134

-----9431149156168

Content-Disposition: form-data; name="key"

user/eric/MyPicture.jpg

-----9431149156168

Content-Disposition: form-data; name="acl"

public-read

-----9431149156168

Content-Disposition: form-data; name="success_action_redirect"

http://johnsmith.s3.amazonaws.com/successful_upload.html

-----9431149156168

Content-Disposition: form-data; name="Content-Type"

image/jpeg

-----9431149156168

Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274

-----9431149156168

Content-Disposition: form-data; name="x-amz-meta-tag"

Some, Tag, For, Picture

-----9431149156168

Amazon Simple Storage Service Developer Guide

POST Object

Content-Disposition: form-data; name="AWSAccessKeyId"

15B4D3461F177624206A

-----9431149156168

Content-Disposition: form-data; name="Policy"

eyJiZXhwaXJh
dGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAge
yJidWNrZXQiOiAi
am9obnNtaXRoIn0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCAidXNlci9lcmlljLyJdLAogI
CAgeyJhY2wiOiAicHVibGllLXJlYWQifSwKICAgIHsic3VjY2Vzcl9hY3Rpb25fcmlkXjY3Qm9i
Ai
aHR0cDovL2pvaG5zbWl0aC5zMy5hbWF6b25hd3MuY29tL3NlY2Nlc3NmdWxzdXBsb2FkLmh0bWwif
SwKICAgIF
sic3RhcnczLXdpdGgiLCAiJENvbnRlbnQtVHlwZSIsICJpbWFnZS8iXSwwKICAgIHsieC1hbXotbWw
0YS1ldWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCAidXNlci9lcmlljLyJdLAogI
ZXRhLXRhZyIsICJiXQogIF0KfQo=

-----9431149156168

Content-Disposition: form-data; name="Signature"

0RavWzkygo6QX9caELEqKi9kDbU=

-----9431149156168

Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"

Content-Type: image/jpeg

...file content...

-----9431149156168

Content-Disposition: form-data; name="submit"

Upload to Amazon S3

-----9431149156168--

The following is the sample response:

Sample Response

HTTP/1.1 303 Redirect

x-amz-request-id: 1AEE782442F35865

x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh

Content-Type: application/xml

Date: Wed, 14 Nov 2007 21:21:33 GMT

Connection: close

Location: http://johnsmith.s3.amazonaws.com/successful_upload.html?bucket=johnsmith&key=user/eric/MyPicture.jpg&etag="39d459dfbc0faabbb5e179358dfb94c3"

Server: AmazonS3

Request Headers

The following table describes request headers.

- *Content-Type*: This must be set to *multipart/form-data*. For more information, see [RFC 1867](#).

Form Data

The following table describes form data parts.





Note

The variable `${filename}` is automatically replaced with the name of the file provided by the user and is recognized by all form fields. If the browser or client provides a full or partial path to the file, only the text following the last slash (/) or backslash (\) will be used (e.g., "C:\Program Files\directory1\file.txt" will be interpreted as "file.txt"). If no file or filename is provided, the variable is replaced with an empty string.

Element Name	Description	Required?
AWSAccessKeyId	The AWS Access Key ID of the owner of the bucket who will be granting an Anonymous user access for a request that satisfies the set of constraints in the Policy.	Yes
acl	Specifies an Amazon S3 access control list. Options include private, public-read, public-read-write, authenticated-read. The default setting is private. If an invalid access control list is specified, an error is generated. For more information on ACLs, see Access Control Lists .	No
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	REST-specific headers. For more information, see PUT Object .	No
key	The name of the uploaded key. To use the filename provided by the user, use the <code>\${filename}</code> variable. For example, if the user Betty uploads the file the file lolcatz.jpg and you specify <code>/user/betty/\${filename}</code> , the file will be stored as <code>/user/betty/lolcatz.jpg</code> . For more information, see Keys .	Yes

Amazon Simple Storage Service Developer Guide
POST Object

Element Name	Description	Required?
policy	<p>Security Policy describing what is permitted in the request. Requests without a security policy are considered anonymous and only work on publicly writable buckets.</p> <p>For more information, see Policy Construction</p>	No
success_action_redirect, redirect	<p>The URL to which the client is redirected upon successful upload.</p> <p>If success_action_redirect is not specified, Amazon S3 returns the empty document type specified in the success_action_status field.</p> <p>If Amazon S3 cannot interpret the URL, it acts as if the field is not present.</p> <p>If the upload fails, Amazon S3 displays an error and does not redirect the user to a URL.</p> <p>For more information, see Redirection.</p> <p> Note</p> <p>The redirect field name is deprecated and support for the redirect field name will be removed in the future.</p>	No
success_action_status	<p>The status code returned to the client upon successful upload if success_action_redirect is not specified.</p> <p>Accepts the values 200, 201, or 204 (default).</p> <p>If the value is set to 200 or 204, Amazon S3 returns an empty document with a 200 or 204 status code.</p> <p>If the value is set to 201, Amazon S3 returns an XML document with a 201 status code. For information on the content of the XML document, see ???.</p> <p>If the value is not set or if it is set to an invalid value, Amazon S3 returns an empty document with a 204 status code.</p> <p> Note</p> <p>Some versions of the Adobe Flash player do not properly handle HTTP responses with an empty body. To support uploads through Adobe Flash, we recommend setting <i>success_action_status</i> to 201.</p>	No
x-amz-security-token	<p>Amazon DevPay security token.</p> <p>Each request that uses Amazon DevPay requires</p>	No

Element Name	Description	Required?
	two <code>x-amz-security-token</code> form fields: one for the product token and one for the user token. For more information, see Using Amazon DevPay with Amazon S3 .	
Other field names prefixed with <code>x-amz-meta-</code>	User-specified metadata. Amazon S3 does not validate or use this data. For more information, see PUT Object .	No
file	File or text content. The file or text content must be the last field in the form. You cannot upload more than one file at a time.	Yes

POST Response Body

If `success_action_status` is set to 200, 204, or is not specified, the response body will be empty. If `success_action_status` is set to 201, Amazon S3 returns a PostResponse XML document which contains the location of the object, the bucket in which it is stored, the key associated with the object, and its Etag.

The following is a sample PostResponse XML document:

```
<PostResponse>

<Location>http://johnsmith.s3.amazonaws.com/user/eric/MyPicture.jpg</Location>

<Bucket>johnsmith</Bucket>

<Key>user/eric/MyPicture.jpg</Key>

<ETag>"39d459dfbc0faabbb5e179358dfb94c3"</ETag>

</PostResponse>
```

Request Body

The multipart form data. For more information, refer to [RFC 1867](#).

Response Headers

- *Etag*: The entity tag is an MD5 hash of the object that you can use to do conditional GET operations using the If-Modified request tag with the GET request operation.

- `success_action_redirect`, `redirect`: The URL to which the client is redirected on successful upload.

Access Control

The signer of the policy must have `WRITE` access to the bucket to add an object. If there is no policy, Anonymous must have write access to the bucket. This is not recommended.

Operations on Objects

Topics

- [PUT Object](#)
- [GET Object](#)
- [HEAD Object](#)
- [DELETE Object](#)

This section describes operations you can perform on Amazon S3 objects.

PUT Object

The `PUT` request operation adds an object to a bucket.

The response indicates that the object has been successfully stored. Amazon S3 never stores partial objects: if you receive a successful response, then you can be confident that the entire object was stored.

If the object already exists in the bucket, the new object overwrites the existing object. Amazon S3 orders all of the requests that it receives. It is possible that if you send two requests nearly simultaneously, we will receive them in a different order than they were sent. The last request received is the one which is stored in Amazon S3. Note that this means if multiple parties are simultaneously writing to the same object, they may all get a successful response even though only one of them wins in the end. This is because Amazon S3 is a distributed system and it may take a few seconds for one part of the system to realize that another part has received an object update. In this release of Amazon S3, there is no ability to lock an object for writing -- such functionality, if required, should be provided at the application layer.

If you specify a location constraint when creating a bucket, all objects added to the bucket are stored in the bucket's location. For example, if you specify a Europe (EU) location constraint for a bucket, all of that bucket's objects are stored in Europe. For more information on location constraints, see [Location Selection](#).



Note

To configure your application to send the request headers prior to sending the request body, use `100-continue`. For `PUT` operations, this helps you avoid sending the message body if the message is rejected based on the headers (e.g., authentication failure or redirect). For more information on `100-continue`, see Section 8.2.3 of <http://www.ietf.org/rfc/rfc2616.txt>.

Example

Write some text and metadata into the "Neo" object in the "quotes" bucket:

Sample Request

```
PUT /Neo HTTP/1.1
x-amz-meta-family: Anderson
Content-Length: 4
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: text/plain
Expect: 100-continue
```

```
woah
```

Sample Response

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdmOdAiIfgSm/FlYsViT1LW94/xUQxMsF7xiEbla0wiIOIx1+zbwZ163pt7
x-amz-meta-family: Anderson
x-amz-request-id: 0A49CE4060975EAC
Date: Wed, 01 Mar 2006 12:00:00 GMT
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
Content-Length: 0
Connection: close
Server: AmazonS3
```

Request Headers

- *Cache-Control*: Can be used to specify caching behavior along the request/reply chain. See <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.9>.
- *Content-Type*: A standard MIME type describing the format of the contents. If none is provided, the default is binary/octet-stream. See <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.17>.
- *Content-Length*: The size of the object, in bytes. This is required. See <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.13>.
- *Content-MD5*:

The base64 encoded 128-bit MD5 digest of the message (without the headers) according to RFC 1864. This header can be used as a message integrity check to verify that the data is the same data that was originally sent. Although it is optional, we recommend using the Content-MD5 mechanism as an end-to-end integrity check. For more information about REST request authentication, see

Authenticating REST Requests.

- *Content-Disposition*: Specifies presentational information for the object. See <http://www.w3.org/Protocols/rfc2616/rfc2616-sec19.html#sec19.5.1>.
- *Content-Encoding*: Specifies what content codings have been applied to the object and thus what decoding mechanisms must be applied in order to obtain the media-type referenced by the Content-Type header field. See <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.11>.
- *Expires*: Gives the date/time after which the response is considered stale. See <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.21>.
- *x-amz-meta-*: Any header starting with this prefix is considered user metadata. It will be stored with the object and returned when you retrieve the object. The total size of the HTTP request, not including the body, must be less than 4 KB.
- *x-amz-acl-*: The canned ACL to apply to the object. Options include private, public-read, public-read-write, and authenticated-read. For more information, see ???.

Request Body

The data to be stored in the object is sent in the request body.

Response Headers

- *ETag*: The entity tag is an MD5 hash of the object that you can use to do conditional GET operations using the If-Modified request tag with the GET request operation.

Access Control

You must have WRITE access to the bucket to add an object.

GET Object

You fetch objects from Amazon S3 using the GET operation. This operation returns the object directly from Amazon S3 using a client/server delivery mechanism. If you want to distribute big files to a large number of people, you may find BitTorrent delivery to be preferable since it uses less bandwidth. For more information, see [Using BitTorrent™ with Amazon S3](#).

Example

Retrieve the "Nelson" object and its metadata from the "quotes" bucket:

Sample Request

```
GET /Nelson HTTP/1.1
```

```
Host: quotes.s3.amazonaws.com
```

```
Date: Wed, 01 Mar 2006 12:00:00 GMT
```

```
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Sample Response

```
HTTP/1.1 200 OK
```

```
x-amz-id-2: j5ULAWpFbJQJpukUsZ4tfXVOjVZExLtEyNTvY5feC+hHIegsN5p578JLTVpkFrpL
```

```
x-amz-request-id: BE39A20848A0D52B
Date: Wed, 01 Mar 2006 12:00:00 GMT
x-amz-meta-family: Muntz
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
Content-Type: text/plain
Content-Length: 5
Connection: close
Server: AmazonS3
```

HA-HA

Request Headers

The GET request method on objects supports several standard HTTP request headers. These are briefly described below, for details see the HTTP spec, RFC 2616 (<http://www.ietf.org/rfc/rfc2616.txt>).

- *If-Modified-Since*: Return the object only if it has been modified since the specified time, otherwise return a 304 (not modified).
- *If-Unmodified-Since*: Return the object only if it has not been modified since the specified time, otherwise return a 412 (precondition failed).
- *If-Match*: Return the object only if its entity tag (ETag) is the same as the one specified, otherwise return a 412 (precondition failed).
- *If-None-Match*: Return the object only if its entity tag (ETag) is different from the one specified, otherwise return a 304 (not modified).
- *Range*: Return only the bytes of the object in the specified range.

Response Headers

- *x-amz-meta-*: If you supplied user metadata when you PUT the object, that metadata is returned in one or more response headers prefixed with x-amz-meta- and with the suffix name that you provided on storage. This metadata is simply returned verbatim; it is not interpreted by Amazon S3.
- *Content-Type*: This is set to the same value you specified in the corresponding header when the data was PUT. The default content type is binary/octet-stream.
- *Content-Disposition*: This is set to the same value you specified in the corresponding header when the data was PUT. Except in the case of a request for a BitTorrent torrent file (see section on using BitTorrent with Amazon S3), if no Content-Disposition was specified at the time of PUT then this header is not returned.

- *Content-Range*: This indicates the range of bytes returned in the event that you requested a subset of the object by setting the Range request header.
- *x-amz-missing-meta*: This is set to the number of metadata entries not returned in x-amz-meta headers. This can happen if you create metadata using an API like SOAP that supports more flexible metadata than the REST API. For example, using SOAP, you can create metadata whose values are not legal HTTP headers.

Access Control

The GET will succeed if you have been granted READ access to the object. If you make a request without an authorization header, then you can read the object if READ access has been granted to the anonymous user.

Chunked and Resumable Downloads

To provide GET flexibility, Amazon S3 supports chunked and resumable downloads.

Select from the following:

- For large object downloads, you might want to break them into smaller chunks. For more information, see [Range GETs](#).
- For GET operations that fail, you can design your application to download the remainder instead of the entire file. For more information, see [REST GET Error Recovery](#).

Range GETs

For some clients, you might want to break large downloads into smaller downloads. To break a download into smaller units, use Range. For example, the following request downloads the first ten megabytes from the bigfile object.

```
GET /bigfile HTTP/1.1
Host: bigbucket.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
Range:bytes=0-10485759
```

Amazon S3 returns the following response:

```
HTTP/1.1 206 Partial Content
x-amz-id-2: j5ULAWpFbJQJpukUsZ4tfXVOjVZExLtEyNTvY5feC+hHIegsN5p578JLTVpkFrpL
x-amz-request-id: BE39A20848A0D52B
Date: Wed, 01 Mar 2006 12:00:00 GMT
x-amz-meta-family: Muntz
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
Content-Type: text/plain
```

Amazon Simple Storage Service Developer Guide

GET Object

```
Content-Length: 10485760
Content-Range: 0-10485759/20232760
Connection: close
Server: AmazonS3
```

<first 10 megabytes of bigfile>

Amazon S3 returns the first ten megabytes of the file, the Etag of the file, and the total size of the file (20232760 bytes) in the Content-Length field.

To ensure the file did not change since the previous portion was downloaded, specify the if-match request header. Although the if-match request header is not required, it is recommended for content that is likely to change.

The following request gets the remainder of the file, using the if-match request header:

```
GET /bigfile HTTP/1.1
Host: bigbucket.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
Range: 10485760-20232760
If-match: "828ef3fdfa96f00ad9f27c383fc9ac7f"
```

Amazon S3 returns the following response and the remainder of the file:

```
HTTP/1.1 206 Partial Content
x-amz-id-2: j5ULAWpFbJQJpukUsZ4tfXVOjVZExLtEyNTvY5feC+hHIegsN5p578JLTVpkFrpL
x-amz-request-id: BE39A20848A0D52B
Date: Wed, 01 Mar 2006 12:00:00 GMT
x-amz-meta-family: Muntz
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
Content-Type: text/plain
Content-Length: 9747000
Content-Range: 10485760-20232760/20232760
Connection: close
Server: AmazonS3
```

<remainder of bigfile>

REST GET Error Recovery

Amazon Simple Storage Service Developer Guide

GET Object

If an object GET fails, you can get the rest of the file by specifying the range to download. For example:

```
GET /bigfile HTTP/1.1
```

```
Host: bigbucket.s3.amazonaws.com
```

```
Date: Wed, 01 Mar 2006 12:00:00 GMT
```

```
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Amazon S3 returns the following response, but a client connection issue causes the GET to fail in the middle of the operation:

```
HTTP/1.1 200 OK
```

```
x-amz-id-2: j5ULAWpFbJQJpukUsZ4tfXVOjVZExLtEyNTvY5feC+hHIegsN5p578JLTVpkFrpL
```

```
x-amz-request-id: BE39A20848A0D52B
```

```
Date: Wed, 01 Mar 2006 12:00:00 GMT
```

```
x-amz-meta-family: Muntz
```

```
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
```

```
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
```

```
Content-Type: text/plain
```

```
Content-Length: 20232760
```

```
Connection: close
```

```
Server: AmazonS3
```

```
<part of bigfile>
```

The client code determines the amount of data downloaded and gets the rest.

```
GET /bigfile HTTP/1.1
```

```
Host: bigbucket.s3.amazonaws.com
```

```
Date: Wed, 01 Mar 2006 12:00:00 GMT
```

```
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

```
Range:bytes=132499-20232760
```

```
If-match:"828ef3fdfa96f00ad9f27c383fc9ac7f"
```

Amazon S3 returns the following response and the remainder of the file:

```
HTTP/1.1 206 Partial Content
```

```
x-amz-id-2: j5ULAWpFbJQJpukUsZ4tfXVOjVZExLtEyNTvY5feC+hHIegsN5p578JLTVpkFrpL
```

```
x-amz-request-id: BE39A20848A0D52B
```

```
Date: Wed, 01 Mar 2006 12:00:00 GMT
```

```
x-amz-meta-family: Muntz
```

```
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
Content-Type: text/plain
Content-Length: 20100261
Content-Range: 132499-20232760/20232760
Connection: close
Server: AmazonS3
```

<remainder of bigfile>

Related Topics

[GET Operation](#)

HEAD Object

The HEAD operation is used to retrieve information about a specific object, without actually fetching the object itself. This is useful if you're only interested in the object metadata, and don't want to waste bandwidth on the object data. A HEAD request has the same options as a GET operation on an object. The response is identical to the GET response, except that there is no response body. For more information, see [GET Object](#).

Example

Retrieve only the metadata for the "Nelson" object in the "quotes" bucket:

Sample Request

```
HEAD /Nelson HTTP/1.0
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-id-2: KZ7XUBI18rqFH91yZmYpWSRPg0/aeqwJXVzNgnk9Pa9GcHUuN2cxfsKk7V3NSUKg
x-amz-request-id: F7B5DF3AB381F03F
Date: Wed, 01 Mar 2006 12:00:00 GMT
x-amz-meta-family: Muntz
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
Content-Type: text/plain
Content-Length: 5
Connection: close
Server: AmazonS3
```

DELETE Object

The DELETE request operation removes the specified object from Amazon S3. Once deleted, there is no

method to restore or undelete an object.



Note

If you delete an object that does not exist, Amazon S3 will return a success (not an error message).

Example

Delete the "Nelson" object from the "quotes" bucket:

Sample Request

```
DELETE /Nelson HTTP/1.0
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: AWS 15B4D3461F177624206A:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Sample Response

```
HTTP/1.1 204 No Content
x-amz-id-2: 4NJT5+xl9kKL1w8YnhfDTbMPvBEIbl8Ek/kuX55i+4FTSINVbcRDVnhi4TZGcjly
x-amz-request-id: 7FA15BA5170D44B0
Date: Wed, 01 Mar 2006 12:00:00 GMT
Connection: close
Server: AmazonS3
```

Access Control

You can delete an object only if you have `WRITE` access to the bucket, regardless of who owns the object or what rights are granted to it.

Using the SOAP API

Topics

- [Common SOAP API Elements](#)
- [The SOAP Error Response](#)
- [Authenticating SOAP Requests](#)
- [Setting Access Policy with SOAP](#)
- [Operations on the Service](#)
- [Operations on Buckets](#)
- [Operations on Objects](#)

This section contains information specific to the Amazon S3 SOAP API.

Common SOAP API Elements

You can interact with Amazon S3 using SOAP 1.1 over HTTP. The Amazon S3 WSDL, which describes the Amazon S3 API in a machine-readable way, is available at: <http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl>. The Amazon S3 schema is available at <http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.xsd>.

Most users will interact with Amazon S3 using a SOAP toolkit tailored for their language and development environment. Different toolkits will expose the Amazon S3 API in different ways. Please refer to your specific toolkit documentation to understand how to use it. This section illustrates the Amazon S3 SOAP operations in a toolkit-independent way by exhibiting the XML requests and responses as they appear "on the wire."

SOAP Endpoint

You can send Amazon S3 SOAP messages to either a SSL secured or un-secured endpoint. Note that authenticated SOAP requests are only accepted over SSL. The available Amazon S3 SOAP endpoints are <http://s3.amazonaws.com/soap> and <https://s3.amazonaws.com/soap> (SSL).

Common Elements

You can include the following authorization-related elements with any SOAP request:

- *AWSAccessKeyId*: The AWS Access Key ID of the requestor.
- *Timestamp*: The current time on your system.
- *Signature*: The signature for the request.

The SOAP Error Response

In SOAP, an error result is returned to the client as a SOAP fault, with the HTTP response code 500. If you do not receive a SOAP fault, then your request was successful. The Amazon S3 SOAP fault code is comprised of a standard SOAP 1.1 fault code (either "Server" or "Client") concatenated with the Amazon S3-specific error code. For example: "Server.InternalError" or "Client.NoSuchBucket". The SOAP fault string element contains a generic, human readable error message in English. Finally, the SOAP fault detail element contains miscellaneous information relevant to the error.

For example, if you attempt to delete the object "Fred", which does not exist, the body of the SOAP response contains a "NoSuchKey" SOAP fault, which looks like:

Example

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
    <Faultstring>The specified key does not exist.</Faultstring>
    <Detail>
      <Key>Fred</Key>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

For more information about the errors, see [Working with Errors](#).

Authenticating SOAP Requests

Every non-anonymous request must contain authentication information to establish the identity of the principal making the request. In SOAP, the authentication information is put into the following elements of the SOAP request:

- *AWSAccessKeyId*: Your AWS Access Key ID
- *Timestamp*: This must be a dateTime (<http://www.w3.org/TR/xmlschema-2/#dateTime>) in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as 2006-01-01T12:00:00.000Z. Authorization will fail if this timestamp is more than 15 minutes away from the clock on Amazon S3 servers.
- *Signature*: The RFC 2104 HMAC-SHA1 digest (<http://www.ietf.org/rfc/rfc2104.txt>) of the concatenation of "AmazonS3" + OPERATION + Timestamp, using your AWS Secret Access Key as the key. For example, in the following CreateBucket sample request, the signature element would contain the HMAC-SHA1 digest of the value "AmazonS3CreateBucket2006-01-01T12:00:00.000Z":

For example, in the following CreateBucket sample request, the signature element would contain the HMAC-SHA1 digest of the value "AmazonS3CreateBucket2006-01-01T12:00:00.000Z":

Example

```
<CreateBucket xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Acl>private</Acl>
  <AWSAccessKeyId>1D9FVFRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-01-01T12:00:00.000Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```



Note

Authenticated SOAP requests must be sent to Amazon S3 over SSL. Only anonymous requests are allowed over non-SSL connections.



Important

Due to different interpretations regarding how extra time precision should be dropped, .NET users should take care not to send Amazon S3 overly specific time stamps. This can be accomplished by manually constructing `DateTime` objects with only millisecond precision.

For more information, see the sample [.NET SOAP libraries](#) for an example of how to do this.

Setting Access Policy with SOAP

Access control can be set at the time a bucket or object is written by including the "AccessControlList" element with the request to `CreateBucket`, `PutObjectInline`, or `PutObject`. The `AccessControlList` element is described in [Authentication and Access Control](#). If no access control list is specified with these operations, the resource is created with a default access policy that gives the requestor `FULL_CONTROL` access (this is the case even if the request is a `PutObjectInline` or `PutObject` request for an object that already exists).

The following sample request writes data to an object, makes the object readable by anonymous principals, and gives the specified user `FULL_CONTROL` rights to the bucket (Most developers will want to give themselves `FULL_CONTROL` access to their own bucket):

Example

The following sample request writes data to an object and makes the object readable by anonymous principals:

Sample Request

```
<PutObjectInline xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
```

```
<ContentLength>5</ContentLength>
<AccessControlList>
  <Grant>
    <Grantee xsi:type="CanonicalUser">
<ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
      <DisplayName>chriscustomer</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
    <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
    </Grantee>
    <Permission>READ</Permission>
  </Grant>
</AccessControlList>
<AWSAccessKeyId>1D9FVFRAYCP1VJEXAMPLE=</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

The access control policy can be read or set for an existing bucket or object using the `GetBucketAccessControlPolicy`, `GetObjectAccessControlPolicy`, `SetBucketAccessControlPolicy`, and `SetObjectAccessControlPolicy` methods. See the detailed explanation of these methods for more information.

Operations on the Service

Topics

- [ListAllMyBuckets](#)

This section describes operations you can perform on the Amazon S3 service.

ListAllMyBuckets

The `ListAllMyBuckets` operation returns a list of all buckets owned by the sender of the request.

Example

Sample Request

```
<ListAllMyBuckets xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <AWSAccessKeyId>1D9FVFRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
```

```
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListAllMyBuckets>
```

Sample Response

```
<ListAllMyBucketsResult xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Owner>
    <ID>bca1fffd86f41caff1a493dc2ad8c2c281e37522a640e161ca5fb16fd081034f</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes</Name>
      <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>samples</Name>
      <CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

Response Body

- *Owner*: This provides information that Amazon S3 uses to represent your identity for purposes of authentication and access control. ID is a unique and permanent identifier for the developer who made the request. DisplayName is a human-readable name representing the developer who made the request. It is not unique, and may change over time.
- *Name*: The name of a bucket. Note that if one of your buckets was recently deleted, the name of the deleted bucket may still be present in this list for a period of time.
- *CreationDate*: The time that the bucket was created.

Access Control

You must authenticate with a valid AWS Access Key ID. Anonymous requests are never allowed to list buckets, and you can only list buckets for which you are the owner.

Operations on Buckets

Topics

- [CreateBucket](#)
- [DeleteBucket](#)
- [ListBucket](#)
- [GetBucketAccessControlPolicy](#)
- [SetBucketAccessControlPolicy](#)
- [SetBucketLoggingStatus](#)
- [GetBucketLoggingStatus](#)

This section describes operations you can perform on Amazon S3 buckets.

CreateBucket

The `CreateBucket` operation creates a bucket. Not every string is an acceptable bucket name. For information on bucket naming restrictions, see [Working with Amazon S3 Buckets](#).

Example

Create a bucket named "quotes":

Sample Request

```
<CreateBucket xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>1D9FVRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

Sample Response

```
<CreateBucketResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CreateBucketResponse>
    <Bucket>quotes</Bucket>
  </CreateBucketResponse>
</CreateBucketResponse>
```

Elements

- *Bucket*: The name of the bucket you are trying to create.
- *AccessControlList*: The access control list for the new bucket. This element is optional. If not provided, the bucket is created with an access policy that give the requestor FULL_CONTROL access.

Access Control

You must authenticate with a valid AWS Access Key ID. Anonymous requests are never allowed to create buckets.

DeleteBucket

The `DeleteBucket` operation deletes a bucket. All objects in the bucket must be deleted before the bucket itself can be deleted.

Example

Delete the "quotes" bucket:

Sample Request

```
<DeleteBucket xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId> 1D9FVRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteBucket>
```

Sample Response

```
<DeleteBucketResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <DeleteBucketResponse>
    <Code>204</Code>
    <Description>No Content</Description>
  </DeleteBucketResponse>
</DeleteBucketResponse>
```

Elements

- *Bucket*: The name of the bucket you want to delete.

Access Control

Only the owner of a bucket is allowed to delete it, regardless the access control policy on the bucket.

ListBucket

The `ListBucket` operation returns information about some of the items in the bucket.

For a general introduction to the list operation, see the [Listing Keys](#) section.

Example

List up to 40 keys in the "quotes" bucket that have the prefix "N" and occur lexicographically after "Ned":

Sample Request

```
<ListBucket xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Prefix>N</Prefix>
  <Marker>Ned</Marker>
  <MaxKeys>40</MaxKeys>
  <AWSAccessKeyId>1D9FVRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Sample Response

```
<ListBucketResult xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Name>quotes</Name>
  <Prefix>N</Prefix>
  <Marker>Ned</Marker>
  <MaxKeys>40</MaxKeys>
  <IsTruncated>>false</IsTruncated>
  <Contents>
    <Key>Nelson</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <Size>5</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>bcaf1ffd86f41caff1a493dc2ad8c2c281e37522a640e161ca5fb16fd081034f</ID>
      <DisplayName>webfile</DisplayName>
    </Owner>
  </Contents>
  <Contents>
    <Key>Neo</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <Size>4</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>bcaf1ffd86f41caff1a493dc2ad8c2c281e37522a640e161ca5fb16fd081034f</ID>
      <DisplayName>webfile</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

Elements

For comprehensive information about the list request parameters, see [Common List Request Parameters](#).

- *Prefix*: Limits the response to keys that begin with the indicated prefix. You can use prefixes to separate a bucket into different sets of keys in a way similar to how a file system uses folders. This is an optional argument.
- *Marker*: Indicates where in the bucket to begin listing. The list includes only keys that occur alphabetically after marker. This is convenient for pagination: To get the next page of results use the last key of the current page as the marker. The most keys you'd like to see in the response body. The server may return less than this number of keys, but will not return more. This is an optional argument.
- *Delimiter*: Causes keys that contain the same string between the prefix and the first occurrence of the delimiter to be rolled up into a single result element in the CommonPrefixes collection. These rolled-up keys are not returned elsewhere in the response.
- *MaxKeys*: This optional argument limits the number of results returned in response to your query. Amazon S3 will return at most this number of results, but possibly less. For the purpose of counting MaxKeys, a 'result' is either a key in the 'Contents' collection, or a delimited prefix in the 'PrefixRollup' collection.

Response Body

For information about the list response, see [Common List Response Elements](#).

Access Control

To list the keys of a bucket you need to have been granted READ access on the bucket.

GetBucketAccessControlPolicy

The `GetBucketAccessControlPolicy` operation fetches the access control policy for a bucket.

Example

Retrieve the access control policy for the "quotes" bucket:

Sample Request

```
<GetBucketAccessControlPolicy xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>1D9FVFRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetBucketAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
<ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Response Body

The response contains the access control policy for the bucket. For an explanation of this response, see [Using Amazon S3](#).

Access Control

You must have `READ_ACP` rights to the bucket in order to retrieve the access control policy for a bucket.

SetBucketAccessControlPolicy

The `SetBucketAccessControlPolicy` operation sets the Access Control Policy for an existing

bucket. If successful, the previous Access Control Policy for the bucket is entirely replaced with the specified Access Control Policy.

Example

Give the specified user (usually the owner) FULL_CONTROL access to the "quotes" bucket.

Sample Request

```
<SetBucketAccessControlPolicy xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f2300e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  <AWSSignature>1D9FVFRAYCP1VJEXAMPLE=</AWSSignature>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</SetBucketAccessControlPolicy >
```

Sample Response

```
<GetBucketAccessControlPolicyResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <GetBucketAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </GetBucketAccessControlPolicyResponse>
</GetBucketAccessControlPolicyResponse>
```

Access Control

You must have WRITE_ACP rights to the bucket in order to set the access control policy for a bucket.

GetBucketLoggingStatus



Important

This document describes Beta functionality that is subject to change in future releases.

The `GetBucketLoggingStatus` retrieves the logging status for an existing bucket.

For a general introduction to this feature, see [Server Access Logging](#). For information about the response document, see [Server Access Logging Configuration API](#).

Example

Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xm
```

```
lns:xsi="http://www.w3.org/2001/XMLSchema-instance" xm
lns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <GetBucketLoggingStatus xm
lns="http://doc.s3.amazonaws.com/2006-03-01">
      <Bucket>mybucket</Bucket>
      <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
      <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
      <Signature>YOUR_SIGNATURE_HERE</Signature>
    </GetBucketLoggingStatus>
  </soap:Body>
</soap:Envelope>
```

Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xm
lns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xm
lns:xsd="http://www.w3.org/2001/XMLSchema" xm
lns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <GetBucketLoggingStatusResponse xm
lns="http://doc.s3.amazonaws.com/2006-03-01">
        <GetBucketLoggingStatusResponse>
          <LoggingEnabled>
            <TargetBucket>mylogs</TargetBucket>
            <TargetPrefix>mybucket-access_log</TargetPrefix>
          </LoggingEnabled>
        </GetBucketLoggingStatusResponse>
      </GetBucketLoggingStatusResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

Access Control

Only the owner of a bucket is permitted to invoke this operation.

SetBucketLoggingStatus



Important

This document describes Beta functionality that is subject to change in future releases.

The `SetBucketLoggingStatus` operation updates the logging status for an existing bucket.

For a general introduction to this feature, see [Server Access Logging](#). For information about the response document, see [Server Access Logging Configuration API](#).

Example

This sample request enables server access logging for the 'mybucket' bucket, and configures the logs to be delivered to 'mylogs' under prefix 'access_log-'

Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soap:Body>
      <SetBucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
        <Bucket>myBucket</Bucket>
        <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
        <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
        <Signature>YOUR_SIGNATURE_HERE</Signature>
        <BucketLoggingStatus>
          <LoggingEnabled>
            <TargetBucket>mylogs</TargetBucket>
            <TargetPrefix>mybucket-access_log</TargetPrefix>
          </LoggingEnabled>
        </BucketLoggingStatus>
      </SetBucketLoggingStatus>
    </soap:Body>
  </soap:Envelope>
```

Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <SetBucketLoggingStatusResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01"/>
    </soapenv:Body>
  </soapenv:Envelope>
```

Access Control

Only the owner of a bucket is permitted to invoke this operation.

Operations on Objects

Topics

- [PutObjectInline](#)
- [PutObject](#)
- [GetObject](#)
- [GetObjectExtended](#)
- [DeleteObject](#)
- [GetObjectAccessControlPolicy](#)
- [SetObjectAccessControlPolicy](#)

This section describes operations you can perform on Amazon S3 objects.

PutObjectInline

The `PutObjectInline` operation adds an object to a bucket. The data for the object is provided in the body of the SOAP message.

If the object already exists in the bucket, the new object overwrites the existing object. Amazon S3 orders all of the requests that it receives. It is possible that if you send two requests nearly simultaneously, we will receive them in a different order than they were sent. The last request received is the one which is stored in Amazon S3. Note that this means if multiple parties are simultaneously writing to the same object, they may all get a successful response even though only one of them wins in the end. This is because Amazon S3 is a distributed system and it may take a few seconds for one part of the system to realize that another part has received an object update. In this release of Amazon S3, there is no ability to lock an object for writing -- such functionality, if required, should be provided at the application layer.

`PutObjectInline` is not suitable for use with large objects. The system limits this operation to working with objects 1MB or smaller. `PutObjectInline` will fail with the `InlineDataTooLargeError` status code if the `Data` parameter encodes an object larger than 1MB. To upload large objects, consider using the non-inline `PutObject` API, or the REST API instead.

Example

Write some text and metadata into the "Nelson" object in the "quotes" bucket, give a user (usually the owner) `FULL_CONTROL` access to the object, and make the object readable by anonymous parties:

Sample Request

```
<PutObjectInline xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
<ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
      <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>1D9FVRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
```

```
<LastModified>2006-01-01T12:00:00.000Z</lastModified>
</PutObjectInlineResponse>
</PutObjectInlineResponse>
```

Elements

- *Bucket*: The bucket in which to add the object.
- *Key*: The key to assign to the object.
- *Metadata*: You can provide name-value metadata pairs in the metadata element. These will be stored with the object.
- *Data*: The base 64 encoded form of the data.
- *ContentLength*: The length of the data in bytes.
- *AccessControlList*: An Access Control List for the resource. This element is optional. If omitted, the requestor is given `FULL_CONTROL` access to the object. If the object already exists, the pre-existing access control policy is replaced.

Response

- *ETag*: The entity tag is an MD5 hash of the object that you can use to do conditional fetches of the object using `GetObjectExtended`.
- *LastModified*: The Amazon S3 timestamp for the saved object.

Access Control

You must have `WRITE` access to the bucket in order to put objects into the bucket.

PutObject

The `PutObject` operation adds an object to a bucket. The data for the object is attached as a `DIME` attachment.

If the object already exists in the bucket, the new object overwrites the existing object. Amazon S3 orders all of the requests that it receives. It is possible that if you send two requests nearly simultaneously, we will receive them in a different order than they were sent. The last request received is the one which is stored in Amazon S3. Note that this means if multiple parties are simultaneously writing to the same object, they may all get a successful response even though only one of them wins in the end. This is because Amazon S3 is a distributed system and it may take a few seconds for one part of the system to realize that another part has received an object update. In this release of Amazon S3, there is no ability to lock an object for writing -- such functionality, if required, should be provided at the application layer.

Example

Put some data and metadata in the "Nelson" object of the "quotes" bucket, give a user (usually the

owner) FULL_CONTROL access to the object, and make the object readable by anonymous parties. In this sample, the actual attachment is not shown:

Sample Request

```
<PutObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
<ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
      <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>1D9FVRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2007-05-11T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObject>
```

Sample Response

```
<PutObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <PutObjectResponse>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <LastModified>2006-03-01T12:00:00.183Z</LastModified>
  </PutObjectResponse>
</PutObjectResponse>
```

Elements

- *Bucket*: The bucket in which to add the object.
- *Key*: The key to assign to the object.
- *Metadata*: You can provide name-value metadata pairs in the metadata element. These will be stored with the object.
- *ContentLength*: The length of the data in bytes.
- *AccessControlList*: An Access Control List for the resource. This element is optional. If omitted, the requestor is given FULL_CONTROL access to the object. If the object already exists, the pre-existing Access Control Policy is replaced.

Response

- *ETag*: The entity tag is an MD5 hash of the object that you can use to do conditional fetches of the object using `GetObjectExtended`.
- *LastModified*: The Amazon S3 timestamp for the saved object.

Access Control

To put objects into a bucket, you must have `WRITE` access to the bucket.

GetObject

`GetObject` is the basic operation for retrieving an object stored in Amazon S3. For more options, use the [GetObjectExtended](#) operation.

Example

Get the "Nelson" object from the "quotes" bucket:

Sample Request

```
<GetObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <AWSAccessKeyId>1D9FVFRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Sample Response

```
<GetObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
```

```
</Metadata>
<Metadata>
  <Name>family</Name>
  <Value>Muntz</Value>
</Metadata>
<Data>aGEtaGE=</Data>
<LastModified>2006-01-01T12:00:00.000Z</LastModified>
<ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
</GetObjectResponse>
</GetObjectResponse>
```

Elements

- *Bucket*: The bucket from which to retrieve the object.
- *Key*: The key that identifies the object.
- *GetMetadata*: The metadata is returned with the object if this is true.
- *GetData*: The object data is returned if this is true.
- *InlineData*: If this is true, then the data is returned, base 64-encoded, as part of the SOAP body of the response. If false, then the data is returned as a SOAP attachment.

The *InlineData* option is not suitable for use with large objects. The system limits this operation to working with 1MB of data or less. A *GetObject* request with the *InlineData* flag set will fail with the *InlineDataTooLargeError* status code if the resulting *Data* parameter would have encoded more than 1MB. To download large objects, consider calling *GetObject* without setting the *InlineData* flag, or use the REST API instead.

Returned Elements

- *Metadata*: The name-value paired metadata stored with the object.
- *Data*: If *InlineData* was true in the request, this contains the base 64 encoded object data.
- *LastModified*: The time that the object was stored in Amazon S3.
- *ETag*: The object's entity tag. This is a hash of the object that can be used to do conditional gets.

Access Control

You can read an object only if you have been granted `READ` access to the object.

Chunked and Resumable Downloads

To provide GET flexibility, Amazon S3 supports chunked and resumable downloads.

Select from the following:

- For large object downloads, you might want to break them into smaller chunks. For more information, see [Range GETs](#).
- For GET operations that fail, you can design your application to download the remainder instead of the entire file. For more information, see [REST GET Error Recovery](#).

Range GETs

For some clients, you might want to break large downloads into smaller downloads. To break a GET into smaller units, use Range.

Before you can break a GET into smaller units, you must determine its size. For example, the following request gets the size of the bigfile object.

```
<ListBucket xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Prefix>bigfile</Prefix>
  <MaxKeys>1</MaxKeys>
  <AWSAccessKeyId>1D9FVFRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Amazon S3 returns the following response:

```
<ListBucketResult xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Name>quotes</Name>
  <Prefix>N</Prefix>
  <MaxKeys>1</MaxKeys>
  <IsTruncated>>false</IsTruncated>
  <Contents>
    <Key>bigfile</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <Size>2023276</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
```

```
<ID>bcaf1ffd86f41caff1a493dc2ad8c2c281e37522a640e161ca5fb16fd081034f</ID>
  <DisplayName>bigfile</DisplayName>
</Owner>
</Contents>
</ListBucketResult>
```

The following request downloads the first megabyte from the bigfile object.

```
<GetObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>0</ByteRangeStart>
  <ByteRangeEnd>1048576</ByteRangeEnd>
  <AWSAccessKeyId>1D9FVRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Amazon S3 returns the first megabyte of the file and the Etag of the file.

```
  <GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
<GetObjectResponse>
  <Status>
    <Code>200</Code>
    <Description>OK</Description>
  </Status>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
```

```
<Data>--first megabyte of bigfile--</Data>
<LastModified>2006-01-01T12:00:00.000Z</LastModified>
<ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
</GetObjectResponse>
</GetObjectResponse>
```

To ensure the file did not change since the previous portion was downloaded, specify the `IfMatch` element. Although the `IfMatch` element is not required, it is recommended for content that is likely to change.

The following request gets the remainder of the file, using the `IfMatch` request header:

```
<GetObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>>true</GetMetadata>
  <GetData>>true</GetData>
  <InlineData>>true</InlineData>
  <ByteRangeStart>10485761</ByteRangeStart>
  <ByteRangeEnd>2023276</ByteRangeEnd>
  <IfMatch>"828ef3fdfa96f00ad9f27c383fc9ac7f"</IfMatch>
  <AWSAccessKeyId>1D9FVFRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Amazon S3 returns the following response and the remainder of the file:

```
<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
```

```
<Metadata>
  <Name>family</Name>
  <Value>>Muntz</Value>
</Metadata>
<Data>--remainder of bigfile--</Data>
<LastModified>2006-01-01T12:00:00.000Z</LastModified>
<ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
</GetObjectResponse>
</GetObjectResponse>
```

REST GET Error Recovery

If an object GET fails, you can get the rest of the file by specifying the range to download. To do so, you must get the size of the object using `ListBucket` and perform a range GET on the remainder of the file. For more information, see [GetObjectExtended](#).

Related Topics

[Operations on Objects](#)

GetObjectExtended

`GetObjectExtended` is exactly like `GetObject`, except that it supports the following additional elements that can be used to accomplish much of the same functionality provided by HTTP GET headers (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>).

`GetObjectExtended` supports the following elements in addition to those supported by `GetObject`:

- *ByteRangeStart*, *ByteRangeEnd*: These elements specify that only a portion of the object data should be retrieved. They follow the behavior of the HTTP byte ranges (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35>).
- *IfModifiedSince*: Return the object only if the object's timestamp is later than the specified timestamp. (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.25>)
- *IfUnmodifiedSince*: Return the object only if the object's timestamp is earlier than or equal to the specified timestamp. (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.28>)
- *IfMatch*: Return the object only if its ETag matches the supplied tag(s). (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24>)
- *IfNoneMatch*: Return the object only if its ETag does not match the supplied tag(s). (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26>)
- *ReturnCompleteObjectOnConditionFailure*: `ReturnCompleteObjectOnConditionFailure`: If true, then if the request includes a range element and one or both of `IfUnmodifiedSince`/`IfMatch` elements, and the condition fails, return the entire object rather than a fault. This enables the `If-Range` functionality described here: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27>

DeleteObject

The `DeleteObject` operation removes the specified object from Amazon S3. Once deleted, there is no method to restore or undelete an object.



Note

If you delete an object that does not exist, Amazon S3 will return a success (not an error message).

Example

Delete the "Nelson" object from the "quotes" bucket:

Sample Request

```
<DeleteObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId> 1D9FVFRAYCP1VJEXAMPLE=</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteObject>
```

Sample Response

```
<DeleteObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <DeleteObjectResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </DeleteObjectResponse>
</DeleteObjectResponse>
```

Elements

- *Bucket*: The bucket that holds the object.
- *Key*: The key that identifies the object.

Access Control

You can delete an object only if you have `WRITE` access to the bucket, regardless of who owns the object or what rights are granted to it.

GetObjectAccessControlPolicy

The `GetObjectAccessControlPolicy` operation fetches the access control policy for an object.

Example

Retrieve the access control policy for the "Nelson" object from the "quotes" bucket:

Sample Request

```
<GetObjectAccessControlPolicy xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
```

```
<Key>Nelson</Key>
<AWSAccessKeyId>1D9FVFRAYCP1VJEXAMPLE=</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObjectAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Response Body

The response contains the access control policy for the bucket. For an explanation of this response, see [Using Amazon S3](#).

Access Control

You must have READ_ACP rights to the object in order to retrieve the access control policy for an object.

SetObjectAccessControlPolicy

The SetObjectAccessControlPolicy operation sets the access control policy for an existing object. If successful, the previous access control policy for the object is entirely replaced with the specified access control policy.

Example

Give the specified user (usually the owner) FULL_CONTROL access to the "Nelson" object from the "quotes" bucket:

Sample Request

```
<SetObjectAccessControlPolicy xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</SetObjectAccessControlPolicy>
```

```
</Grantee>
  <Permission>FULL_CONTROL</Permission>
</Grant>
</AccessControlList>
<AWSAccessKeyId>1D9FVRAYCP1VJEXAMPLE=</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</SetObjectAccessControlPolicy>
```

Sample Response

```
<SetObjectAccessControlPolicyResponse xm
lns="http://doc.s3.amazonaws.com/2006-03-01">
  <SetObjectAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </SetObjectAccessControlPolicyResponse>
</SetObjectAccessControlPolicyResponse>
```

Access Control

You must have `WRITE_ACP` rights to the object in order to set the access control policy for a bucket.

Using BitTorrent™ with Amazon S3

Topics

- [How You are Charged for BitTorrent Delivery](#)
- [Using BitTorrent to Retrieve Objects Stored in Amazon S3](#)
- [Publishing Content Using Amazon S3 and BitTorrent](#)

BitTorrent™ is an open, peer-to-peer protocol for distributing files, invented by Bram Cohen. You can use the BitTorrent protocol to retrieve any publicly-accessible object in Amazon S3. This section describes why you might want to use BitTorrent to distribute your data out of Amazon S3 and how to do so.

Amazon S3 supports the BitTorrent protocol so that developers can save costs when distributing content at high scale. Amazon S3 is useful for simple, reliable storage of any data. The default distribution mechanism for Amazon S3 data is via client/server download. In client/server distribution, the entire object is transferred point-to-point from Amazon S3 to every authorized user who requests that object. While client/server delivery is appropriate for a wide variety of use cases, it is not optimal for everybody. Specifically, the costs of client/server distribution increase linearly as the number of downloaders increases. This can make it expensive to distribute popular objects. BitTorrent addresses this problem by recruiting the very clients that are downloading the object as distributors themselves: Each client downloads some pieces of the object from Amazon S3 and some from other clients, while simultaneously uploading pieces of the same object to other interested "peers." The benefit for publishers is that for large, popular files the amount of data actually supplied by Amazon S3 can be substantially lower than what it would have been serving the same clients via client/server download. Less data transferred means lower costs for the publisher of the object.

How You are Charged for BitTorrent Delivery

There is no extra charge for use of BitTorrent with Amazon S3. Data transfer via the BitTorrent protocol is metered at the same rate as client/server delivery. To be precise, whenever a downloading BitTorrent client requests a "piece" of an object from the Amazon S3 "seeder," charges accrue just as if an anonymous request for that piece had been made using the REST or SOAP protocol. These charges will appear on your Amazon S3 bill and usage reports in the same way. The difference is that if a lot of clients are requesting the same object simultaneously via BitTorrent, then the amount of data Amazon S3 must serve to satisfy those clients will be lower than with client/server delivery. This is because the BitTorrent clients are simultaneously uploading and downloading amongst themselves.

The data transfer savings achieved from use of BitTorrent can vary widely depending on how popular your object is. Less popular objects require heavier use of the "seeder" to serve clients, and thus the difference between BitTorrent distribution costs and client/server distribution costs may be small for such objects. In particular, if only one client is ever downloading a particular object at a time, the cost of BitTorrent delivery will be the same as direct download.

Using BitTorrent to Retrieve Objects Stored in Amazon S3

Any object in Amazon S3 that can be read anonymously can also be downloaded via BitTorrent. Doing so requires use of a BitTorrent client application. Amazon does not distribute a BitTorrent client application, but there are many free clients available. The Amazon S3BitTorrent implementation has been tested to work with the official BitTorrent client, available at <http://www.bittorrent.com/>.

The starting point for a BitTorrent download is a .torrent file. This small file describes for BitTorrent clients both the data to be downloaded and where to get started finding that data. A .torrent file is a small fraction of the size of the actual object to be downloaded. Once you feed your BitTorrent client application an Amazon S3 generated .torrent file, it should start downloading immediately from Amazon S3 *and* from any "peer" BitTorrent clients.

Retrieving a .torrent file for any publicly available object is easy. Simply add a "?torrent" query string parameter at the end of the REST GET request for the object. No authentication is required. Once you have a BitTorrent client installed, downloading an object using BitTorrent download may be as easy as opening this URL in your web browser.

There is no mechanism to fetch the .torrent for an Amazon S3 object using the SOAP API.

Example

Retrieve the Torrent file for the "Nelson" object in the "quotes" bucket:

Sample Request

```
GET /quotes/Nelson?torrent HTTP/1.0
Date: Wed, 01 Mar 2006 12:00:00 GMT
```

Sample Response

```
HTTP/1.1 200 OK
x-amz-request-id: 7CD745EBB7AB5ED9
Date: Wed, 01 Mar 2006 12:00:00 GMT
Content-Disposition: attachment; filename=Nelson.torrent;
Content-Type: application/x-bittorrent
Content-Length: 537
Server: AmazonS3
```

<body: a Bencoded dictionary as defined by the BitTorrent specification>

Publishing Content Using Amazon S3 and BitTorrent

Every anonymously readable object stored in Amazon S3 is automatically available for download using

BitTorrent. The process for changing the ACL on an object to allow anonymous READ operations is described in [Authentication and Access Control](#).

You can direct your clients to your BitTorrent accessible objects by giving them the .torrent file directly or by publishing a link to the ?torrent URL of your object. One important thing to note is that the .torrent file describing an Amazon S3 object is generated on-demand, the first time it is requested (via the REST ?torrent resource). Generating the .torrent for an object takes time proportional to the size of that object. For large objects, this time can be significant. Therefore, before publishing a ?torrent link, we suggest making the first request for it yourself. Amazon S3 may take several minutes to respond to this first request, as it generates the .torrent file. Unless you update the object in question, subsequent requests for the .torrent will be fast. Following this procedure before distributing a ?torrent link will ensure a smooth BitTorrent downloading experience for your customers.

To stop distributing a file using BitTorrent, simply remove anonymous access to it. This can be accomplished by either deleting the file from Amazon S3, or modifying your access control policy to prohibit anonymous reads. After doing so, Amazon S3 will no longer act as a "seeder" in the BitTorrent network for your file, and will no longer serve the .torrent file via the ?torrent REST API. However, once a .torrent for your file has been published, this action may not be sufficient to stop public downloads of your object that happen exclusively using the BitTorrent peer to peer network.

Glossary

account	AWS account associated with a particular developer.
bucket	A container for objects stored in Amazon S3. Every object is contained within a bucket. For example, if the object named <code>photos/puppy.jpg</code> is stored in the <code>johnsmith</code> bucket, then it is addressable using the URL <code>http://johnsmith.s3.amazonaws.com/photos/puppy.jpg</code>
consistency model	The method through which Amazon S3 achieves high availability, which involves replicating data across multiple servers within Amazon's data centers. After a "success" is returned, your data is safely stored. However, information about the changes might not be immediately replicated across Amazon S3.
key	The unique identifier for an object within a bucket. Every object in a bucket has exactly one key. Since a bucket and key together uniquely identify each object, Amazon S3 can be thought of as a basic data map between "bucket + key" and the object itself. Every object in Amazon S3 can be uniquely addressed through the combination of the Service endpoint, bucket name, and key, as in <code>http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl</code> , where "doc" is the name of the bucket, and "2006-03-01/AmazonS3.wsdl" is the key.
metadata	The metadata is 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. The developer can also specify custom metadata at the time the Object is stored.
object	The fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The data portion is opaque to Amazon S3.
service endpoint	The host and port with which you are trying to communicate within the destination URL. For virtual hosted-style requests, this is <code>mybucket.s3.amazonaws.com</code> . For path-style requests, this is <code>s3.amazonaws.com</code>

Document Conventions

This section lists the common typographical and symbol use conventions for AWS technical publications.

Typographical Conventions

This section describes common typographical use conventions.

Convention	Description/Example
Call-outs	A call-out is a number in the body text to give you a visual reference. The reference point is for further discussion elsewhere. You can use this resource regularly. 1
Code in text	Inline code samples (including XML) and commands are identified with a special font. You can use the command <code>java -version</code> .
Code blocks	Blocks of sample code are set apart from the body and marked accordingly. <pre># ls -l /var/www/html/index.html -rw-rw-r-- 1 root root 1872 Jun 21 09:33 / var/www/html/index.html # date Wed Jun 21 09:33:42 EDT 2006</pre>
Emphasis	Unusual or important words and phrases are marked with a special font. You <i>must</i> sign up for an account before you can use the service.
Internal cross references	References to a section in the same document are marked. See Document Conventions .
Logical values,	A special font is used for expressions that are important to identify, but are

Amazon Simple Storage Service Developer Guide

Typographical Conventions

Convention	Description/Example
constants, and regular expressions, abstracta	not code. If the value is <code>null</code> , the returned response will be <code>false</code> .
Product and feature names	Named AWS products and features are identified on first use. Create an <i>Amazon Machine Image</i> (AMI).
Operations	In-text references to operations. Use the <code>GetHITResponse</code> operation.
Parameters	In-text references to parameters. The operation accepts the parameter <i>AccountID</i> .
Response elements	In-text references to responses. A container for one <code>CollectionParent</code> and one or more <code>CollectionItems</code> .
Technical publication references	References to other AWS publications. If the reference is hyperlinked, it is also underscored. For detailed conceptual information, see the <i>Amazon Mechanical Turk Developer Guide</i> .
User entered values	A special font marks text that the user types. At the password prompt, type MyPassword .
User interface controls and labels	Denotes named items on the UI for easy identification. On the File menu, click Properties .
Variables	When you see this style, you must change the value of the content when you copy the text of a sample to a command line. <code>% ec2-register <your-s3-bucket>/image.manifest</code> See also the symbol convention below.

Symbol Conventions

This section describes the common use of symbols.

Convention	Symbol	Description/Example
Mutually exclusive parameters	(Parentheses and vertical bars)	Within a code description, bar separators denote options from which one must be chosen. <code>% data = hdfread (start stride edge)</code>
Optional parameters XML variable text	[square brackets]	Within a code description, square brackets denote completely optional commands or parameters. <code>% sed [-n, -quiet]</code> Use square brackets in XML examples to differentiate them from tags. <code><CustomerId>[ID]</CustomerId></code>
Variables	<arrow brackets>	Within a code sample, arrow brackets denote a variable that must be replaced with a valid value. <code>% ec2-register <your-s3-bucket>/image.manifest</code>

Index

Symbols

100-continue, 75

A

access control, 21
 access logs, 49
 access policy
 REST, 70
 SOAP, 118
 Adobe Flash, 77
 Amazon DevPay, 37
 API, 8
 REST, 9, 57
 SOAP, 9, 116
 audience, 3
 authentication, 21
 debugging, 68
 REST, 60
 SOAP, 117
 authentication header, 61

B

billing, 10
 BitTorrent, 140
 charges, 140
 publishing, 141
 retrieving objects, 141
 browser uploads, 77
 bucket location, get
 get, 100
 buckets, 7, 11
 access control, 14
 billing, 14
 configuration, 13
 creating, 121
 location selection, 13
 putting, 97
 REST operations, 96
 restrictions, 12
 SOAP operations, 120
 virtual hosting, 71

C

CanonicalizedAmzHeaders element, 63
 changes, 1
 charges, 10
 chunked downloads, 21
 components, 7
 concepts, 7
 API, 8
 buckets, 7
 components, 7

 keys, 8
 objects, 8
 operations, 8
 REST API, 9
 SOAP API, 9
 configuring logging, 47
 consistency model, 9
 costs, 10
 CreateBucket
 SOAP, 121
 creating buckets
 SOAP, 121

D

data model, 9
 DELETE
 REST, 100, 114
 SOAP, 121
 DeleteObject
 SOAP, 136
 deleting objects
 REST, 114
 SOAP, 136
 delimiter, 16, 19
 DevPay, 37
 DNS, 36
 DNS routing, 31, 33, 35
 downloads, chunked and resumable, 21

E

elements
 REST, 58
 SOAP, 116
 errors, 41
 details, 42
 isolation, 41
 list, 42
 messages, 42
 response, 41
 responses, 40
 REST response, 59
 SlowDown, 41
 SOAP response, 117

F

features, 6
 file size, maximum, 6
 Flash, Adobe, 77

G

GET
 object, REST, 109
 REST, 98, 100
 get bucket location
 REST, 100
 GET policy

REST, 95
 GetBucketAccessControlPolicy
 SOAP, 124
 GetBucketLoggingStatus
 SOAP, 125
 GetObject
 SOAP, 131
 GetObjectAccessControlPolicy
 SOAP, 137
 GetObjectExtended
 SOAP, 136
 glossary, 143
 guide organization, 4

H

HEAD object
 REST, 114
 HTTP user agents, 75

I

introduction, 6
 IsTruncated, 16

J

Java, 36
 JVM cache, 36

K

keys, 8
 listing, 15
 listing hierarchically, 19
 multi-page results, 18
 request parameters, 16
 responses, 16
 using, 15

L

ListAllMyBuckets
 SOAP, 119
 ListBucket
 SOAP, 122
 listing keys, hierarchically, 19
 location constraints, 13
 logs, 46
 best effort delivery, 50
 changing settings, 49
 configuration, 47
 delivery, 49
 format, 50
 setting up, 53

M

marker, 16
 MaxKeys, 16
 metadata, using, 19

model, 9

N

NextMarker, 16

O

object size, maximum, 6
 objects, 8
 getting, 20
 REST operations, 107
 SOAP operations, 127
 using, 14
 operations, 8
 organization of guide, 4
 overview, 6

P

pagination, 18
 paying, 10
 performance optimization, 36
 PHP virtual machine, 36
 POST, 77, 101
 prefix, 16, 19
 PUT
 REST, 97
 PUT, REST, 107
 PutObject
 SOAP, 129
 PutObjectInline
 SOAP, 127
 putting buckets
 REST, 97

R

redirection, 74
 permanent, 35
 request, 31
 temporary, 33
 referer, 50
 request redirection, 31
 access policy, 74
 request routing, 31
 resources, related, 4
 REST
 access policy, 70
 API, 57
 authentication, 60
 examples, 64
 header, 61
 bucket operations, 96
 debugging authentication, 68
 DELETE, 100
 DELETE objects, 114
 elements, 58
 GET, 95, 98, 100
 GET object, 109

- HEAD object, 114
- object operations, 107
- POST, 77, 101
- PUT, 97
- PUT object, 107
- service operations, 95
- StringToSign, 63
- time stamp, 63
- restrictions, 12
- resumable downloads, 21
- routing, 31
 - DNS, 31

S

- server access logs, 46, 49
- service
 - REST operations, 95
 - SOAP operations, 119
- SetBucketAccessControlPolicy
 - SOAP, 124
- SetBucketLoggingStatus
 - SOAP, 126
- SetObjectAccessControlPolicy
 - SOAP, 138, 140
- size, object, 6
- SOAP
 - access policy, 118
 - API, 116
 - authentication, 117
 - bucket operations, 120
 - CreateBucket, 121
 - DELETE, 121
 - DELETE objects, 136
 - elements, 116
 - error response, 117
 - GetBucketAccessControlPolicy, 124
 - GetBucketLoggingStatus, 125
 - GetObject, 131
 - GetObjectAccessControlPolicy, 137
 - GetObjectExtended, 136
 - ListAllMyBuckets, 119
 - ListBucket, 122
 - object operations, 127
 - PutObject, 129
 - PutObjectInline, 127
 - SetBucketAccessControlPolicy, 124
 - SetBucketLoggingStatus, 126
 - SetObjectAccessControlPolicy, 138
- SOPA
 - service operations, 119
- storage limit, 6
- StringToSign, 63
- system metadata, 19

T

- TCP optimization, 36
- time stamp, 63

- TTLs, clients, 36

U

- uploads, browser, 77
- user metadata, 19
- using Amazon S3, 11

V

- virtual hosted buckets, 71
- virtual machines, 36