Amazon Web Services

Metadata Tutorial

Created by Zarsha Mian

Amazon Web Services

Table of Contents

What is Metadata?

Every Amazon S3 object has data, a key, and metadata. Object metadata refers to a set of name-value pairs. Metadata is data that describes data; since it is not the actual data itself, it can be safely made public because it provides summary details about an object. Metadata is added to objects, like text files, and provides important information about the contents of the file. It can be used to distinguish and identify objects based on their contents, help users find relevant information, and aid in the organization of valuable resources. There are two different kinds of metadata, known as system metadata and user-defined metadata

System Metadata

For each object stored in a bucket, Amazon S3 maintains a set of system metadata, which is processed as needed for object management. System metadata has two categories:

1. System controlled: system metadata where only Amazon S3 can modify the value. An example of this would be object creation date, which would contain the date that the object was created.
2. User controlled: system metadata where the user can control the value. For example, if you configure your bucket as a website and choose to redirect a page request to another page or external URL, the page redirect value (a webpage object found in your bucket) would be stored by Amazon S3 as system metadata whose value you control. The storage class configured for an object and whether or not the object has server-side encryption enabled are other examples of user-controlled system metadata. When you create objects, you are able to configure values for these system metadata items or update the values when you need to.

The values of the following system-defined metadata are user controlled and can be modified by users:

1. Content-Type: object type.
2. Content-MD5: the Base64-encoded 128-bit MD5 digest of the object.
3. X-amz-server-side-encryption: indicates whether server-side encryption is enabled for the object, and whether that encryption is from the AWS Key Management Service (AMS KMS) or from Amazon S3 managed encryption (SSE-S3).

4. X-amz-storage-class: storage class used for storing the object.
5. X-amz-website-redirect-location: redirects requests for the associated object to another object in the same bucket or an external URL.
6. X-amz-server-side-encryption-aws-kms-key-id: if x-amz-server-side-encryption is present and has a value of aws:kms, this indicates the ID of the AWS KMS symmetric customer master key (CMK) that was used for the object.

The values of the following system-defined metadata are system controlled and can NOT be modified by users:

1. Date: current date and time.
2. Content-Length: object size in bytes.
3. Last-Modified: object creation or the last modified date, whichever is the latest and most current.
4. X-amz-version-id: object version. When you enable versioning on a bucket, Amazon S3 assigns a version number to objects added to the bucket.
5. X-amz-delete-marker: in a bucket that has versioning enabled, this marker is a Boolean value that indicates whether or not the object is a delete marker.

User-Defined Object Metadata

When uploading an object, you can assign metadata to it. You can provide this optional information as a name-value (key-value) pair when you send a POST or PUT request to create the object. When uploading objects using the REST API, the user-defined metadata names must begin with "x-amz-meta-" to distinguish them from other HTTP headers.

When metadata is retrieved through the REST API, the prefix "x-amz-meta" is returned and 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. Rather, the "x-amz-missing-meta" header is returned with a value of the number of unprintable metadata entries. As mentioned earlier, user-defined metadata is a set of key-value pairs, and these user-defined metadata keys are stored by Amazon S3 in lowercase and allows arbitrary Unicode characters in your metadata values.

To avoid issues around the presentation of the metadata values, use US-ASCII characters when using REST and UTF-8 when using browser-based uploads via POST. When using non US-ASCII characters in your metadata values, the provided Unicode string is examined for non US-ASCII characters. If the string contains on US-ASCII characters, it is presented as is. Otherwise, if the string contains non US-ASCII characters, it is first character-encoded using UTF-8 and then encoded into US-ASCII.

The PUT request header is limited to 8KB in size, and within the PUT request header, the user-defined metadata is limited to 2KB in size. The size of user-defined metadata is measured by taking the sum of the number of bytes in the UTF-8 encoding of each key and value.

Browser-Based Upload Using POST and AWS Signature Version 4

A POST policy is the security policy that describes what is permitted in a request. A POST policy must include certain fields to provide relevant information that Amazon S3 can use to re-calculate the signature upon receiving the request. To learn more about what a POST policy is and how it works, please refer to the Creating a Post Policy section of the Amazon Web Services tutorial.

In this example, we will be creating a POST request to upload a text file to an Amazon S3 bucket. The text file contains the names of famous contemporary authors, and is saved on your computer by the name authors.txt. The metadata associated with the file needs to correlate with the upload. Metadata for a document could include information such as the file's author, size, or keywords to describe the document. In our example, we'll use a metadata key called subject, which will be set to authors and will automatically apply to every valid upload. We will also include a unique metadata tag, allowing the user to enter in the value.

To learn more about AWS Signature Version 4, which is used to permit this upload, please refer to the Signature Version 4 section of the Amazon Web Services tutorial for a detailed explanation.

Step 1: POST Policy

Below is a sample POST policy that sets conditions for the Amazon S3 upload and includes additional metadata that describes the upload. This example POST policy has subject metadata and a metadata tag associated with the uploaded file. It is important to replace the default input (marked in red) with your own input and conditions for your unique POST policy.

```
{ "expiration": "2021-12-30T12:00:00.000Z",
  "conditions": [
    {"bucket": "example.com"},
    ["starts-with", "$key", "user/user1/"],
    {"acl": "public-read"},
    {"success_action_redirect":
"http://example.com.s3.amazonaws.com/upload_success.html"},
    ["starts-with", "$Content-Type", "text/plain"],
    {"x-amz-meta-subject": "authors"},
    {"x-amz-server-side-encryption": "AES256"},
    ["starts-with", "$x-amz-meta-tag", ""],

    {"x-amz-credential": " AKIAIOSFODNN7EXAMPLE/20200519/us-east-
2/s3/aws4_request"},
    {"x-amz-algorithm": "AWS4-HMAC-SHA256"},
    {"x-amz-date": "20200519T000000Z" }
  ]
}
```

The POST policy sets the following conditions on the request:

- The upload must occur before noon UTC on December 30th, 2021.
- The content can be uploaded only to the bucket named example.com. NOTE: The bucket must be available in the region specified in the credential scope. The signature you provide is valid only within this scope.
- You can provide any key name that starts with user/user1. For example, a valid upload would have a key name such as user/user1/authors.txt. **NOTE:** If you want to be the only person who can use the HTML form and the policy associated with it to upload directly into your bucket, you could set this value to the name of the user account that you use on your computer, such as user/*your-name*.
- The ACL must be set to public-read. This grants the public read permissions.
- If the upload succeeds, the user's browser will be redirected to http://example.com.s3.amazonaws.com/upload_success.html.
- The object must be a plain text file.

- The x-amz-meta-subject key must be set to authors. NOTE: The text file we are using in this example contains the name of famous authors. If the text file contained information about something else, the subject value would relate to the content within the text file, and would need to be set inside of your POST policy in order to apply to all of the valid uploads. When a file is uploaded using the POST request, the metadata key-value pair will be associated with the uploaded text file.
- The x-amz-meta-tag can be any value. NOTE: You can store any value, and it will be associated with the file uploaded. A tag should contain information related to the contents of the file. For example, we can use it to describe the genre of the authors mentioned in the file, which in this case is contemporary.
- The x-amz-credential contains your access key ID, the date, region, and service. The service for our purposes is S3. The credential is a string in the following form:

  <your-access-key-id>/<date>/<aws-region>/<aws-service>/aws4_request
- The signing algorithm for AWS Signature Version 4 is AWS4-HMAC-SHA256. This value must be provided when AWS Signature Version 4 is used for authentication.
- The date in x-amz-date, which is in ISO8601 format, must match the date in the credential parameter. This is the same date used in creating the signing key.

NOTE: When creating a POST policy, it is important to replace the example access key ID mentioned above with your user's access key ID.

Step 2: Base64-Encode, or String to Sign

Once the POST policy is created, the next step is to encode the POST policy into string to sign. One way to encode the POST policy into string to sign is to use an online encoder, such as the one found at www.base64encode.org. To achieve the string to sign result used in this tutorial, which is also re-calculated by Amazon S3, be sure to set the Destination character set value to UTF-8 and the Destination newline separator to CRLF (Windows) on the website. The Base64-encoded version changes based on these settings, and we need to use the string to sign that Amazon will calculate and match to our value in order to accept our request.

The Base64-encoded version of the POST policy above (without changing the default settings, like the credential parameter, to your own personalized values) is the following:

eyAiZXhwaXJhdGlvbiI6ICIyMDIxLTEyLTMwVDEyOjAwOjAwLjAwMFoiLA0KICAiY
29uZGl0aW9ucyI6IFsNCiAgICB7ImJ1Y2tldCI6ICJleGFtcGxlLmNvbSJ9LA0KICAgIFsi
c3RhcnRzLXdpdGgiLCAiJGtleSIsICJ1c2VyL3VzZXIxLyJdLA0KICAgIHsiYWNsIjogIn
B1YmxpYy1yZWFkIn0sDQogICAgeyJzdWNjZXNzX2FjdGlvbl9yZWRpcmVjdCI6ICJod
HRwOi8vZXhhbXBsZS5jb20uczMuYW1hem9uYXdzLmNvbS91cGxvYWRfc3VjY2Vzcy5
odG1sIn0sDQogICAgWyJzdGFydHMtd2l0aCIsICIkQ29udGVudC1UeXBlIiwgInRleHQv
cGxhaW4iXSwNCiAgICB7IngtYW16LW1ldGEtc3ViamVjdCI6ICJhdXRob3JzIn0sDQogI
CAgeyJ4LWFtei1zZXJ2ZXItc2lkZS1lbmNyeXB0aW9uIjogIkFFUzI1NiJ9LA0KICAgIFsic
3RhcnRzLXdpdGgiLCAiJHgtYW16LW1ldGEtdGFnIiwgIiJdLA0KDQogICAgeyJ4LWFt
ei1jcmVkZW50aWFsIjogIiBBS0lBSU9TRk9ETk43RVhBTVBMRS8yMDIwMDUxOS91c
y1lYXN0LTIvczMvYXdzNF9yZXF1ZXN0In0sDQogICAgeyJ4LWFtei1hbGdvcml0aG0iOi
AiQVdTNC1ITUFDLVNIQTI1NiJ9LA0KICAgIHsieC1hbXotZGF0ZSI6ICIyMDIwMDU
xOVQwMDAwMDBaIiB9DQogIF0NCn0=

Step 3: Signature Calculation

After the POST policy is encoded to string to sign, the signature must be calculated since AWS Signature Version 4 is being used. First, a signing key is derived from your AWS secret access key; this derived signing key is specific to the date, service, and region, which means it offers a greater degree of protection. The signing key and Base64-encoded string to sign previously created are then used as inputs to a keyed hash function. The hex-encoded result from the keyed hash function is the signature. The following Ruby code generates the signature:

```
require 'uri'
require 'openssl'
require 'net/http'
require 'cgi'

def getSignatureKey(key, dateStamp, regionName, serviceName)
  kDate    = OpenSSL::HMAC.digest('sha256', "AWS4" + key, dateStamp)
  kRegion  = OpenSSL::HMAC.digest('sha256', kDate, regionName)
  kService = OpenSSL::HMAC.digest('sha256', kRegion, serviceName)
  kSigning = OpenSSL::HMAC.digest('sha256', kService, "aws4_request")
  kSigning
end

secret_key = 'enter your secret access key here in single quotations'
datestamp = '20200519' # enter date from policy in YYYYMMDD format
region = 'us-east-2' # enter region from policy here
service = 's3'

string_to_sign = 'enter your string to sign value here in single quotations'

signing_key = getSignatureKey(secret_key, datestamp, region, service)
```

```
signature = OpenSSL::HMAC.hexdigest('sha256', signing_key, string_to_sign)
print signature
```

Remember, the signature in this tutorial is calculated while using example credentials. Your signature will be different based on user credentials. These example calculations can be used to verify your calculations. The signature value for this example is as follows:

58ac16dd732982a9a9f315bd8b189aa50eacef170ec96996a395969ba1cd0adb

Step 4: HTML Form

After the POST policy is created, encoded into string to sign, and a signature is calculated, an HTML form must be created that specifies the preceding POST policy and supports a POST request to the bucket, in this case example.com. The following HTML form would enable users to upload text files to the bucket example.com. Your request will succeed if the signature you provide matches the signature calculated by Amazon S3. Copy and paste the content below into a text editor, and replace the default example values with your own personalized values. Then, save as an HTML file. You can save it with a title such as upload_form.html.

There are two metadata fields in this form:

- X-amz-meta-subject: describes what the file is about. In this case, the file is a list of famous authors, so the subject's value is set to authors.
- X-amz-meta-tag: the metadata tag, which can hold any value. The value entered into this field during the time of upload will then be associated with the uploaded file inside of the bucket.

```
<html>
  <head>

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

  </head>
  <body>

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

```
   <input type="input"  name="key" value="user/user1/${filename}" /><br />
   <input type="hidden" name="acl" value="public-read" />
   <input type="hidden" name="success_action_redirect"
value="http://example.com.s3.amazonaws.com/upload_success.html" />
   Content-Type:
   <input type="input"  name="Content-Type" value="text/plain" /><br />
   <input type="input" name="x-amz-meta-subject" value="authors" />
   <input type="hidden" name="x-amz-server-side-encryption" value="AES256" />
   <input type="hidden"   name="X-Amz-Credential" value="
AKIAIOSFODNN7EXAMPLE/20200519/us-east-2/s3/aws4_request" />
   <input type="hidden"   name="X-Amz-Algorithm" value="AWS4-HMAC-SHA256" />
   <input type="text"   name="X-Amz-Date" value="20200519T000000Z" />


   Tags for File:
   <input type="input"  name="x-amz-meta-tag" value="" /><br />
   <input type="hidden" name="Policy" value='<Base64-encoded policy string>' />
   <input type="hidden" name="X-Amz-Signature" value="<signature-value>" />
   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>
```

NOTE: Be sure to enter in your string to sign value in the Policy field and your calculated signature in the Signature field. Make any other additional modifications by replacing default values with your personalized conditions.

Making Upload Form Available to Users

Once your HTML form is ready, upload it into your Amazon S3 bucket by selecting the name of your bucket in the Amazon S3 console's bucket list, and pressing the upload button. Select the HTML form file you wish to upload, set the permissions and properties by following the prompts on your screen, review your settings, and then press Upload. For more information about uploading content into your Amazon S3 bucket, please refer to the Uploading Index and Website Content section of the Amazon Web Services tutorial.

When the HTML form file is in your bucket, provide a link to your website's users so that they can use the form to upload objects (in this case, text files) into your bucket. To do this, create a link on one of your website's pages to the form. For example, you can use the following line somewhere inside your website's HTML page files:

<p><a href="upload_form.html">Upload to S3 Form</a></p>

Using HTML Form to Upload Text Files and Assign Metadata

When users navigate to the HTML form on your website that allows them to upload objects into your bucket, they will see metadata fields based on the conditions mentioned in the POST policy. Since the x-amz-meta-subject key was assigned a value of authors, the value will be automatically entered in the subject metadata text field. However, since the x-amz-meta-tag was assigned no value, it can be given any value by the user. For this example, we decided to enter the genre of the authors as a metadata tag for the uploaded text file. The word contemporary can be entered in the empty text field for the metadata tag.

There will be a Choose File button that users will click to choose which text file they wish to upload into the bucket using the form. Once the Upload to Amazon S3 button is pressed by the user, the object will be uploaded into the bucket if all conditions are satisfied and the signature calculated by Amazon S3 matches the signature you calculated and mentioned in your HTML form.

Seeing Metadata Assigned to Objects in Bucket

After metadata is assigned to an object and it is uploaded into the bucket using the HTML form mentioned earlier, you can confirm that the metadata entered during the time of upload is in fact associated with the specific object. In this case, you would be confirming that the value authors was assigned to the subject metadata key and that the metadata tag was assigned a value relating to the text file's contents, in this case the word contemporary to describe the genre of the authors. To see the metadata associated with the text file that was uploaded, follow these steps:

1. Sign in to the **AWS Management Console** and open the **Amazon S3 console**.
2. In the **Buckets** section, click on the *name* of the bucket that contains the object, the text file that was uploaded.
3. In the **Overview** tab, click the *name* of the object that contains the metadata. Based on the information mentioned in the POST policy and HTML form that we used for the browser-based upload, the file should be found by clicking the *users* folder,

clicking the folder with the *user's name* (in this case, user1), and clicking the *name* of the text file that was uploaded (in this case, authors.txt).

4. Under the name of the object, click on the **Properties** tab.
5. Click *Metadata* to expand the metadata property section. The metadata associated with the file (Content-Type, x-amz-meta-subject, and x-amz-meta-tag) will all be listed with their assigned values.

NOTE: Once the object is uploaded into your bucket, you will be able to edit the metadata values assigned to the text file. Since the value authors was assigned to the subject key in the POST policy and HTML form, it was automatically filled into the text field and no other value would have been accepted during the time of upload. However, it is possible to change this value without any problems after it is already in your bucket should you wish to do so.

Adding User-Defined Metadata to an S3 Object

Metadata is additional information about an object in Amazon S3. Users can assign user-defined metadata to an object when uploading the object or after the object has been uploaded. This user-defined metadata is stored with the object and is returned when the object is downloaded. Amazon S3 does not process user-defined metadata.

You can assign user-defined data to an object that is already stored in a bucket. To do so, follow these steps:

1. Sign in to the **AWS Management Console** and open the **Amazon S3 console**.
2. In the **Buckets** section, click on the *name* of the bucket that contains the object.
3. In the **Overview** tab, click the *name* of the object that you would like to add metadata to.
4. Under the name of the object, click on the **Properties** tab.
5. Click *Metadata* to expand the metadata property section.
6. Click *Add Metadata*. The section will expand to show New Metadata, as well as any existing metadata already associated with the object.
7. Click *Select a key* to open a dropdown menu containing keys.
8. Choose the *x-amz-meta* key from the dropdown menu. Any metadata starting with x-amz-meta is user-defined metadata.

9. In the *left box* under Key, following the x-amz-meta- key, type a *custom name* to set as the metadata key. For example, for the custom name alternative-name, the metadata key would be x-amz-meta-alternative-name.
10. In the *right box* under Value, type in the *value* for the custom key. For example, the value for the alternative-name key could be sample-picture.
11. Click the *Save* button. The metadata property section will close and under Metadata, the number of metadata associated with the object will be shown.

Users also have the option to edit or delete metadata associated with an object stored in a bucket. To do so, follow these steps:

1. Under **Properties**, click *Metadata* to open the metadata property section.
2. A list of the key-value metadata pairs will be shown. Click the *circle* next to the key-value pair that you would like to edit or delete. The *Delete* and *Edit* options will turn blue.
3. To delete the metadata you selected, click the *Delete* button. Then press *Save*.
4. To edit the metadata you selected, click the *Edit* button.
5. Change the *key or value* for the metadata you chose. Then press *Save* to save the changes you made.

NOTE: If you change the Metadata properties, a new object will be created and will replace the old one or, turn it into the latest version of the object if versioning is enabled. The role responsible for the change will become the owner of the new object or the latest version of the object. To see details about the latest version of an object, click Latest Version next to the object's name above the Properties tab.

Additional Resources

If you followed the example found in this tutorial, you should have been able to create a browser-based upload using POST and AWS Signature Version 4 and allowed users to upload a text file with assigned metadata into a specific Amazon S3 bucket. If you would like to learn more information about Amazon Web Services and metadata, please refer to the mentioned links.

Object Key and Metadata:
https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMetadata.html#object-metadata

Adding Metadata to Amazon S3 Objects:

https://docs.aws.amazon.com/AmazonS3/latest/user-guide/add-object-metadata.html#add-object-metadata-user-defined

Viewing Object Properties:

https://docs.aws.amazon.com/AmazonS3/latest/user-guide/view-object-properties.html