This page provides an overview of signed URLs, which you use to give time-limited resource access to anyone in possession of the URL, regardless of whether they have a Google account. To learn how to create a signed URL, see V4 Signing Process with Cloud Storage Tools (/storage/docs/access-control/signing-urls-with-helpers) and V4 Signing Process with Your Own Program (/storage/docs/access-control/signing-urls-manually). To learn about other ways of controlling access to buckets and objects, see Overview of Access Control (/storage/docs/access-control/index).

ant: Signed URLs can only be used to access resources in Cloud Storage through XML API endpoints age/docs/request-endpoints).

A *signed URL* is a URL that provides limited permission and time to make a request. Signed URLs contain authentication information in their query string, allowing users without credentials to perform specific actions on a resource. When you generate a signed URL, you specify a user or service account which must have sufficient permission to make the request that the signed URL will make. After you generate a signed URL, anyone who possesses it can use the signed URL to perform specified actions, such as reading an object, within a specified period of time.

In some scenarios, you might not want to require your users to have a Google account in order to access Cloud Storage, but you still want to control access using your application-specific logic. The typical way to address this use case is to provide a signed URL to a user, which gives the user read, write, or delete access to that resource for a limited time. You specify an expiration time when you create the signed URL. Anyone who knows the URL can access the resource until the expiration time for the URL is reached or the key used to sign the URL is rotated.

Cloud Storage supports several methods for generating a signed URL:

- **V4 signing with service account authentication**[BETA]: This signing mechanism is described below.

- **V2 signing with service account authentication**: For more information about this signing mechanism, go here (/storage/docs/access-control/signed-urls-v2).
- **Signing with HMAC authentication**: If you're an Amazon Simple Storage Service (Amazon S3) user, you can use your existing workflows to generate signed URLs for Cloud Storage. Simply specify Cloud Storage resources, point to the host `storage.googleapis.com`, and use Google HMAC credentials in the process of generating the signed URL.

The following is an example of a signed URL that was created following the V4 signing process with service account authentication:

This signed URL provided access to read the object `cat.jpeg` in the bucket `example-bucket`. The query parameters that make this a signed URL are:

- `X-Goog-Algorithm`: The algorithm used to sign the URL.
- `X-Goog-Credential`: Information about the credentials used to create the signed URL.
- `X-Goog-Date`: The date and time the signed URL became usable, in the ISO 8601 (https://en.wikipedia.org/wiki/ISO_8601) basic format `YYYYMMDD'T'HHMMSS'Z'`.
- `X-Goog-Expires`: The length of time the signed URL remained valid, measured in seconds from the value in `X-Goog-Date`. In this example the Signed URL expires in 15 minutes. The longest expiration value is 604800 seconds (7days).
- `X-Goog-SignedHeaders`: Headers that had to be included as part of any request that used the signed URL.

- `X-Goog-Signature`: The authentication string that allowed requests using this signed URL to access `cat.jpeg`.

When using signed URLs with resumable uploads (/storage/docs/resumable-uploads) to upload objects to your bucket, you only need to use the signed URL in the initial `POST` request. No data is uploaded in the `POST` request; instead, the request returns a session URI which is used in subsequent `PUT` requests to upload data. Since the session URI is, in effect, an authentication token, the `PUT` requests do not need to use the original signed URL. This behavior allows the `POST` request to be made by the server, avoiding the need for clients to have to deal with signed URLs themselves.

ant: Be sure to transmit the session URI over HTTPS when giving it to a client.

Resumable uploads are pinned in the region they start in. For example, if you create a resumable upload URL in the US and give it to a client in Asia, the upload still goes through the US. Performing a resumable upload in a region where it wasn't initiated can cause slow uploads. To avoid this, have the initial `POST` request constructed and signed by the server, but then give the signed URL to the client so that the upload is initiated from their location. Once initiated, the client can use the resulting session URI normally to make `PUT` requests that do not need to be signed.

When working with signed URLs, keep in mind the following:

- Signed URLs can generally be made for any XML API request (/storage/docs/access-control/signing-urls-manually); however, the Node.js Cloud Storage Client Libraries currently can only make signed URLs for individual objects. For example, it cannot be used to make signed URLs for listing objects in a bucket.
- When specifying credentials, it is recommended that you identify your service account by using its email address; however, use of the service account ID is also supported.

Signed URLs use canonical requests (/storage/docs/authentication/canonical-requests) as part of the information encoded in their `X-Goog-Signature` query string parameter. When you make a signed URL with Cloud Storage tools (/storage/docs/access-control/signing-urls-with-helpers), the required canonical request is created and incorporated automatically. However, when you make a signed URL with your own program (/storage/docs/access-control/signing-urls-manually), you need to define the canonical request yourself.

The credential scope appears in both the string-to-sign and the `X-Goog-Credential` query string parameter. It has the following structure:

- `[DATE]`: Date formatted as YYYYMMDD, which must match the day used in the string-to-sign.
- `[LOCATION]`: The region where the resource resides or will be created. For Cloud Storage resources, the value of `[LOCATION]` is arbitrary: the `[LOCATION]` parameter exists to maintain compatibility with Amazon Simple Storage Service (Amazon S3).
- `storage`: The service name.
- `goog4_request`: The type of signed URL.

Example: 20181102/us/storage/goog4_request

When generating a signed URL using a program (/storage/docs/access-control/signing-urls-manually), one option for signing the string is to use tools provided by Google Cloud.

Signing within a App Engine application uses the App Engine Identity service, which utilizes App Engine service account credentials. For example, using the Python App Identity API (/appengine/docs/python/appidentity/), you can:

- Use `google.appengine.api.app_identity.sign_blob()` to sign the bytes from your constructed string, providing the `Signature` you need when assembling the signed URL.
- Use `google.appengine.api.app_identity.get_service_account_name()` to retrieve a service account name, which is the `GoogleAccessId` you need when assembling the signed URL.

App Engine also provides support in other languages:

- App Identity API for Java (/appengine/docs/java/appidentity/).
- App Identity API for PHP Overview (/appengine/docs/php/appidentity/).
- App Identity Go Functions (/appengine/docs/go/appidentity/).

The App Identity service rotates the private keys when it signs blobs. Signed URLs generated from the App Identity service are usable for at least one hour, but may stop working prior to the set expiration time. Given this, signed URLs generated from the App Identity service are best used for short-lived access to resources.

Signing can be accomplished using the IAM signBlob (/iam/credentials/reference/rest/v1/projects.serviceAccounts/signBlob) method.

- Create a signed URL with Cloud Storage tools (/storage/docs/access-control/signing-urls-with-helpers) such as gsutil.
- Create a signed URL with your own program (/storage/docs/access-control/signing-urls-manually).
- Learn more about Canonical requests (/storage/docs/authentication/canonical-requests).