Authenticate for using REST

This page describes how to authenticate when you make a REST request to a Google API.

For information about how to authenticate when you use Google client libraries, see Authenticate using client libraries (/docs/authentication/client-libraries).

Before you begin

To run the samples on this page, complete the following steps:

1. Install (/sdk/docs/install) the Google Cloud CLI.
2. To initialize (/sdk/docs/initializing) the gcloud CLI, run the following command:

   ```
   gcloud init
   ```

3. Enable the Cloud Resource Manager and Identity and Access Management (IAM) APIs:

   ```
   gcloud services enable cloudresourcemanager.googleapis.com
    iam.googleapis.com
   ```

If you don't want to use the gcloud CLI, you can skip these steps and use service account impersonation (#impersonated-sa) or the metadata server (#metadata-server) to generate a token.

Types of credentials

You can use the following types of credentials to authenticate a REST call:

- Your gcloud CLI credentials (#user-creds).

  This approach is the easiest and most secure way to provide credentials to a REST method in a local development environment. If your user account has the necessary

Identity and Access Management (IAM) permissions for the method you want to call, this is the preferred approach.

Your gcloud credentials are not the same as the credentials you provide to ADC using the gcloud CLI. For more information, see gcloud CLI authentication configuration and ADC configuration (/docs/authentication/gcloud#gcloud-credentials).

- The credentials provided to Application Default Credentials (ADC) (#rest-request).

  This method is the preferred option for authenticating a REST call in a production environment, because ADC finds credentials from the resource where your code is running (such as a Compute Engine virtual machine). You can also use ADC to authenticate in a local development environment. In this scenario, the gcloud CLI creates a file that contains your credentials in your local file system.

- The credentials provided by impersonating a service account (#impersonated-sa).

  This method requires more setup. If you want to use your existing credentials to obtain short-lived credentials for another service account, such as testing with a service account locally or requesting temporary elevated privileges, use this approach.

- The credentials returned by the metadata server (#metadata-server).

  This method works only in environments with access to a metadata server. The credentials returned by the metadata server are the same as the credentials that would be found by Application Default Credentials (/docs/authentication/application-default-credentials) using the attached service account, but you explicitly request the access token from the metadata server and then provide it with the REST request. Querying the metadata server for credentials requires an HTTP GET request; this method does not rely on the Google Cloud CLI.

gcloud CLI credentials

To run the example below, you need the `resourcemanager.projects.get` permission on the project. The `resourcemanager.projects.get` permission is included in a variety of roles—for example, the Browser role (/iam/docs/understanding-roles#project-roles) (`roles/browser`).

1. Use the `gcloud auth print-access-token` command (/sdk/gcloud/reference/auth/print-access-token) to insert an access token generated from your user credentials.

   The following example gets details for the specified project. You can use the same pattern for any REST request.

Before using any of the request data, make the following replacements:

- `PROJECT_ID`: Your Google Cloud project ID or name.

To send your request, choose one of these options:

curlPowerShell (#powershell)
(#curl)

Execute the following command:

```
curl -X GET \
     -H "Authorization: Bearer $(gcloud auth print-access-token)" \
     "https://cloudresourcemanager.googleapis.com/v3/projects/PROJE
```

The details for your project are returned.

For APIs that require a quota project, you must set one explicitly for the request. For more information, see Set the quota project with a REST request (#set-billing-project) on this page.

Application Default Credentials

To run the example below, the principal associated with the credentials you provide to ADC needs the **`resourcemanager.projects.get`** permission on the project. The **`resourcemanager.projects.get`** permission is included in a variety of roles—for example, the Browser role (/iam/docs/understanding-roles#project-roles) (**`roles/browser`**).

1. Provide credentials to ADC (/docs/authentication/provide-credentials-adc).

   If you are running on a Google Cloud compute resource, you should not provide your user credentials to ADC. Instead, use the attached service account to provide credentials. For more information, see Set up ADC for a resource with an attached service account (/docs/authentication/set-up-adc-attached-service-account).

2. Use the **`gcloud auth application-default print-access-token`** command (/sdk/gcloud/reference/auth/application-default/print-access-token) to insert the access token returned by ADC into your REST request.

   The following example gets details for the specified project. You can use the same pattern for any REST request.

   Before using any of the request data, make the following replacements:

- `PROJECT_ID`: Your Google Cloud project ID or name.

To send your request, choose one of these options:

curlPowerShell (#powershell)
(#curl)

Execute the following command:

```
curl -X GET \
     -H "Authorization: Bearer $(gcloud auth application-default pr
     "https://cloudresourcemanager.googleapis.com/v3/projects/PROJE
```

The details for your project are returned.

If your request returns an error message about end-user credentials not being supported by this API, see Set the quota project with a REST request (#set-billing-project) on this page.

Impersonated service account

The simplest way to impersonate a service account to generate an access token is by using the gcloud CLI. However, if you need to generate the token programmatically, or you don't want to use the gcloud CLI, you can use impersonation to generate a short-lived token.

For more information about impersonating a service account, see Use service account impersonation (/docs/authentication/use-service-account-impersonation).

1. Review the required permissions.
    - The principal you want to use to perform the impersonation must have the **`iam.serviceAccounts.getAccessToken`** permission on the impersonated service account (also called the *privilege-bearing service account*). The **`iam.serviceAccounts.getAccessToken`** permission is included in the Service Account Token Creator role (**`roles/iam.serviceAccountTokenCreator`**). If you are using your user account, you need to add this permission even if you have the Owner role (**`roles/owner`**) on the project. For more information, see Setting required permissions (/iam/docs/create-short-lived-credentials-direct#sa-credentials-permissions).

2. Identify or create the privilege-bearing service account—the service account you will impersonate.

   The privilege-bearing service account must have the permissions required to make the API method call.

gcloudShort-lived token ...
(#gcloud)

1. Use the `gcloud auth print-access-token` command (/sdk/gcloud/reference/auth/print-access-token) with the `--impersonate-service-account` flag (/sdk/gcloud/reference#--impersonate-service-account) to insert an access token for the privilege-bearing service account into your REST request.

The following example gets details for the specified project. You can use the same pattern for any REST request.

To run this example, the service account you impersonate needs the `resourcemanager.projects.get` permission. The `resourcemanager.projects.get` permission is included in a variety of roles—for example, the Browser role (/iam/docs/understanding-roles#project-roles) (`roles/browser`).

Make the following replacements:

- *`PRIV_SA`*: The email address of the privilege-bearing service account. For example, `my-sa@my-project.iam.gserviceaccount.com`.
- *`PROJECT_ID`*: Your Google Cloud project ID or name.

```
curl -X GET \
    -H "Authorization: Bearer $(gcloud auth print-access-token --imperso
    "https://cloudresourcemanager.googleapis.com/v3/projects/PROJECT_ID
```

Metadata server

To get an access token from the metadata server, you must make the REST call using one of the services that has access to a metadata server:

- Compute Engine (/compute/docs/instances/verifying-instance-identity#request_signature)

- App Engine standard environment (/appengine/docs/standard/python3/runtime#metadata_server)
- App Engine flexible environment (/appengine/docs/flexible/python/runtime#metadata_server)
- Cloud Run functions (/functions/docs/securing/function-identity#metadata-server)
- Cloud Run (/run/docs/container-contract#metadata-server)
- Google Kubernetes Engine (/kubernetes-engine/docs/concepts/workload-identity#instance_metadata)
- Cloud Build (/build/docs/securing-builds/authorize-service-to-service-access)

You use a command-line tool such as `curl` to get an access token, and then insert it into your REST request.

1. Query the metadata server for an access token:

   ```
   curl "http://metadata.google.internal/computeMetadata/v1/instance/service
       -H "Metadata-Flavor: Google"
   ```

   The request returns a response similar to the following example:

   ```
   {
         "access_token":"ya29.AHES6ZRN3-HlhAPya30GnW_bHSb_QtAi85nHq39HE3C2L
         "expires_in":3599,
         "token_type":"Bearer"
    }
   ```

2. Insert the access token into your REST request, making the following replacements:

   - *ACCESS_TOKEN*: The access token returned in the previous step.
   - *PROJECT_ID*: Your Google Cloud project ID or name.

   ```
   curl -X GET \
       -H "Authorization: Bearer ACCESS_TOKEN" \
   ```

```
        "https://cloudresourcemanager.googleapis.com/v3/projects/PROJECT_ID
```

Set the quota project with a REST request

To call some APIs with user credentials, you must also set the project that is billed for your usage and used to track quota. If your API call returns an error message saying that user credentials are not supported, or that the quota project is not set, you must explicitly set the quota project for the request. To set the quota project, include the `x-goog-user-project` header with your request.

For more information about when you might encounter this issue, see User credentials not working (/docs/authentication/troubleshoot-adc#user-creds-client-based).

You must have the `serviceusage.services.use` IAM permission for a project to be able to designate it as your billing project. The `serviceusage.services.use` permission is included in the Service Usage Consumer IAM role. If you don't have the `serviceusage.services.use` permission for any project, contact your security administrator or a project owner who can give you the Service Usage Consumer role in the project.

The following example uses the Cloud Translation API to translate the word "hello" into Spanish. The Cloud Translation API is an API that needs a quota project to be specified. To run the sample, create a file named `request.json` with the request body content.

Before using any of the request data, make the following replacements:

- ***PROJECT_ID***: The ID or name of the Google Cloud project to use as a billing project.

Request JSON body:

```
{
  "q": "hello",
  "source": "en",
  "target": "es"
}
```

To send your request, choose one of these options:

Save the request body in a file named **`request.json`**, and execute the following command:

```
curl -X POST \
     -H "Authorization: Bearer $(gcloud auth print-access-token)" \
     -H "x-goog-user-project: PROJECT_ID" \
     -H "Content-Type: application/json; charset=utf-8" \
     -d @request.json \
     "https://translation.googleapis.com/language/translate/v2"
```

The translation request succeeds. You can try the command without the **`x-goog-user-project`** header to see what happens when you do not specify the billing project.

What's next

- See an overview of authentication (/docs/authentication).
- Learn how to authenticate with client libraries (/docs/authentication/client-libraries).
- Understand gcloud CLI authentication configuration and ADC configuration (/docs/authentication/gcloud#gcloud-credentials).

Last updated 2024-12-20 UTC.