Google Workspace APIs

Authentication Best Practices

Google Cloud

Contents

About this document

Document details	
Purpose	This document clarifies the different authentication options available when using the Google Workspace APIs and presents the rationale and best practices for each.
Intended audience	A developer or collaboration engineer who needs to use a Google Workspace API to perform an automated task.
Key assumptions	That the audience has a basic understanding of Google Workspace APIs and a basic understanding of Google Cloud Platform service accounts.
Delivery note	This document captures best practices for authentication when using any Google Workspace API. It is applicable at any stage of the delivery cycle.

1. Two options

There are two options for using Google Workspace APIs as part of an automated process:

- Creating a dedicated Google Workspace user with 3-legged OAuth to authorize account usage
- Configuring a GCP service account to have Google Workspace domain-wide delegation of authority

Note that all Google Cloud Platform (GCP) service accounts have the ability to do "domain-wide delegation of authority" in the GCP console panel, but a Google Workspace super admin must grant API client access in the Google Workspace Admin console.

2. Google Workspace user with 3-legged OAuth

A Google Workspace customer can use 3-legged OAuth and authorize as a specific Google Workspace user. Following this process allows for the creation of a dedicated user for the specific application. However, using this process requires an interactive browser session for the initial authorization. For example, Google Cloud Directory Sync uses this method.

For applications that use Google API Client Libraries to interact with Google Workspace APIs, all Client Libraries support Application Default Credentials (example from Python Client Library). NOTE: these are different from the Google Cloud Platform client libraries to access GCP products with.

2.1 Obtaining a credential

If you are using OAuth 2.0 Client ID from GCP and the Google Cloud SDK, you can use the `gcloud auth application-default login` to obtain a credential that can be used by an automation process to authenticate as a Google Workspace user to make API requests.

To obtain a credential that can be used by an automation script:

1. Create an OAuth 2.0 client ID to represent the application that is calling the APIs.
2. Download the JSON representation of the client ID from **GCP Console > APIs and Services > Credentials > OAuth 2.0 client IDs**.
3. Wherever you have the Google Cloud SDK installed, run the following command with the correct *client-id-file* and *scopes* parameters:

```
gcloud auth application-default login \
   --client-id-file=client_id.json \
   --no-launch-browser \
   --scopes="\
https://www.googleapis.com/auth/admin.directory.group,\
https://www.googleapis.com/auth/admin.directory.user,\
https://www.googleapis.com/auth/cloud-platform"
```

4. Copy the login link into your browser and authenticate as the user you would like to make the API calls through.
5. Review the output for the location of the Application Default Credentials file, which usually appears following the text `Credentials saved to file`.
6. Copy the JSON file that contains the Application Default Credentials file and pass the path of this file to applications using the environment variable `GOOGLE_APPLICATION_CREDENTIALS=<path_to_credential_file>`

The credential will remain valid until it is removed from the Google Workspace user's account.

You can view (and revoke if desired) the credential at myaccount.google.com/permissions, listed under "Third-party apps with account access." The title of the app will be the GCP project ID that the OAuth 2.0 client ID belongs to.

2.2 Typical use cases

There are two typical use cases for this option:

- To authenticate various applications such as Cloud Directory Sync as individual Google Workspace users
- To perform an administrative task as part of an automation script that relies on a GCP service account for authentication (for example, to automate the creation of groups or user with an automation tool like terraform-provider-gsuite)

2.3 Typical customer concerns

The Google Workspace user account used for the automation might be suspended or the credential may be revoked, breaking the automation scripts.

2.4 Best practices

The following best practices help to mitigate the typical customer concerns:

- Create a regular user in Google Workspace that will act as a Google Workspace service account user specifically for the automation use case. Assign this Google Workspace service account user the minimum administrator roles to perform the task.
- Securely manage this special Google Workspace service account user:
    - Place the user in a Google Workspace organization unit with the minimum set of Google Workspace services enabled.
    - Store the user's password in a password manager and allow only authorised administrators access to the secret.
    - Configure user activity alerts to alert administrators of any changes to the user.
- Document the process of obtaining a new credential, ensuring that when the application loses access, there is appropriate logging and failure handling to inform the operator that the credentials have expired.
- Generate the credential with minimal scopes, aiming for least privilege scopes.
- Securely manage the credential file:
    - The credential JSON file is extremely sensitive, as it provides access to anyone to make API calls on behalf of the user.
    - Ensure that credential is handled securely and not shared or accessible to others.

3. GCP service account with domain-wide delegation

With this option, you grant permission to a GCP service account to authenticate as any user in the Google Workspace account. Anyone with access to the GCP service account key can then impersonate any user in the Google Workspace directory.

3.1 How it works

When the super admin adds the client ID of the service account to the Admin console, they need to explicitly set the API scopes they wish to grant access to.

Once a service account has been granted an API scope, it can make API requests using OAuth 2.0 by impersonating any user (including a super admin) in the Google Workspace account that:

- Has permissions for the API being called (via a pre built or custom role)
- For some Google Workspace APIs, such the Drive API, the user accessing the API has to have logged in at least once and accepted the Google Workspace Terms of Service

before using the API. There is no official listing of APIs that enforce this. Please be aware that this may be required for the API you are automating against.

Actions performed by an impersonated user appear in the admin audit log (Reports > Admin logs section of the Google Workspace UI). The UI and Reports API does not explicitly show that the action was done by an impersonated user. **The log entry appears as though the action were performed by the user as normal**.

3.2 Typical use cases

There are two typical use cases for this option:

- To perform user actions on behalf of multiple users or all users in a Google Workspace account.
    - Automatically removing sharing privileges from Drive files that are put to External
    - Moving data from an IT source (not available to common users) into a specific spreadsheet that isn't in a Team Drive
    - Automation related to Google's advertising products, where access is controlled at the level of user or profile
- To gather information about multiple users or all users in a Google Workspace account.
    - Detect and notify users who are approaching storage capacity on non-Google files in Drive.
    - Profile the types of files being stored in a user's Drive.
- The Google Workspace API you are automating against only supports domain-wide delegation, for example the Alert Center API only supports service accounts.

3.3 Typical customer concerns

1. Any user can be impersonated - no way to bind the Service Account to just one Google Workspace user (this is precisely what the feature is for, it is "Domain-Wide")
2. The admin audit log does not explicitly say that the Google Workspace user performed the action via impersonation from a GCP Service Account, but the Token Audit log shows the service account action

3.4 Best practices

The following best practices help to mitigate the typical customer concerns.

3.4.1 For all use cases

- Consider whether you actually need to use domain-wide delegation via service account for your use case.

    - You can assign specific admin roles to a service account. In case having admin privileges is sufficient, there is no need to use Domain-Wide Delegation.
    - If that is not enough but you can meet your goals without using domain-wide delegation, such as by using a "Google Workspace user with 3-legged OAuth," then that is a better option.

- When configuring the client ID in the Admin console, ensure that the minimum API scopes are configured. Review the OAuth 2.0 Scopes for Google APIs and assign the least privilege.
- Securely manage the GCP service account:
    - The GCP Service Account private key is extremely sensitive, as it enables anyone to impersonate any user within the API scopes configured. Note that access to Directory API scopes might mean that API callers could then escalate their own permissions (for example, create a user and assign a super admin role to it), so consider that when sharing the credentials and authorizing API scopes. Specifically, if read-only is what is needed, use that. Follow the principle of least privilege.
    - Review IAM permissions on the GCP project that the service account resides in, ensuring that only the appropriate employees have the ability to use and create private keys for the Service Account. Again, this avenue could be exploited for privilege escalation.
    - Once a key is downloaded for usage, ensure that it is handled securely and not shared or accessible to others
- Secure the usage of the GCP service account key to authenticate with the APIs during automation:
    - Log each usage so that it can be reconciled with the Google Workspace Admin logs to ensure each action in the Google Workspace Admin logs was as expected. This will make it easier to reconcile intended actions taken by the service account with unintended uses in the event that the service account key is leaked. This is a logging measure and not a mitigation technique.

3.4.2 For use cases where an API requires one or more users to perform an action

For example, a script needs to be run, and it relies on a number of identities to complete the automation task. The right approach is to use a GCP service account with delegation, and use the various APIs' `subject` argument to impersonate the identities required. These identities should be created for impersonation purposes only and should not be human users. This makes it easier to review the logs and determine "who did what when".

- Create one or more Google Workspace service account users specifically for impersonation by the GCP service account. Assign these Google Workspace service account users the minimum administrator roles to perform the task.

- Securely manage this special Google Workspace Service Account user:
    - Place the user in a Google Workspace organization unit with the minimum set of Google Workspace services enabled.
    - Assign a random password for the user, to prevent people from directly logging in as that user.
    - Configure user activity alerts to alert administrators of changes to the user.

- Secure the usage of the GCP service account key to authenticate with the APIs during automation:
    - Where the GCP service account is used to make API calls, ensure that the automation code is using only the expected Google Workspace service account users. Consider enforcing this as a check during code review or parametrizing the value and passing the value in at runtime from a configuration provider that provides ACLs (for example, pass the service account private key and the Google Workspace service account user email as encrypted parameters in a build pipeline).