This page explains how to create short-lived credentials for service accounts to impersonate their identities.

Service accounts can use short-lived credentials to authenticate calls to Google Cloud APIs and other non-Google APIs. Short-lived credentials have a limited lifetime, with durations of just a few hours or shorter. Short-lived service account credentials are useful for scenarios where you need to grant limited access to resources for trusted service accounts, and they reduce the risk compare to using long-lived credentials like service account keys.

Supported credential types include OAuth 2.0 access tokens, OpenID Connect ID tokens, self-signed JSON Web Tokens (JWTs), and self-signed binary objects (blobs). The most commonly used credential types are OAuth 2.0 access tokens (#sa-credentials-oauth) and OpenID Connect (OIDC) ID tokens (#sa-credentials-oidc). Some example scenarios include the following:

- **OAuth 2.0 access token**: An OAuth 2.0 access token is useful for authenticating the identity of a service account to Google Cloud APIs. Consider the following example use case: to get elevated permissions on a project, a service administrator can impersonate a service account to call Google Cloud APIs by creating an OAuth 2.0 access token belonging to that service account. The token has a short lifetime so that the elevated permissions are temporary. This is useful especially when there is an emergency in a production environment, and a service administrator needs a short-term elevated authorization for debugging.
- **OIDC ID token**: An OIDC ID token is useful for authenticating the identity of a service account to services that accept OpenID Connect (https://openid.net/connect/). Consider the following example use case: by creating an OIDC ID token belonging to a service account, a service running on Google Cloud can authenticate itself to another service deployed on a third-party cloud provider, such as a data pipeline job. If the target service is configured with OIDC, the authentication will succeed.

- Understand Cloud IAM service accounts (/iam/docs/service-accounts)
- If you haven't already, enable billing and the Cloud IAM API by following the steps in the Quickstart (/iam/docs/quickstart#before-you-begin).

To get started, create a new service account (/iam/docs/creating-managing-service-accounts#creating).

When creating a service account, ensure that it has been granted the fewest permissions necessary to accomplish th
ed task. See Granting roles to service account (/iam/docs/granting-roles-to-service-accounts) for more information.

There are two different flows that allow a caller to create short-lived credentials for a service account. Each flow requires the appropriate permissions:

- **Direct request**: The caller is authenticated as either a Google account or a service account and makes a direct request to create short-lived credentials. Two identities are involved in this flow: the caller, and the service account for whom the credential is created.
- **Delegated request**: Like the direct request flow, the caller is authenticated as either a Google account or a service account, but instead the request is delegated to one or more service accounts in a *delegation chain*. In this flow, multiple service accounts act as intermediaries between the original caller and the service account for whom the credential is created. Each service account in the delegation chain must have the required permissions to pass along the request.

  This flow is useful for scenarios where a project contains tiers of limited-privilege service accounts, each of which has been configured to perform a specific or limited function on certain resources. For example, one service account is only granted permissions for Cloud Storage resources, another is only granted permissions for Compute Engine resources, and so on. To successfully delegate a request across service accounts, each must be listed in the delegation chain.

If the service account being granted permissions to create short-lived credentials is the same service account making
st, you may introduce a security vulnerability for your Google Cloud project. In other words, we discourage making the
st where the caller and the requested service account are the same identity.

As described above (#sa-credentials-permissions), a direct request involves only two identities: the caller, and the service account for whom the credential is created. In this flow, consider the following identities:

- Service Account 1 (*SA-1*), the caller who issues a request for the short-lived credentials.
- Service Account 2 (*SA-2*), the limited-privilege account for whom the credential is created.

To grant *SA-1* permissions to create short-lived credentials, grant it the Service Account Token Creator role (`roles/iam.serviceAccountTokenCreator`) on *SA-2*. This is an example of a service account being treated like a resource as opposed to an identity: *SA-1* must be granted the permission to issue credentials for *SA-2*. For more information about treating service accounts as an identity *or* a resource, see Service account permissions (/iam/docs/service-accounts#service_account_permissions).

The following steps use the REST API to grant the required permissions, however you can also use the Cloud Console or the `gcloud` command-line tool (/sdk/gcloud/).

As described above (#sa-credentials-permissions), a delegated request involves more than two identities: the caller, one or more service accounts in a *delegation chain*, and finally the service account. In this flow, consider the following identities:

- Service Account 1 (*SA-1*), the caller who issues a request for the short-lived credentials.
- Service Account 2 (*SA-2*), an intermediary service account that will delegate the initial request to *SA-3*.
- Service Account 3 (*SA-3*), the limited-privilege account for whom the credential is created.

To allow delegation, each account must grant the Service Account Token Creator role (`roles/iam.serviceAccountTokenCreator`) to the previous account in the chain.

In this particular example, *SA-1* must be granted the Service Account Token Creator role (`roles/iam.serviceAccountTokenCreator`) on *SA-2*. This is an example of a service account being treated like a resource as opposed to an identity: *SA-1* must be granted the permission to delegate access for *SA-2*. For more information about treating service accounts as an identity *or* a resource, see Service account permissions (/iam/docs/service-accounts#service_account_permissions).

In this example flow, there is only one intermediary service account. To delegate access through more than one service account, you must also assign this role to any other service account in the chain.

Next, *SA-2* must also be granted the Service Account Token Creator role (`roles/iam.serviceAccountTokenCreator`) on *SA-3*. This allows *SA-2* to create short-lived credentials for *SA-3*.

The following steps use the REST API to grant the required permissions, however you can also use the Cloud Console or the `gcloud` command-line tool (/sdk/gcloud/).

After you have granted the appropriate permissions for each identity, you can request short-lived credentials belonging to the desired service account. The following credential types are supported:

- OAuth 2.0 access tokens (#sa-credentials-oauth)
- OpenID Connect ID tokens (#sa-credentials-oidc)
- Self-signed JSON Web Tokens (JWTs) (#sa-credentials-jwt)
- Self-signed binary objects (blobs) (#sa-credentials-blob)

To understand how to specify a delegation chain for these requests, see the Specifying a delegation chain (#sa-credentials-delegated-chain) section below.

OAuth 2.0 access tokens are valid for 1 hour (3,600 seconds). To generate an OAuth 2.0 access token for a service account:

OpenID Connect ID tokens are valid for 1 hour (3,600 seconds). To generate an ID token for a service account:

Self-signed JWTs are useful in a variety of scenarios, such as:

- Authenticating a call to a Google API as described in Google's Authentication Guide (https://developers.google.com/identity/protocols/OAuth2ServiceAccount#jwt-auth).
- Securely communicating between Google Cloud or non-Google services, such as App Engine applications. In this scenario, one application can sign a token that can be verified by another application for authentication purposes.
- Treating a service account as an identity provider by signing a JWT that contains arbitrary claims about a user, account, or device.

To generate a self-signed JSON Web Token (JWT) for a service account:

Self-signed blobs are useful in scenarios when you need to securely transmit arbitrary binary data, usually for authentication purposes. For example, if you want to use a custom protocol/token type (not JWT), you can include that data in a signed blob for use by a downstream service.

To generate a self-signed blob for a service account:

When using a delegated request flow (#sa-credentials-delegated) to create short-lived service account credentials, the request body for each API must specify the service account delegation chain in the correct order and in the following format:

`projects/-/serviceAccounts/`***`ACCOUNT-EMAIL-OR-UNIQUEID`***

For example, in a delegation chain that flows from *SA-1* (caller) to *SA-2* (delegated) to *SA-3* (delegated) to *SA-4*, the **`delegates[]`** field would contain *SA-2* and *SA-3* in the following order:

The caller and the service account for whom the credential is created are not included in the delegation chain.