# Ephemeral tokens

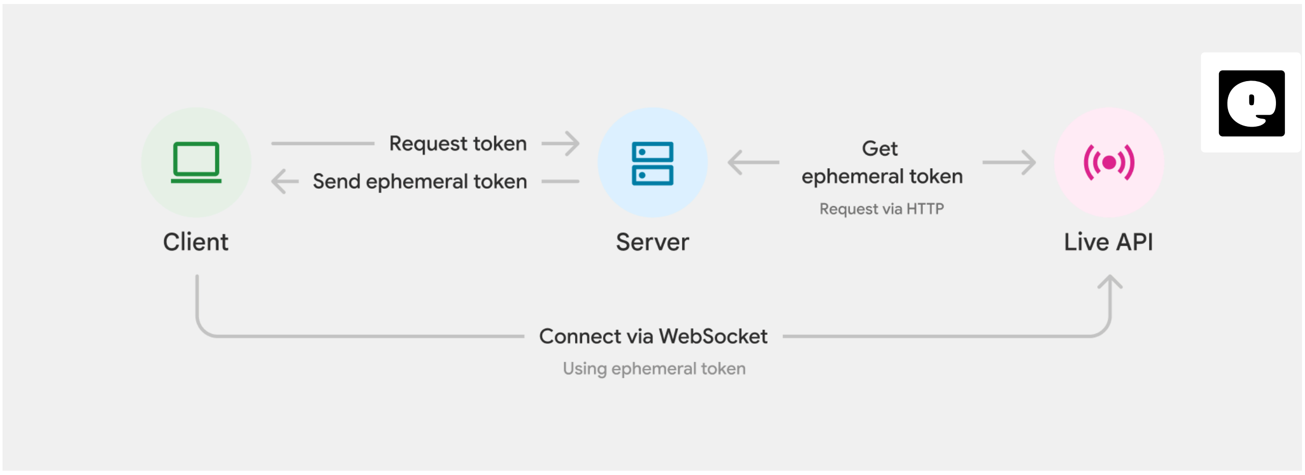
Ephemeral tokens are short-lived authentication tokens for accessing the Gemini API through WebSockets (https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API). They are designed to enhance security when you are connecting directly from a user's device to the API (a client-to-server (/gemini-api/docs/live#implementation-approach) implementation). Like standard API keys, ephemeral tokens can be extracted from client-side applications such as web browsers or mobile apps. But because ephemeral tokens expire quickly and can be restricted, they significantly reduce the security risks in a production environment. You should use them when accessing the Live API directly from client-side applications to enhance API key security.

**Note:** At this time, ephemeral tokens are only compatible with Live API (/gemini-api/docs/live).

## How ephemeral tokens work

Here's how ephemeral tokens work at a high level:

1. Your client (e.g. web app) authenticates with your backend.
2. Your backend requests an ephemeral token from Gemini API's provisioning service.
3. Gemini API issues a short-lived token.
4. Your backend sends the token to the client for WebSocket connections to Live API. You can do this by swapping your API key with an ephemeral token.
5. The client then uses the token as if it were an API key.



This enhances security because even if extracted, the token is short-lived, unlike a long-lived API key deployed client-side. Since the client sends data directly to Gemini, this also improves latency and avoids your backends needing to proxy the real time data.

## Create an ephemeral token

Here is a simplified example of how to get an ephemeral token from Gemini. By default, you'll have 1 minute to start new Live API sessions using the token from this request (**`newSessionExpireTime`**), and 30 minutes to send messages over that connection (**`expireTime`**).

Python (#python) JavaScript (#javascript)

```
import { GoogleGenAI } from "@google/genai";

const client = new GoogleGenAI({});
const expireTime = new Date(Date.now() + 30 * 60 * 1000).toISOString();

  const token: AuthToken = await client.authTokens.create({
    config: {
      uses: 1, // The default
      expireTime: expireTime // Default is 30 mins
      newSessionExpireTime: new Date(Date.now() + (1 * 60 * 1000)), // D
      httpOptions: {apiVersion: 'v1alpha'},
    },
  });
```

For **`expireTime`** value constraints, defaults, and other field specs, see the API reference (https://ai.google.dev/api/live#ephemeral-auth-tokens). Within the **`expireTime`** timeframe, you'll need **`sessionResumption`** (/gemini-api/docs/live-session#session-resumption) to reconnect the call every 10 minutes (this can be done with the same token even if **`uses: 1`**).

It's also possible to lock an ephemeral token to a set of configurations. This might be useful to further improve security of your application and keep your system instructions on the server side.

Python (#python) JavaScript (#javascript)

```
import { GoogleGenAI } from "@google/genai";

const client = new GoogleGenAI({});
const expireTime = new Date(Date.now() + 30 * 60 * 1000).toISOString();

const token = await client.authTokens.create({
    config: {
        uses: 1, // The default
        expireTime: expireTime,
        liveConnectConstraints: {
            model: 'gemini-2.5-flash-native-audio-preview-12-2025',
            config: {
                sessionResumption: {},
                temperature: 0.7,
                responseModalities: ['AUDIO']
            }
        },
        httpOptions: {
            apiVersion: 'v1alpha'
        }
    }
});

// You'll need to pass the value under token.name back to your client tc
```

You can also lock a subset of fields, see the SDK documentation (https://googleapis.github.io/python-genai/genai.html#genai.types.CreateAuthTokenConfig.lock_additional_fields) for more info.

## Connect to Live API with an ephemeral token

Once you have an ephemeral token, you use it as if it were an API key (but remember, it only works for the live API, and only with the `v1alpha` version of the API).

The use of ephemeral tokens only adds value when deploying applications that follow client-to-server implementation (/gemini-api/docs/live#implementation-approach) approach.

JavaScript (#javascript)

```
import { GoogleGenAI, Modality } from '@google/genai';

// Use the token generated in the "Create an ephemeral token" section he
const ai = new GoogleGenAI({
  apiKey: token.name
});
const model = 'gemini-2.5-flash-native-audio-preview-12-2025';
const config = { responseModalities: [Modality.AUDIO] };

async function main() {

  const session = await ai.live.connect({
    model: model,
    config: config,
    callbacks: { ... },
  });

  // Send content...

  session.close();
}

main();
```

**Note:** If not using the SDK, note that ephemeral tokens must either be passed in an **`access_token`** query parameter, or in an HTTP **`Authorization`** prefixed by the auth-scheme (https://datatracker.ietf.org/doc/html/rfc7235#section-2.1) **`Token`**.

See Get started with Live API (/gemini-api/docs/live) for more examples.

## Best practices

- Set a short expiration duration using the **`expire_time`** parameter.
- Tokens expire, requiring re-initiation of the provisioning process.
- Verify secure authentication for your own backend. Ephemeral tokens will only be as secure as your backend authentication method.
- Generally, avoid using ephemeral tokens for backend-to-Gemini connections, as this path is typically considered secure.

## Limitations

Ephemeral tokens are only compatible with Live API (/gemini-api/docs/live) at this time.

## What's next

- Read the Live API reference (https://ai.google.dev/api/live#ephemeral-auth-tokens) on ephemeral tokens for more information.

Last updated 2025-12-18 UTC.