


Plugin authentication

Plugins offer numerous authentication schemas to accommodate various use cases. To specify the authentication schema for your plugin, use the manifest file. Our [plugin domain policy](#) outlines our strategy for addressing domain security issues. For examples of available authentication options, refer to the [examples section](#), which showcases all the different choices.

The `ai-plugin.json` file requires an `auth` schema to be set. Even if you elect to use no authentication, it is still required to specify `"auth": { "type": "none" }`.

We support only localhost development without authentication; if you want to use service, user, or OAuth authentication, you need to set up a remote server.

Service level

 We suggest service level auth as it gives developers control over how their plugin is being used but also doesn't introduce overhead for users.

If you want to specifically enable OpenAI plugins to work with your API, you can provide a client secret during the plugin installation flow. This means that all traffic from OpenAI plugins will be authenticated but not on a user level. This flow benefits from a simple end user experience but less control from an API perspective.

- To start, select "Develop your own plugin" in the ChatGPT plugin store, and enter the domain where your plugin is hosted.
- In `ai-plugin.json`, set `auth.type` to `"service_http"` as is shown in our [service level auth example](#).
- You will be prompted for your service access token, which is a string specified in your code.
 - We securely store an encrypted copy of your service access token to enable plugin installation without additional authentication.

- The service access token is sent in the `Authorization` header for plugin requests.
- Once you add your service access token into the ChatGPT UI, you will be presented with a verification token.
- Add the verification token to your `ai-plugin.json` file under the auth section as shown below.

```
1 "auth": {  
2   "type": "service_http",  
3   "authorization_type": "bearer",  
4   "verification_tokens": {  
5     "openai": "Replace_this_string_with_the_verification_token_ge  
6   }  
7 },
```

The verification tokens are designed to support multiple applications. You can simply add the additional applications you want your plugin to support:

```
1 "verification_tokens": {  
2   "openai": "Replace_this_string_with_the_verification_token_ge  
3   "other_service": "abc123"  
4 }
```

OAuth

The plugin protocol is compatible with OAuth. A simple example of the OAuth flow we are expecting should look something like the following:

- To start, select "Develop your own plugin" in the ChatGPT plugin store, and enter the domain where your plugin is hosted (cannot be localhost).
- In `ai-plugin.json`, set `auth.type` to `"oauth"` as is shown in our [OAuth example](#).

- Then, you will be prompted to enter the OAuth client ID and client secret.
 - The client ID and secret can be simple text strings but should [follow OAuth best practices](#).
 - We store an encrypted version of the client secret, while the client ID is available to end users.
- Once you add your client ID and client secret into the ChatGPT UI, you will be presented with a verification token.
- Add the verification token to your `ai-plugin.json` file under the auth section as shown below.
- OAuth requests will include the following information: `request={ 'grant_type' : 'authorization_code', 'client_id': 'id_set_by_developer', 'client_secret': 'secret_set_by_developer', 'code': 'abc123', 'redirect_uri': 'https://chat.openai.com/aip/plugin-some_plugin_id/oauth/callback' }`
- In order for someone to use a plugin with OAuth, they will need to install the plugin and then be presented with a "Sign in with" button in the ChatGPT UI.
- The `authorization_url` endpoint should return a response that looks like: `{ "access_token": "example_token", "token_type": "bearer", "refresh_token": "example_token", "expires_in": 59 }`
- During the user sign in process, ChatGPT makes a request to your `authorization_url` using the specified `authorization_content_type`, we expect to get back an access token and optionally a [refresh token](#) which we use to periodically fetch a new access token.
- Each time a user makes a request to the plugin, the user's token will be passed in the Authorization header: ("Authorization": "[Bearer/Basic][user's token]").

i We require that OAuth applications make use of the [state parameter](#) for security reasons.

Below is an example of what the OAuth configuration inside of the `ai-plugin.json` file might look like:

```


1  "auth": {
2    "type": "oauth",
3    "client_url": "https://example.com/authorize",
4    "scope": "",
5    "authorization_url": "https://example.com/auth/",
6    "authorization_content_type": "application/json",
7    "verification_tokens": {
8      "openai": "Replace_this_string_with_the_verification_token_g
9    }
10 },

```

To better understand the URL structure for OAuth, here is a short description of the fields:

- When you set up your plugin with ChatGPT, you will be asked to provide your OAuth `client_id` and `client_secret`.
- When a user logs into the plugin, ChatGPT will direct the user's browser to `"[client_url]?response_type=code&client_id=[client_id]&scope=[scope]&state=xyz123&redirect_uri=https%3A%2F%2Fchat.openai.com%2Faip%2F[p`
- The `plugin_id` is passed via the request made to your OAuth endpoint (note that it is not visible in the ChatGPT UI today but may be in the future). You can inspect the request there to see the `plugin_id`. We expect the `state` to be passed along when you redirect back to `redirect_uri`. If the `state` doesn't match the initial `state`, or has expired, the authentication flow will fail.
- After your plugin redirects back to the given `redirect_uri`, ChatGPT will complete the OAuth flow by making a POST request to the `authorization_url` with content type `authorization_content_type` and parameters `{ "grant_type": "authorization_code", "client_id": [client_id], "client_secret": [client_secret], "code": [the code that was returned with the redirect], "redirect_uri": [the same redirect uri as before] }`.

No authentication


 We do not recommend the use of "no authentication", consider using "service authentication".

We support no-auth flow for applications that do not require authentication, where a user is able to send requests directly to your API without any restrictions. This is particularly useful if you have an open API that you want to make available to everyone, as it allows traffic from sources other than just OpenAI plugin requests.

```
1  "auth": {  
2    "type": "none"  
3  },
```



User level

 Due to current UI limitations, we are not allowing plugins with "user authentication" into the plugin store. We expect this may change in the future.

Just like how a user might already be using your API, we allow user level authentication through enabling end users to copy and paste their secret API key into the ChatGPT UI during plugin install. While we encrypt the secret key when we store it in our database, we do not recommend this approach given the poor user experience.

- To start, a user pastes in their access token when installing the plugin
- We store an encrypted version of the token
- We then pass it in the Authorization header when making requests to the plugin ("Authorization": "[Bearer/Basic][user's token]")

```
1  "auth": {
```



```
2   "type": "user_http",  
3   "authorization_type": "bearer",  
4 },
```

Was this page useful? 