Google's OAuth 2.0 APIs

Setting up OAuth 2.0

- Before your application can use Google's OAuth 2.0 authentication system for user login, you must set up a project in the Google API Console to obtain OAuth 2.0 credentials, set a redirect URI, and (optionally) customize the branding information that your users see on the user-consent screen. You can also use the API Console to create a service account, enable billing, set up filtering, and do other tasks. For more details, see the Google API Console Help.

Obtain OAuth 2.0 credentials

- You need OAuth 2.0 credentials, including a client ID and client secret, to authenticate users and gain access to Google's APIs.
- To view the client ID and client secret for a given OAuth 2.0 credential, click the following text: Select credential. In the window that opens, choose your project and the credential you want, then click **View**.
- Or, view your client ID and client secret from the **Credentials page** in API Console:
- Go to the Credentials page.
- Click the name of your credential or the pencil (create) icon. Your client ID and secret are at the top of the page.

Set a redirect URI

- The redirect URI that you set in the API Console determines where Google sends responses to your authentication requests.
- To create, view, or edit the redirect URIs for a given OAuth 2.0 credential, do the following:
- Go to the Credentials page.
- In the **OAuth 2.0 client IDs** section of the page, click a credential.
- View or edit the redirect URIs.
- If there is no **OAuth 2.0 client IDs** section on the Credentials page, then your project has no OAuth credentials. To create one, click **Create credentials**.

Credentials Page

Google APIs adbi

Search for APIs and Services

API APIs & Services

- Dashboard
- Library
- Credentials
- OAuth consent screen
- Domain verification
- Page usage agreements

Credentials

+ CREATE CREDENTIALS DELETE

Create credentials to access your enabled APIs. Learn more

API Keys

Name	Creation date	Restrictions	Key	Usage with all services (last 30 days)
No API keys to display				

OAuth 2.0 Client IDs

Name	Creation date	Type	Client ID
Web client 1	Jun 10, 2020	Web application	732879764903-sf4t...

Service Accounts

Manage service accounts

Email	Name	Usage with all services (last 30 days)
No service accounts to display		

Google APIs adbi

Search for APIs and Services

API APIs & Services

- Dashboard
- Library
- Credentials
- OAuth consent screen
- Domain verification
- Page usage agreements

Client ID for Web application

DOWNLOAD JSON RESET SECRET DELETE

Name *

Web client 1

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.

Client ID	732879764903-sf4tkpm4os716o51hla8t32shskji026.apps.googleusercontent.com
Client secret	MimDLU5EmlqrsLAz6PVagRb3
Creation date	June 10, 2020 at 7:19:05 PM GMT-4

Credentials to authenticate users and gain access to Google's APIs

Authorized JavaScript origins

For use with requests from a browser

+ ADD URI

Authorized redirect URIs

For use with requests from a web server

URIs

https://oauth.pstmn.io/v1/callback

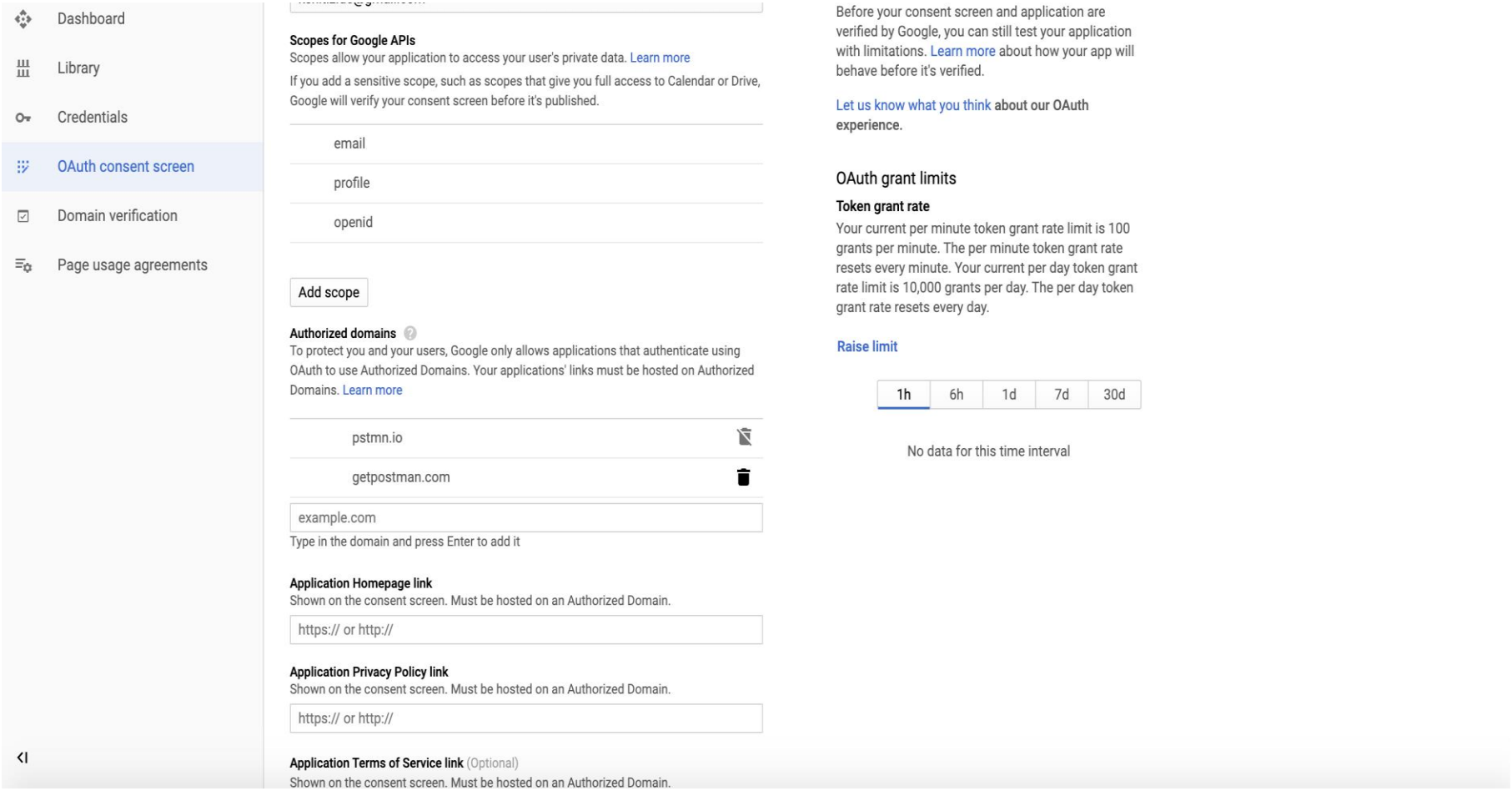
The redirect URI that you set in the API Console determines where Google sends responses to your authentication requests.

+ ADD URI

SAVE CANCEL

Customize the user consent screen

- For your users, the OAuth 2.0 authentication experience includes a consent screen that describes the information that the user is releasing and the terms that apply. For example, when the user logs in, they might be asked to give your app access to their email address and basic account information. You request access to this information using the scope parameter, which your app includes in its authentication request. You can also use scopes to request access to other Google APIs.
- The user consent screen also presents branding information such as your product name, logo, and a homepage URL. You control the branding information in the API Console.
- To enable your project's consent screen:
- Open the Consent Screen page in the Google API Console.
- If prompted, select a project, or create a new one.
- Fill out the form and click **Save**.
- The following consent dialog shows what a user would see when a combination of OAuth 2.0 and Google Drive scopes are present in the request. (This generic dialog was generated using the Google OAuth 2.0 Playground, so it does not include branding information that would be set in the API Console.)

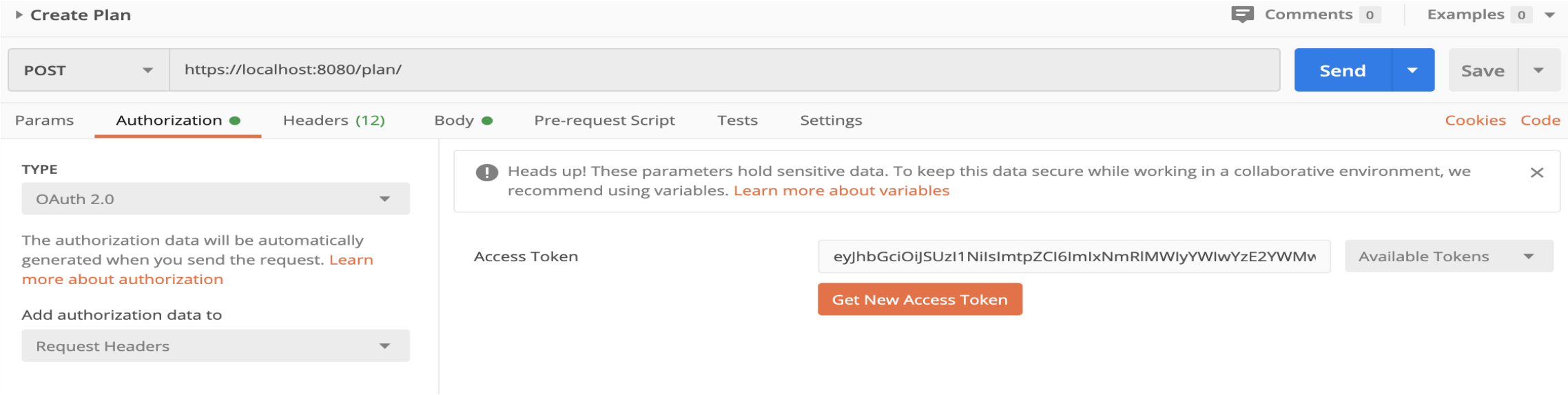


Set the authorized domains to postman.io or getpostman.com as shown in the fig

Send an authentication request to Google

- The next step is forming an HTTPS GET request with the appropriate URI parameters. Note the use of HTTPS rather than HTTP in all the steps of this process; HTTP connections are refused. You should retrieve the base URI from the Discovery document using the authorization_endpoint metadata value. The following discussion assumes the base URI is https://accounts.google.com/o/oauth2/v2/auth.
- For a basic request, specify the following parameters:
- **client_id**, which you obtain from the API Console Credentials page .
- **response_type**, which in a basic authorization code flow request should be code. (Read more at response_type.)
- **scope**, which in a basic request should be openid email. (Read more at scope.)
- **redirect_uri** should be the HTTP endpoint on your server that will receive the response from Google. The value must exactly match one of the authorized redirect URIs for the OAuth 2.0 client, which you configured in the API Console Credentials page. If this value doesn't match an authorized URI, the request will fail with a redirect_uri_mismatch error.
- **state** should include the value of the anti-forgery unique session token, as well as any other information needed to recover the context when the user returns to your application, e.g., the starting URL. (Read more at state.)
- **nonce** is a random value generated by your app that enables replay protection when present.
- **login_hint** can be the user's email address or the sub string, which is equivalent to the user's Google ID. If you do not provide a login_hint and the user is currently logged in, the consent screen includes a request for approval to release the user's email address to your app. (Read more at login_hint.)
- Use the **hd** parameter to optimize the OpenID Connect flow for users of a particular G Suite domain. (Read more at hd.)

Postman Configuration :



- Select the type as Oauth2.0
- Set the Add authorization data to Request Headers as shown in the fig.
- Then click on Get New Access Token Button to create and send the authentication request to google

Authentication Request to Google:

GET NEW ACCESS TOKEN

Token Name	adbi_google_idp
Grant Type	Authorization Code
Callback URL	https://oauth.pstmn.io/v1/callback
	☑ Authorize using browser
Auth URL	https://accounts.google.com/o/oauth2/v2/auth
Access Token URL	https://accounts.google.com/o/oauth2/token
Client ID	732879764903-sf4tkpm4os716o51hla8t32shskji026.apps.googleusercor
Client Secret	MimDLU5EmlqrsLAz6PVagRb3
Scope	openid
State	State
Client Authentication	Send client credentials in body

Cancel | Request Token

A successful response to this request contains the following fields in a JSON array:

Fields	
`access_token`	A token that can be sent to a Google API.
`expires_in`	The remaining lifetime of the access token in seconds.
`id_token`	A JWT that contains identity information about the user that is digitally signed by Google.
`scope`	The scopes of access granted by the `access_token` expressed as a list of space-delimited, case-sensitive strings.
`token_type`	Identifies the type of token returned. At this time, this field always has the value `Bearer`.
`refresh_token`	(optional) This field is only present if the `access_type` parameter was set to `offline` in the authentication request. For details, see Refresh tokens.

This is how Google's response in postman looks like:

MANAGE ACCESS TOKENS

All Tokens Delete

~~adbi_google_idp~~

adbi_google_idp

Token Name	adbi_google_idp
Access Token	ya29.a0AfH6SMAH2u4IZIgZsve41H2u_tbPPx1qk6T_ni9FKNwP6XHPp0e7RnNcc9rIxThQY99Yz3zbqGZ0agdzOdsqahAhhpc-_Ey71lC9pSGLqI0UlS1nuV3JjsDyUlJa_ClXj0FwjTZbXq7NG4mXXWZruJnUNp9ENp2D1UXn
Token Type	Bearer
expires_in	3597
scope	openid
id_token	eyJhbGciOiJSUzI1NiIsImtpZCI6ImIxNmRlMWIyYWIwYzE2YWMwYWNmNjYyZWYwMWY3NTY3ZTU0NDI1MmEiLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJhY2NvdW50cy5nb29nbGUuY29tIiwiYXpwIjoiNzMyODc5NzY0OTAzLXNmNHRrcG00b3M3MTZvNTFobGE4dDMyc2hza2ppMDI2LmFwcHMuZ29vZ2xldXNlcmNvbnRlbnQuY29tIiwiYXVkIjoiNzMyODc5NzY0OTAzLXNmNHRrcG00b3M3MTZvNTFobGE4dDMyc2hza2ppMDI2LmFwcHMuZ29vZ2xldXNlcmNvbnRlbnQuY29tIiwic3ViIjoiMTE2MzU1NDgyNTQ0ODU3NzkzODE1IiwiYXRfaGFzaCI6IjUyRjdaWnFIVEthY1hjc0

Reference Links:

Goolge’ step by step guide to setup Oauth 2.0:

https://developers.google.com/api-client-library/java/google-api-java-client/setup

Authenticating token with a backend server:

https://developers.google.com/identity/sign-in/web/backend-auth