Last updated July 26, 2017. | Unknown author

Loading the client library

This page documents all the methods and classes defined in the JavaScript client library.

gapi.load(*libraries, callbackOrConfig*)

Asynchronously loads the gapi libraries requested. Use this method to load the gapi.client library.

Arguments:

Name	Type	Description
libraries	string	A colon (:) separated list of gapi libraries. Ex: "client:auth2".
callbackOrConfig	function|object	Either: • A callback function that is called when the libraries have finished loading. • An object encapsulating the various configuration parameters for this method. Only callback is required. **Name** / **Type** / **Description** *callback* / function / The function called when the libraries have finished loading. *onerror* / function / The function called if the libraries failed to load. *timeout* / number / The number of milliseconds to wait before calling the *ontimeout* function, if the libraries still haven't loaded. *ontimeout* / function / The function called if the libraries loading has taken more time than specified by the *timeout* parameter.

Example:

```
gapi.load('client', {
  callback: function() {
    // Handle gapi.client initialization.
    initGapiClient();
  },
  onerror: function() {
    // Handle loading error.
    alert('gapi.client failed to load!');
  },
  timeout: 5000, // 5 seconds.
  ontimeout: function() {
    // Handle timeout.
    alert('gapi.client could not load in a timely manner!');
  }
});
```

Client setup

gapi.client.init(*args*)

Initializes the JavaScript client with API key, OAuth client ID, scope, and API discovery document(s). If OAuth client ID and scope are provided, this function will load the gapi.auth2 module to perform OAuth. The gapi.client.init function can be run multiple times, such as to set up more APIs, to change API key, or initialize OAuth lazily. Note that the scope and clientId parameters cannot be provided multiple times, since the gapi.auth2 module can only be initialized once.

Arguments:

Name	Type	Description
args	object	An object encapsulating the various arguments for this method. Every argument is optional.

Name	Type	Description
apiKey	string	The API Key to use.
discoveryDocs	array	An array of discovery doc URLs or discovery doc JSON objects (Example).
clientId	string	The app's client ID, found and created in the Google Developers Console.
scope	string	The scopes to request, as a space-delimited string.

Returns:

Type	Description
goog. Thenable	The return value is a **Promise**-like goog.Thenable object that resolves when all initializations, including setting the API key, loading discovery documents, and initializing auth, are done.

gapi.client.load(*urlOrObject*)

Loads the client library interface to a particular API with discovery document URL or JSON object. Returns a **Promise**-like goog.Thenable object that resolves when the API interface is loaded. The loaded API interface will be in the form gapi.client.*api.collection.method*. For example, the Moderator API would create methods like gapi.client.moderator.series.list.

Arguments:

Name	Type	Description
urlOrObject	string | object	The Discovery Document URL or parsed Discovery Document JSON object (Example).

Returns:

Type	Description
goog.Thenable	The return value is a **Promise**-like goog.Thenable object that resolves when the API interface is loaded.

---

gapi.client.load(*name, version, callback*)

Deprecated. Please load APIs with discovery documents. Loads the client library interface to a particular API. If a callback is not provided, a goog.Thenable is returned. The loaded API interface will be in the form gapi.client.*api.collection.method*. For example, the Moderator API would create methods like gapi.client.moderator.series.list.

Arguments:

Name	Type	Description
name	string	The name of the API to load.
version	string	The version of the API to load.
callback	function	(optional) the function that is called once the API interface is loaded. If not provided, a goog.Thenable is returned.

---

gapi.client.setApiKey(*apiKey*)

Sets the API key for the application, which can be found in the Developer Console. Some APIs require this to be set in order to work.

Arguments:

Name	Type	Description
apiKey	string	The API key to set.

---

gapi.client.setToken(*tokenObject*)

Sets the authentication token to use in requests. This should be used if the token was obtained without using the gapi.auth2 authentication library (for instance, when using Firebase to authenticate users).

Arguments:

Name	Type	Description
tokenObject	object	An object containing the access_token to use in API requests.

Name	Type	Description
access_token	string	The access token granted to the user.

API requests

gapi.client.request(*args*)

Creates a HTTP request for making RESTful requests.

Arguments:

Name	Type	Description
args	object	An object encapsulating the various arguments for this method. The path is required, the rest are optional. The values are described in detail below.

Name	Type	Description
path	string	The URL to handle the request.
method	string	The HTTP request method to use. Default is GET.
params	object	URL params in key-value pair form.
headers	object	Additional HTTP request headers.
body	string | object	The HTTP request body (applies to PUT or POST).

Returns:

Type	Description
gapi.client.Request | undefined	The returned gapi.client.Request object implements goog.Thenable and can be used like a **Promise** that fulfills with the response object or rejects with a reason object.

gapi.client.Request

An object encapsulating an HTTP request. This object is not instantiated directly, rather it is returned by gapi.client.request. There are two ways to execute a request. We recommend that you treat the object as a promise and use the then method, but you can also use the execute method and pass in a callback.

gapi.client.Request.then(*onFulfilled, onRejected, context*)

For more information about using promises, see Using Promises.

gapi.client.Request.execute(*callback*)

Executes the request and runs the supplied callback on response.

Arguments:

Name	Type	Description
callback(*jsonResp*, *rawResp*)	function	The callback function which executes when the request succeeds or fails. jsonResp contains the response parsed as JSON. If the response is not JSON, this field will be false. rawResp is the HTTP response. It is JSON, and can be parsed to an object which includes body, headers, status, and statusText fields.

Batch API requests

gapi.client.newBatch()

Creates a batch object for batching individual requests.

Returns:

Type	Description
gapi.client.Batch | undefined	The returned gapi.client.Batch implements goog.Thenable interface and can be used like a Promise that fulfills with a batch response object and rejects with a reason object.

gapi.client.Batch

Represents an HTTP Batch operation. Individual HTTP requests are added with the add method and the batch can be executed using then or execute. We recommend that you treat the batch object as a promise and use then. This class defines the following methods:

gapi.client.Batch.add(*request*,*opt_params*)

Adds a gapi.client.Request to the batch.

Arguments:

Name	Type	Description
request	gapi.client.Request	The HTTP request to add to this batch. This parameter is required.
opt_params	Object	Optional extra parameters for this batch entry. Accepted fields are id and callback:

Name	Type	Description
id	string	Identifies the response for this request in the map of batch responses. If one is not provided, the system generates a random ID.
callback(individualResponse, rawBatchResponse)	function	individualResponse is the response for this request only. Its format is defined by the API method being called. rawBatchResponse is the raw batch ID-response map as a string. It contains all responses to all requests in the batch.

gapi.client.Batch.then(*onFulfilled, onRejected, context*)

For more information about using promises, see Using Promises.

gapi.client.Batch.execute(*callback*)

Executes all requests in the batch. The supplied callback is executed on success or failure.

Name	Type	Description
callback(responseMap, rawBatchResponse)	function	The callback to execute when the batch returns. responseMap is an ID-response map of each requests response. rawBatchResponse is the same response, but as an unparsed JSON-string.

Auth setup

gapi.auth2.init(*params*)

Initializes the GoogleAuth object. You must call this method before calling gapi.auth2.GoogleAuth's methods.

When you initialize the GoogleAuth object, you configure the object with your OAuth 2.0 client ID and any additional options you want to specify. Then, if the user has already signed in, the GoogleAuth object restores the user's sign-in state from the previous session.

Arguments	

params	An object containing key-value pairs of client configuration data. See gapi.auth2.ClientConfig for the different properties configurable. For example:
	{
	  client_id: 'CLIENT_ID.apps.googleusercontent.com'
	}

Returns	
gapi.auth2. GoogleAuth	The gapi.auth2.GoogleAuth object. Use the then() method to get a Promise that is resolved when the gapi.auth2.GoogleAuth object finishes initializing.

GoogleAuth.then(*onInit*, *onError*)

Calls the *onInit* function when the GoogleAuth object is fully initialized. If an error is raised while initializing (this can happen in old unsupported browsers), the *onError* function will be called instead.

Arguments	
onInit	The function called with the GoogleAuth object when it is fully initialized.
onError	The function called with an object containing an error property, if GoogleAuth failed to initialize.

Returns	
Promise	A Promise that is fulfilled when the *onInit* function has completed, or rejected if an initialization error was raised. It resolves with the returned value from the *onInit* function, if any.

Warning: do not call Promise.resolve() and similar with the result of gapi.auth2.init(). As the GoogleAuth object returned implements the then() method that resolves with itself, it will create an infinite recursion.

Error codes

idpiframe_initialization_failed

Failed to initialize a required iframe from Google, for instance, due to an unsupported environment. A details property will give more information on the error raised.

gapi.auth2.ClientConfig

Interface that represents the different configuration parameters for the gapi.auth2.init method.

Parameters		
client_id	string	**Required.** The app's client ID, found and created in the Google Developers Console.
cookie_policy	string	The domains for which to create sign-in cookies. Either a URI, single_host_origin, or none. Defaults to single_host_origin if unspecified.
scope	string	The scopes to request, as a space-delimited string. Optional if fetch_basic_profile is not set to false.
fetch_basic_profile	boolean	Fetch users' basic profile information when they sign in. Adds 'profile', 'email' and 'openid' to the requested scopes. True if unspecified.
hosted_domain	string	The G Suite domain to which users must belong to sign in. This is susceptible to modification by clients, so be sure to verify the hosted domain property of the returned user. Use GoogleUser.getHostedDomain() on the client, and the hd claim in the ID Token on the server to verify the domain is what you expected.
openid_realm	string	Used only for OpenID 2.0 client migration. Set to the value of the realm that you are currently using for OpenID 2.0, as described in OpenID 2.0 (Migration).
ux_mode	string	The UX mode to use for the sign-in flow. By default, it will open the consent flow in a popup. Valid values are popup and redirect.
redirect_uri	string	If using ux_mode='redirect', this parameter allows you to override the default redirect_uri that will be used at the end of the consent flow. The default redirect_uri is the current URL stripped of query parameters and hash fragment.

Authentication

GoogleAuth is a singleton class that provides methods to allow the user to sign in with a Google account, get the user's current sign-in status, get specific data from the user's Google profile, request additional scopes, and sign out from the current account.

gapi.auth2.getAuthInstance()

Returns the GoogleAuth object. You must initialize the GoogleAuth object with gapi.auth2.init() before calling this method.

Returns	
gapi.auth2.GoogleAuth	The gapi.auth2.GoogleAuth object. Use this object to call gapi.auth2.GoogleAuth's methods.

GoogleAuth.isSignedIn.get()

Returns whether the current user is currently signed in.

Returns	
Boolean	true if the user is signed in, or false if the user is signed out or the GoogleAuth object isn't initialized.

GoogleAuth.isSignedIn.listen(*listener*)

Listen for changes in the current user's sign-in state.

Arguments	
listener	A function that takes a boolean value. listen() passes true to this function when the user signs in, and false when the user signs out.

GoogleAuth.signIn()

Signs in the user with the options specified to gapi.auth2.init().

Returns	
Promise	A Promise that is fulfilled with the GoogleUser instance when the user successfully authenticates and grants the requested scopes, or rejected with an object containing an error property if an error happened (see below for error codes).

Error codes

See GoogleAuth.signIn(*options*).

GoogleAuth.signIn(*options*)

Signs in the user using the specified options.

Arguments	
options	Either:

- A gapi.auth2.SignInOptions object containing key-value pairs of sign-in parameters. For example:

  ```
  {
    scope: 'profile email'
  }
  ```

- An instance of gapi.auth2.SigninOptionsBuilder. For example:

  ```
  options = new gapi.auth2.SigninOptionsBuilder();
  options.setAppPackageName('com.example.app');
  options.setFetchBasicProfile(True);
  options.setPrompt('select_account');
  options.setScope('profile').setScope('email');
  ```

Returns

Promise	A Promise that is fulfilled with the GoogleUser instance when the user successfully authenticates and grants the requested scopes, or rejected with an object containing an error property if an error happened (see below for error codes).

Error codes

popup_closed_by_user

The user closed the popup before finishing the sign in flow.

access_denied

The user denied the permission to the scopes required.

immediate_failed

No user could be automatically selected without prompting the consent flow. Error raised when using **signIn** with **prompt: 'none'** option. This option should not be required to use, as **gapi.auth2.init** will automatically sign in the user if previously signed in during a previous session.

gapi.auth2.SignInOptions

Interface that represents the different configuration parameters for the GoogleAuth.signIn(*options*) method.

Parameters

app_package_name	string	The package name of the Android app to install over the air. See Android app installs from your web site. Optional.
fetch_basic_profile	boolean	Fetch users' basic profile information when they sign in. Adds 'profile', 'email' and 'openid' to the requested scopes. Optional. True if unspecified.
prompt	string	Forces a specific mode for the consent flow. Optional. The possible values are: • consent The authorization server prompts the user for consent before returning information to the application. • select_account The authorization server prompts the user to select a Google account.

This allows a user who has multiple accounts to select amongst the multiple accounts that they may have current sessions for.

- none (**not recommended**)
  The authorization server will not display any authentication or user consent screens; it will return an error if the user is not already authenticated and has not previously consented to the requested scopes. As gapi.auth2.init will automatically sign in a user to the application if previously signed in, calling signIn({prompt: 'none'}) will usually fail.

scope	string	The scopes to request, as a space-delimited string, on top of the scopes defined in the gapi.auth2.init params. Optional if fetch_basic_profile is not set to false.
ux_mode	string	The UX mode to use for the sign-in flow. By default, it will open the consent flow in a popup. Valid values are popup and redirect.
redirect_uri	string	If using ux_mode='redirect', this parameter allows you to override the default redirect_uri that will be used at the end of the consent flow. The default redirect_uri is the current URL stripped of query parameters and hash fragment.

GoogleAuth.signOut()

Signs out the current account from the application.

Returns	
Promise	A Promise that is fulfilled when the user has been signed out.

GoogleAuth.disconnect()

Revokes all of the scopes that the user granted.

GoogleAuth.grantOfflineAccess(*options*)

Get permission from the user to access the specified scopes offline.

Arguments	

options	A gapi.auth2.OfflineAccessOptions object containing key-value pairs of parameters. For example:
	 {
	 scope: 'profile email'
	 }

Returns	

Promise	A Promise that is fulfilled when the user grants the requested scopes, passing an object containing the authorization code to the Promise's fulfillment handler. For example:
	 auth2.grantOfflineAccess().then(function(resp) {
	 var auth_code = resp.code;
	 });

Error codes

popup_closed_by_user

The user closed the popup before finishing the consent flow.

access_denied
: The user denied the permission to the scopes required.

immediate_failed
: No user could be automatically selected without prompting the consent flow. Error raised when using signIn with prompt: 'none' option. This option should not be required to use, as gapi.auth2.init will automatically sign in the user if previously signed in during a previous session.

---

gapi.auth2.OfflineAccessOptions

Interface that represents the different configuration parameters for the GoogleAuth.grantOfflineAccess(*options*) method.

Parameters

app_package_name	string	The package name of the Android app to install over the air. See Android app installs from your web site. Optional.
prompt	string	Forces a specific mode for the consent flow. Optional. The possible values are: • consent The authorization server prompts the user for consent before returning information to the application. • select_account The authorization server prompts the user to select a Google account. This allows a user who has multiple accounts to select amongst the multiple accounts that they may have current sessions for. Note that none is not available for the offline code flow.
scope	string	The scopes to request, as a space-delimited string, on top of the scopes defined in the gapi.auth2.init params. Optional if fetch_basic_profile is not set to false.

GoogleAuth.attachClickHandler(*container*, *options*, *onsuccess*, *onfailure*)

Attaches the sign-in flow to the specified container's click handler.

Arguments

container	The ID of, or a reference to, the div element to which to attach the click handler.
options	An object containing key-value pairs of parameters. See GoogleAuth.signIn().
onsuccess	The function to call after sign-in completes.
onfailure	The function to call if sign-in fails.

Users

A GoogleUser object represents one user account. GoogleUser objects are typically obtained by calling GoogleAuth.currentUser.get().

GoogleAuth.currentUser.get()

Returns a GoogleUser object that represents the current user. Note that in a newly-initialized GoogleAuth instance, the current user has not been set. Use the currentUser.listen() method or the GoogleAuth.then() to get an initialized GoogleAuth instance.

Returns	
GoogleUser	The current user

GoogleAuth.currentUser.listen(*listener*)

Listen for changes in currentUser.

Arguments	
listener	A function that takes a GoogleUser parameter. listen passes this function a GoogleUser instance on every change that modifies currentUser.

GoogleUser.getId()

Get the user's unique ID string.

> Do not use the Google IDs returned by getId() to communicate the currently signed in user to your backend server. Instead, send ID tokens, which can be securely validated on the server.

Returns	
String	The user's unique ID

GoogleUser.isSignedIn()

Returns true if the user is signed in.

Returns	
Boolean	True if the user is signed in

GoogleUser.getHostedDomain()

Get the user's G Suite domain if the user signed in with a G Suite account.

Returns	
String	The user's G Suite domain

GoogleUser.getGrantedScopes()

Get the scopes that the user granted as a space-delimited string.

Returns	
String	The scopes granted by the user

GoogleUser.getBasicProfile()

Get the user's basic profile information.

> Do not use the user's profile information to communicate the currently signed in user to your backend server. Instead, send ID tokens, which can be securely validated on the server.

Returns	
gapi.auth2.BasicProfile	You can retrieve the properties of gapi.auth2.BasicProfile with the following methods: • BasicProfile.getId() • BasicProfile.getName() • BasicProfile.getGivenName() • BasicProfile.getFamilyName() • BasicProfile.getImageUrl() • BasicProfile.getEmail()

GoogleUser.getAuthResponse(*includeAuthorizationData*)

Get the response object from the user's auth session.

Arguments	
includeAuthorizationData	**Optional:** A boolean that specifies whether to always return an access token and scopes. By default, the access token and requested scopes are not returned when fetch_basic_profile is true (the default value) and no additional scopes are requested.

Returns	
gapi.auth2.AuthResponse	A gapi.auth2.AuthResponse object.

GoogleUser.reloadAuthResponse()

Forces a refresh of the access token, and then returns a Promise for the new AuthResponse.

Returns	
Promise	A Promise that is fulfilled with the reloaded gapi.auth2.AuthResponse when reloading the OAuth token is done.

gapi.auth2.AuthResponse

The response returned when calling GoogleUser.getAuthResponse(*includeAuthorizationData*) or GoogleUser.reloadAuthResponse() methods.

Properties

access_token	string	The Access Token granted.
id_token	string	The ID Token granted.
scope	string	The scopes granted in the Access Token.
expires_in	number	The number of seconds until the Access Token expires.
first_issued_at	number	The timestamp at which the user first granted the scopes requested.
expires_at	number	The timestamp at which the Access Token will expire.

GoogleUser.hasGrantedScopes(*scopes*)

Returns true if the user granted the specified scopes.

Arguments

scopes	A space-delimited string of scopes.

Returns

Boolean	True if the scopes were granted

GoogleUser.grant(*options*)

Request additional scopes to the user.

See
GoogleAuth.signIn()
for the list of parameters and the error code.

GoogleUser.grantOfflineAccess(*scopes*)

Get permission from the user to access the specified scopes offline. When you use GoogleUser.grantOfflineAccess(), the sign-in flow skips the account chooser step.

See
GoogleAuth.grantOfflineAccess()
for more details on the method.

GoogleUser.disconnect()

Revokes all of the scopes that the user granted for the application.

UI elements

gapi.signin2.render(*id*, *options*)

Renders a sign-in button in the element with the given ID, using the settings specified by the *options* object.

Arguments

id	The ID of the element in which to render the sign-in button.
options	An object containing the settings to use to render the button. For example: ``{ scope: 'email', width: 200, height: 50, longtitle: true, theme: 'dark', onsuccess: handleSuccess, onfailure: handleFailure }`` You can specify the following options: (see Parameters below)

```
{
  scope: 'email',
  width: 200,
  height: 50,
  longtitle: true,
  theme: 'dark',
  onsuccess: handleSuccess,
  onfailure: handleFailure
}
```

You can specify the following options:

Parameters

scope	The scopes to request when the user signs in (default: `profile`).
width	The width of the button in pixels (default: `120`).
height	The height of the button in pixels (default: `36`).
longtitle	Display long labels such as "Sign in with Google" rather than "Sign in" (default: `false`). When you use long titles, you should increase the width of the button from its default.
theme	The color theme of the button: either `light` or `dark` (default: `light`).
onsuccess	The callback function to call when a user successfully signs in. This function must take one argument: an instance of `gapi.auth2.GoogleUser` (default: none).
onfailure	The callback function to call when sign-in fails. This function takes no arguments (default: none).
app_package_name	The package name of the Android app to install over the air. See Android app installs from your web site. Optional. (default: none)

Advanced

Warning: this section covers features that are not recommended for most use cases. Make sure that the methods described in the Guides don't work for your use case before using such features.

gapi.auth2.authorize(*params*, *callback*)

Performs a one time OAuth 2.0 authorization. Depending on the parameters used, this will open a popup to the Google sign-in flow or try to load the requested response silently, without user interaction.

Some use cases where this method is useful include:

- Your application only needs to requests a Google API endpoint once, for instance to load the user's favorite YouTube videos the first time they sign in.
- Your application has its own session management infrastructure, and it only requires the ID Token once to identify the user in your backend.
- Several Client IDs are used within the same page.

> **Warning:** do not use this method alongside the recommended gapi.auth2.init and signIn flow. These are two distinct behaviors (Authorization for gapi.auth2.authorize vs Authentication for gapi.auth2.init/signIn) and will have unexpected issues if used within the same application.

Arguments	

params	An object containing key-value pairs of configuration data. See gapi.auth2.AuthorizeConfig for the different properties configurable. For example:
	{
	  client_id: 'CLIENT_ID.apps.googleusercontent.com',
	  scope: 'email profile openid',
	  response_type: 'id_token permission'
	}

callback	A function called with a gapi.auth2.AuthorizeResponse object after the request has been completed (either successfully or with a failure).

Example

```
gapi.auth2.authorize({
  client_id: 'CLIENT_ID.apps.googleusercontent.com',
  scope: 'email profile openid',
  response_type: 'id_token permission'
}, function(response) {
  if (response.error) {
    // An error happened!
    return;
  }
  // The user authorized the application for the scopes requested.
  var accessToken = response.access_token;
  var idToken = response.id_token;
  // You can also now use gapi.client to perform authenticated requests.
});
```

Error codes

idpiframe_initialization_failed
: Failed to initialize a required iframe from Google, for instance, due to an unsupported environment. A `details` property will give more information on the error raised.

popup_closed_by_user
: The user closed the popup before finishing the sign in flow.

access_denied
: The user denied the permission to the scopes required.

immediate_failed
: No user could be automatically selected without prompting the consent flow. Error raised when using `signIn` with `prompt: 'none'` option.

---

gapi.auth2.AuthorizeConfig

Interface that represents the different configuration parameters for the gapi.auth2.authorize method.

Properties

client_id	string	**Required**. The app's client ID, found and created in the Google Developers Console.
scope	string	**Required**. The scopes to request, as a space-delimited string.
response_type	string	A list of space-delimited response type. Defaults to `'permission'`. The possible values are: • `id_token`, to retrieve an ID Token • `permission` (or `token`), to retrieve an Access Token • `code`, to retrieve an Authorization Code
prompt	string	Forces a specific mode for the consent flow. The possible values are: • `consent` The authorization server prompts the user for consent before returning information to the application. • `select_account` The authorization server prompts the user to select a Google account. This allows a user who has multiple accounts to select amongst the multiple accounts that they may have current sessions for. • `none` The authorization server will not display any authentication or user consent screens; it will return an error if the user is not already authenticated and has not previously consented to the requested scopes. If `code` is requested as response type, the code returned will only be exchangeable for an `access_token`, not a `refresh_token`. **Note :** under the hood, the library caches and automatically refreshes the last Access Token obtained. When using `prompt: 'none'`, most of the time, no HTTP request to the backend will be necessary to obtain a valid token.

cookie_policy	string	The domains for which to create sign-in cookies. Either a URI, single_host_origin, or none. Defaults to single_host_origin if unspecified.
hosted_domain	string	The G Suite domain to which users must belong to sign in. This is susceptible to modification by clients, so be sure to verify the hosted domain property of the returned user.
login_hint	string	The email, or User ID, of a user to pre-select in the sign-in flow. This is susceptible to modification by the user. This does not apply when prompt: "none" is used.
app_package_name	string	The package name of the Android app to install over the air. See Android app installs from your web site.
openid_realm	string	Used only for OpenID 2.0 client migration. Set to the value of the realm that you are currently using for OpenID 2.0, as described in OpenID 2.0 (Migration).
include_granted_scopes	boolean	Whether to request an Access Token that includes all the scopes previously granted by the user to the app, or only the scopes requested in the current call. Defaults to true.

gapi.auth2.AuthorizeResponse

The response returned to the callback of the gapi.auth2.authorize method.

Properties

access_token	string	The Access Token granted. Only present if permission or token was specified in the response_type.
id_token	string	The ID Token granted. Only present if id_token was specified in the response_type.
code	string	The Authorization Code granted. Only present if code was specified in the response_type.
scope	string	The scopes granted in the Access Token. Only present if permission or token was specified in the response_type.
expires_in	number	The number of seconds until the Access Token expires. Only present if permission or token was specified in the response_type.
first_issued_at	number	The timestamp at which the user first granted the scopes requested. Only present if permission or token was specified in the response_type.
expires_at	number	The timestamp at which the Access Token will expire. Only present if permission or token was specified in the response_type.
error	string	When the request failed, this contains the error code.
error_subtype	string	When the request failed, this can contain additional information to the error code also returned.

Viewed using Just Read