OAuth 2.0 Compliant REST API

The following document is a guide for using Gigya’s REST API conforming with the OAuth 2.0 standard. The REST API Reference provides specification to the various REST API methods supported by Gigya.

Note: Gigya offers a set of SDKs that wrap around Gigya’s REST API. The SDKs make it simple to integrate Gigya’s service for various development environments. Check out our SDKs documentation and find out if there is an SDK available for your preferred language. Otherwise, please continue with this guide and learn how to use our REST API directly.

The OAuth 2.0 Protocol

The OAuth 2.0 specifies an authorization flow prior to using the REST API methods. Your application must obtain authorization in order to access the user’s social profile or perform social activities, such as publishing newsfeed.

OAuth 2.0 supports many options in the authorization flow for different use cases. In this section we describe the main use cases and their corresponding flows.

Use case 1 - Smart Client Application Flow

This use-case applies to Social Login using Gigya’s socialize APIs (without RaaS).

This is the use case of a smart client application, such as mobile or desktop client application. In this case the client application communicates directly with our REST API.

Implementation Steps

1. Redirect to Login page
   (parameter response_type="token")

2. Response (receive access_token)

3. Call REST API method (pass access_token)

4. Receive method response
1. Login the user via the Gigya service, by redirecting the user to our login end-point URL. Pass the response_type="token" parameter. Read more in the Login end-point section below.
2. In the Login response you will receive the access_token. This token is your authorization ticket for accessing the current user’s social profile.
3. Pass the access_token with each REST API method call, related with the current user. Read how to make an API call below.
4. Receive REST API method response.

Use Case 2 - Direct Server to Server Flow

This use-case applies when using Gigya Registration-as-a-Service (RaaS).

In this use case the REST API calls are not bound to an active user (there is no active session). There are two options in this use case:

- For methods which must be associated to a specific user (such as setStatus, getFriendsInfo, etc.), retrieve the user’s UID from your DB and pass it as a parameter.
- For methods which are not associated with a user (such as reports.getSocializeStats), there is no need to pass a UID parameter.

![Diagram](image)

Implementation Steps

Your Client Application:

1. Login the user via the Gigya service, using our Web SDK method calls: socialize.login or socialize.notifyLogin. You may also use our ready made Login add-on.
2. In the Login response you will receive the User object.

Notes:

a. To learn how to implement client side login using our Web SDK, see Social Login.

b. An alternative implementation to steps 1 & 2 is to use the Login end-point to log in the user (as described in use case 1), and then use the socialize.getUserInfo to retrieve the User data.
3. Pass the User object to your server application.

**Your Server Application:**

1. Obtain the access_token using the getToken end-point and pass the grant_type="none" parameter. The getToken method must be called over HTTPS. Read more in the section below.
2. Receive the getToken method response. The response will contain the access_token. Note that in this use case, the token is not related to a specific user.
3. Pass the access_token with each REST API method call. If the method call is associated with a specific user, pass the user's UID as a parameter. Read how to make an API call below.
4. Receive REST API method response.

**Alternate Use Case 2 - Hybrid Client/Server Flow**

As above, in this case you use both the client and server applications to communicate with Gigya's REST and Client API's, but in reverse order. First the server requests an access token, then that token is set as a global Configuration object when loading the Gigya Web SDK.

**Implementation steps**

1. Obtain the access_token using the getToken end-point and pass the grant_type="none" parameter. The getToken method must be called over HTTPS. Read more in the section below.
2. Receive the getToken method response. The response will contain the access_token.

**Your Client Application:**

1. Load the Gigya Web SDK, passing the access_token as a global Configuration object. For example:

```javascript
// Load Gigya dynamically with an oauth_token
function loadGigya(accessToken){
    var s = document.createElement('script');
    s.type = 'text/javascript';
    s.async = true;
    s.src = "http://cdn.gigya.com/js/gigya.js?apikey=<Your-API-Key>";
    s.text = '{oauth_token: "' + accessToken + '"}';
    document.getElementsByTagName('head')[0].appendChild(s);
}
```

2. Gigya services will now be loaded with the user already logged in.

**Use Case 3 - Web Server Flow**

This use-case applies to Social Login using Gigya's socialize APIs (without RaaS).

These use cases may be useful for web sites that require extra security, for example, eCommerce sites. In this use case the server application performs the REST API calls. Since it is not safe to transfer the access_token from the client to the server, we will use a code instead. Your server will use the code to retrieve the access_token over HTTPS.

**Note:** Gigya supports this flow to conform with the OAuth 2.0 standard. Nevertheless, we encourage you to prefer Case 2 over this flow.
Implementation Steps

Your Client Application:

1. Login the user via the Gigya service, using the Login Plugin or the socialize.login API method of our Web SDK. **Important**: Set the authCodeOnly parameter to true.
2. In the onLogin event data (or in the socialize.login API method response) you will receive the authCode.
3. Pass the authCode to your server application. (Alternatively, you may redirect the user to our login end-point URL. Pass the response_type="code" parameter. Receive the code in the response, and pass it to the server. Read more in the Login end-point section below.)

Your Server Application:

1. Obtain the access_token using the getToken end-point. Pass the code as a parameter and the grant_type="authorization_code" parameter. The getToken must be called over HTTPS. Read more in the section below.
2. Receive the getToken method response. The response will contain the access_token. This token is your authorization ticket for accessing the current user's social profile.
3. Pass the access_token with each REST API method call, related to the user. Read how to make an API call below.
4. Receive the REST API method response.

The Login End-Point

Gigya provides an end-point for the user to log in using the user's social network or webmail account. This login end-point requires user interaction. To log the user in, redirect the user to the specified URL and pass the required Parameters.

The login URL shows the login screen of the requested provider. In some cases, such as Facebook, users will also be asked to give the site permission to access their personal data. When the login process completes, Gigya will redirect the user to the specified URL.

The login operation is currently supported by the following providers: Amazon, Blogger, Facebook, FourSquare, GooglePlus, Instagram, Kakao, LINE, LinkedIn, Livedoor, Messenger, mixi, Naver, Netlog, Odnoklassniki, Orange France, PayPalOAuth, Tencent QQ, Renren, Sina Weibo, Spiceworks, Twitter, VKontakte, WeChat, WordPress, Xing, Yahoo, Yahoo Japan.
The Login End-Point URL

https://socialize.<Data_Center>/socialize.login

Where <Data_Center> is:

- us1.gigya.com - For the US data center.
- eu1.gigya.com - For the European data center.
- au1.gigya.com - For the Australian data center.
- ru1.gigya.com - For the Russian data center.
- cn1.gigya-api.cn - For the Chinese data center.

If you are not sure of your site's data center, see Finding Your Data Center.

Code Example

In the following example, clicking the button opens the URL in a new window:

```html
<script type="text/javascript">
    function open_win() {
        window.open("https://socialize.us1.gigya.com/socialize.login?client_id=3_mK2cmEzlkhqe4MurtPncxbv4wKSvAHsew0iVwhVULs3c_Jx_HLX434jAVpq5M_W6"+"redirect_uri=http%3A%2F%2Fdevelopers.gigya.com%2Fdisplay%2FGD%2BLogin%2BSuccess%2BPage&amp;x_provider=facebook&amp;response_type=token");
    }
</script>
<form>
<input type=button value="Login to Facebook" onclick="open_win()">
</form>
```

Working Example

(Note that the Logout button, above, will only log you out from Gigya, and not any social networks you are currently connected to).

Parameters

<table>
<thead>
<tr>
<th>Required</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_provider</td>
<td>string</td>
<td></td>
<td>The provider which will be used for authenticating the user. The following values are currently supported for use with this parameter: amazon, blogger, facebook, foursquare, googleplus, instagram, kakao, line, linkedin, livedoor, messenger, mixi, naver, netlog, odnoklassniki, orangefrance, paypal, qq, renren, sina, spiceworks, twitter, vkontakte, wechat, wordpress, xing, yahoo, yahoojapan.</td>
</tr>
</tbody>
</table>
The Gigya API key. You may obtain a Gigya API key from the Site Dashboard page on Gigya's website.

The URL to redirect to after the login completes. This must be verified against the registered domain for this API key.

May be either "token" or "code". See extended explanation below.

A string that will be passed as another parameter to the redirect_uri after the login completes.

A context ID of the operation.

The default value of this parameter is false. If set to true, the account is not considered final until socialize.notifyRegistration is called. While being not-final the identities associated with this account can be connected to another account without causing an error.

Gigya no longer supports temporary users.

Defines the language of Gigya's user interface and error message. For the list of languages supported, please refer to the languages table.

When the login process completes (either successfully or with error), Gigya server redirects the user to the URL defined in the redirect_uri parameter and appends response parameters.

The login end-point defines two types of responses: "token" or "code". The "token" response type is used in Use case 1 and the "code" response type is used in Use case 3. Please, specify the response type using the response_type required parameter.

When the requested response_type is "token", the response parameters are appended as a URL fragment (following a # sign). The following parameters are appended, after a successful login:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access_token</td>
<td>string</td>
<td>A Gigya session key. You may use this token to invoke Gigya's REST API methods. Pass this token as a parameter with each REST API method call. Read how to make an API call below.</td>
</tr>
<tr>
<td>expires_in</td>
<td>number</td>
<td>The duration in seconds of the access token's lifetime. Note: This field is only returned when the token has an expiration time. If this field is not available then the token will not expire.</td>
</tr>
<tr>
<td>state</td>
<td>string</td>
<td>The state string passed by your application as parameter to the login end-point. (See table of Parameters above).</td>
</tr>
</tbody>
</table>

When the requested response_type is "code", the response parameters are appended as query string parameters (following a ? sign). The following parameters are appended, after a successful login:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>string</td>
<td>A Gigya authorization code. You may use this code to obtain an access token. The expiration time of this code is 2 minutes.</td>
</tr>
<tr>
<td>state</td>
<td>string</td>
<td>The state string passed by your application as parameter to the login end-point. (See table of Parameters above).</td>
</tr>
</tbody>
</table>

If the login fails, Gigya will append the following parameters to the response URL (regardless of the response_type parameter value):

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
error | string | The OAuth 2.0 error code. For a complete list of error codes, see the OAuth 2.0 Error Codes table below.
error_description | string | The Gigya error code and error message (separated by a dash). For example: "500001 - Server error". For the complete list of Gigya error codes and descriptions, please refer to the Gigya Error Codes table.
state | string | The state string passed by your application as parameter to the login end-point. (See table of Parameters above).

OAuth 2.0 Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_request</td>
<td>The request is missing a required parameter, includes an unknown parameter or parameter value, or is otherwise malformed.</td>
</tr>
<tr>
<td>invalid_client</td>
<td>The client identifier provided is invalid.</td>
</tr>
<tr>
<td>unauthorized_client</td>
<td>The client is not authorized to use the requested response type.</td>
</tr>
<tr>
<td>redirect_uri_mismatch</td>
<td>The redirection URI provided does not match a pre-registered value.</td>
</tr>
<tr>
<td>access_denied</td>
<td>The end-user or authorization server denied the request.</td>
</tr>
<tr>
<td>unsupported_response_type</td>
<td>The requested response type is not supported by the authorization server.</td>
</tr>
</tbody>
</table>

Getting an Access Token

This method retrieves an access token.

Notes:
- You may invoke the getToken method only over HTTPS. Calls over HTTP will be rejected.
- Please use POST with this request and pass all parameters in the body of the request.

Your Application Identification

The getToken method requires an API Key and a Secret Key which are obtained from the Site Dashboard page on Gigya’s website. The Secret Key must be kept secret and never transmitted to an untrusted client or over insecure networks. There are two ways in which you may pass these keys with the getToken method:

1. Using HTTP Basic Authorization header (preferred method):

   The Authorization value should be constructed as follows: BASE64( <ApiKey> ":" <Secret> ).

   For example:
   ```
   POST /token HTTP/1.1
   Host: server.example.com
   Authorization: Basic PUT-YOUR-KEYS-STRING-HERE
   Content-Type: application/x-www-form-urlencoded
   ```

   1. Using query parameters:

      - `client_id` - The Gigya API key
      - `client_secret` - The Gigya Secret Key

The getToken Request URL
Where `<Data_Center>` is:

- `us1.gigya.com` - For the US data center.
- `eu1.gigya.com` - For the European data center.
- `au1.gigya.com` - For the Australian data center.
- `ru1.gigya.com` - For the Russian data center.
- `cn1.gigya-api.cn` - For the Chinese data center.

If you are not sure of your site's data center, see Finding Your Data Center.

### Parameters

The behavior of the `getToken` method is determined by the `grant_type` parameter.

There are two types of behaviors: "grant_type=none" or "grant_type=authorization_code". The "none" grant type is used in Use case 2 and the "authorization_code" grant type is used in Use case 3. The set of parameters which the method receives depends on the grant type.

#### Parameters for `grant_type=authorization_code`:

<table>
<thead>
<tr>
<th>Required</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>grant_type</code></td>
<td>string</td>
<td><code>authorization_code</code></td>
</tr>
<tr>
<td></td>
<td><code>code</code></td>
<td>string</td>
<td>The verification code received from the login response (Read more in Use case 3 flow).</td>
</tr>
<tr>
<td></td>
<td><code>redirect_uri</code></td>
<td>string</td>
<td>This parameter is used for validation. The URL must be the same redirect_uri provided in the login request (Read more in Use case 3 flow).</td>
</tr>
<tr>
<td></td>
<td><code>regSource</code></td>
<td>string</td>
<td>Records the source of the registration. The default value is the URL of the current page but it can be any string value. regSource is stored in the account and can be used by verification emails to determine which page should be opened (see accounts.set Policies). Can also be set via the Global Conf object.</td>
</tr>
</tbody>
</table>

#### Parameters for `grant_type=client_credentials`:

(You may use a user or application key and corresponding secret only using this method)

<table>
<thead>
<tr>
<th>Required</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>grant_type</code></td>
<td>string</td>
<td><code>client_credentials</code></td>
</tr>
<tr>
<td></td>
<td><code>x_siteUID</code></td>
<td>string</td>
<td>You may provide the UID of the user that has been logged in by your site. This will generate an access token associated with the user and will allow you to invoke REST API methods, without passing the UID parameter. When passing this parameter, this method becomes equivalent to the <code>socialize.notifyLogin</code> API method, notifying the Gigya service that the user has been logged-in by the site. Note that any providerSessions parameter will be ignored.</td>
</tr>
</tbody>
</table>
x_userInfo | JSON string | This parameter is relevant only if the x_siteUID parameter (see above) is specified. This parameter allows you to provide Gigya with site's user profile data. This will ensure consistent user experience. Gigya will use this information, for example, in Plugins that show user info, such as the Chat and the Comments Plugins. The object may include the following fields: nickname, photoURL, thumbnailURL, firstName, lastName, gender, age, email. For example:

```
x_siteUID=12345&userInfo;={firstName:"David", gender:"male", age:30}
```

x_sessionExpiration | integer | The time in seconds that Gigya should keep the session valid for the user. To end the session when the browser closes, assign the value '0'. If this parameter is not specified, the session will be valid forever.

regSource | string | Records the source of the registration. The default value is the URL of the current page but it can be any string value. regSource is stored in the account and can be used by verification emails to determine which page should be opened (see accounts.set Policies). Can also be set via the Global Conf object.

Parameters for grant_type=none:
(You may use a user or application key and corresponding secret only using this method)

<table>
<thead>
<tr>
<th>Required</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>grant_type</td>
<td>string</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>x_siteUID</td>
<td>string</td>
<td>You may provide the UID of the user that has been logged in by your site. This will generate an access token associated with the user and will allow you to invoke REST API methods, without passing the UID parameter. When passing this parameter, this method becomes equivalent to the socialize.notifyLogin API method, notifying the Gigya service that the user has been logged-in by the site. Note that any providerSessions parameter will be ignored.</td>
</tr>
</tbody>
</table>
|          | x_userInfo | JSON string | This parameter is relevant only if the x_siteUID parameter (see above) is specified. This parameter allows you to provide Gigya with site's user profile data. This will ensure consistent user experience. Gigya will use this information, for example, in Plugins that show user info, such as the Chat and the Comments Plugins. The object may include the following fields: nickname, photoURL, thumbnailURL, firstName, lastName, gender, age, email. For example:

```
x_siteUID=12345&userInfo;={firstName:"David", gender:"male", age:30}
```

|          | x_sessionExpiration | integer | The time in seconds that Gigya should keep the session valid for the user. To end the session when the browser closes, assign the value '0'. If this parameter is not specified, the session will be valid forever. |
|          | regSource | string | Records the source of the registration. The default value is the URL of the current page but it can be any string value. regSource is stored in the account and can be used by verification emails to determine which page should be opened (see accounts.set Policies). Can also be set via the Global Conf object. |

Response

Upon successful generation of the access token you will receive a response in JSON format including the following parameters:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access_token</td>
<td>string</td>
<td>A Gigya session key. You may use this token to invoke Gigya's REST API methods. Pass this token as a parameter with each REST API method call. Read how to make an API call below.</td>
</tr>
</tbody>
</table>
expires_in  number  The duration in seconds of the access token's lifetime. Note: This field is only returned when the token has an expiration time. If this field is not available then the token will not expire.

For example:

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
{
  "access_token":"SlAV32hkKG",
  "expires_in":3600
}

Error Response

If the getToken method fails, Gigya will append the following parameters to the response:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>string</td>
<td>The OAuth 2.0 error code. For a complete list of error codes, see the ‘OAuth 2.0 Error Codes’ table below.</td>
</tr>
<tr>
<td>error_description</td>
<td>string</td>
<td>The Gigya error code and error message (separated by a dash). For example: &quot;500001 - Server error&quot;. For the complete list of Gigya error codes and descriptions, please refer to the Gigya Error Codes table.</td>
</tr>
<tr>
<td>state</td>
<td>string</td>
<td>The state string passed by your application as parameter to the login end-point. (See Login's table of Parameters above).</td>
</tr>
</tbody>
</table>

OAuth 2.0 Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_request</td>
<td>The request is missing a required parameter or includes an unknown parameter or parameter value or includes repeated parameter or includes multiple credentials or utilizes more than one mechanism for authenticating the client or is otherwise malformed.</td>
</tr>
<tr>
<td>invalid_client</td>
<td>The client identifier provided is invalid or the client failed to authenticate or the client provided multiple client credentials.</td>
</tr>
<tr>
<td>unauthorized_client</td>
<td>The client is not authorized to use the access grant type provided.</td>
</tr>
<tr>
<td>invalid_grant</td>
<td>The provided access grant is invalid, expired, or revoked (e.g. invalid assertion, expired authorization token, bad end-user basic credentials, or mismatching authorization code and redirection URI).</td>
</tr>
<tr>
<td>unsupported_grant_type</td>
<td>The access grant included, its type or another attribute, is not supported by the authorization server.</td>
</tr>
</tbody>
</table>

Making an API Call

A Sample API Call:

The following is an example of an HTTP GET request:
The call is composed of the following elements:

1. The Gigya REST API URL:
   "http://<namespace>.<Data_Center>/<namespace>.<apiMethod>"
   - Where <namespace> is:
     - accounts - For the RaaS Accounts API
     - comments - For the Comments API
     - ds - For the Data Store API
     - fidm - For the Identity Management APIs (SAML / OIDC)
     - gm - For the Loyalty (Game Mechanics) API
     - ids - For the Profile Management API
     - reports - For the Reports API
     - socialize - For the Socialize API
   - Where <Data_Center> is:
     - us1.gigya.com - For the US data center.
     - eu1.gigya.com - For the European data center.
     - au1.gigya.com - For the Australian data center.
     - ru1.gigya.com - For the Russian data center.
     - cn1.gigya-api.cn - For the Chinese data center.
   - If you are not sure of your site's data center, see Finding Your Data Center.
2. The method name, which in the above example, would be:
   setStatus
3. Method parameters. In the above example:
   ?status=Hello

Note: all parameter values should be in UTF-8 and URL-encoded.

Passing the Access Token

As explained above the Access Token is your authorization ticket for accessing social data. The Access Token must be passed with each REST API call. There are two methods of passing the Access Token:

1. Using the Authorization HTTP header (preferred method):
   GET /socialize.XXX HTTP/1.1
   Host: server.example.com
   Authorization: OAuth PUT-YOUR-ACCESS-TOKEN-STRING-HERE

2. Passing the access token as a request parameter named "oauth_token".
   In this case, the request from the sample API call above would be:


Receiving Responses

After sending a REST request, you will receive a response, which is a JSON object.

JSON Response Example:
"socialize.removeConnectionResponse":
{
    "statusCode": 200,
    "errorCode": 0,
    "statusReason": "OK",
    "callId": "72fba4d7ff1041c180f0ea38f37dd24f"
}

The name of the root element in the response consists of the name of the request API method with the word "Response" appended to it, as demonstrated in the example above: (in the response for socialize.removeConnection method).

All responses include two child elements, <statusCode> and <statusReason>, which comply with the HTTP status and status reason:

- <statusCode> - This is the equivalent of the HTTP status code, such as 200, 400, 40, etc.
- <statusReason> - The equivalent of the HTTP reason code, such as "OK", "Bad Request", "Unauthorized", etc.

Some methods also return additional data as additional children of the root element. See the documentation of each method for details on the specific data it returns.

A field that does not contain data will not appear in the response.

### Error Handling

If an error occurs, the response will include two additional child elements:

- <errorCode> - An application-specific error code. This value is used to distinguish between different causes for the same statusCode.
- <errorMessage> - A detailed textual error message designated for the purpose of logging during the process of debugging.

**JSON Response with Error Example:**

```
{
    "socialize.setStatusResponse":
    {
        "errorMessage": "Missing required parameter",
        "errorDetails": "Missing required parameter: uid",
        "statusCode": 400,
        "errorCode": 400002,
        "statusReason": "Bad Request",
        "callId": "dd1fc79e451b4dbacefaf1072e91aff"
    }
}
```

The system will return HTTP status "200 OK" for all application-level errors and return the detailed error in the body of the response. Please note that some errors, such as 503, or "Server Unreachable" will still be returned on the HTTP level and should be handled accordingly.

### Error Codes
For the complete list of error codes and descriptions, please refer to the Error Codes table.