RestAPI

The REST API allows your backend to make calls directly to your servers. There are two authentication mechanisms depending on the type of call:

App Access Token: This is a string composed of "OC|$APPID|$APPSECRET", where $APPID and $APPSECRET are per-application values that can be found on the "Platform" tab of the developer console.

User Access Token: a per-user value returned from the ovr_AccessToken_Get() SDK call. This is used to access content that the user owns.

The examples below use `curl` for clarity, but of course in real applications you will likely use your HTTP client library of choice.

For all GET APIs, the "fields" parameter can be customized to get just the fields you want.

Identity

1. Fetch the userid for a particular token.

```
$ curl -G -d "access_token=$USER_ACCESSTOKEN" https://graph.oculus.com/me
{"id":"1095130347203668"}
```

2. To confirm the identity of a particular user in your backend, pass the result of ovr_UserProof_Generate() and ovr_GetLoggedInUserID() to your your backend. You may verify the id is valid (e.g is entitled to your app and not forged) by running:

```
$ curl -d "access_token=$APP_ACCESSTOKEN" -d "nonce=..." -d "user_id=..."
  https://graph.oculus.com/user_nonce_validate
{"is_valid":true}
```

Note that the nonce is only good for one check and is then invalidated.

Leaderboards

1. Create or modify a leaderboard

```
$ curl -d "access_token=$APP_ACCESSTOKEN" -d "api_name=MY_NEW_LEADERBOARD"
  -d "sort_order=HIGHER_IS_BETTER" -d "entry_write_policy=CLIENT_AUTHORATATIVE"
  https://graph.oculus.com/$APPID/leaderboards
{"id":"1074233745960170"}
```

Parameters

- api_name: The name used to refer to the leaderboard in this API and in the client SDK.
- sort_order: Determines the order of entries in the leaderboard. Allowed Values

  "HIGHER_IS_BETTER" – Descending order. The entry with the highest score is rank 1.
  "LOWER_IS_BETTER" – Ascending order. The entry with the lowest score is rank 1.

- entry_write_policy: Determines who is allowed to write leaderboard entries

  "CLIENT_AUTHORITATIVE" - (default) Users can write to the leaderboard with the client SDK directly. Entries can still be modified with App Access Tokens.

  "SERVER_AUTHORITATIVE" - Entries can only be written with App Access Tokens. This is useful in cases where trusted servers are running the game simulation. In those cases, using this option can significantly reduce cheating because only trusted entities can write entries. The leaderboard can still be queries with the client SDK.

2. Query the metadata for a leaderboard

```
   $ curl -G -d "access_token=$APP_ACCESSTOKEN|$USER_ACCESSTOKEN" -d
"api_name=MY_NEW_LEADERBOARD"
     -d 'fields' => 'sort_order,entry_write_policy,entry_count'
     https://graph.oculus.com/$APPID/leaderboards
   {"data":[{"id":"1074233745960170", "sort_order":"HIGHER_IS_BETTER",
"entry_write_policy":"CLIENT_AUTHORATATIVE","entry_count":2500}]}
```

See the create API for the definition of the fields. The additional field "entry_count" provides the total count of entries on the leaderboard.

3. Delete a leaderboard. Note that the id parameter in the URL is the ID returned from the "create" or metadata query APIs above.

```
   $ curl -X DELETE -d "access_token=$APP_ACCESSTOKEN"
     https://graph.oculus.com/1074233745960170
   {"success":true}
```

4. Remove all entries from a leaderboard (Scheduled to ship in the weekly push on 3/1/16)

```
   $ curl -d "access_token=$APP_ACCESSTOKEN" -d "api_name=MY_NEW_LEADERBOARD"
     https://graph.oculus.com/leaderboard_remove_all_entries
   {"success":true}
```

5. Query a leaderboard

```
   $ curl -G -d "access_token=$APP_ACCESSTOKEN|$USER_ACCESSTOKEN" -d
"api_name=MY_NEW_LEADERBOARD"
     -d "filter=NONE" -d "start_at=OFFSET" -d "offset=10" -d "summary=true" -d
"limit=2"
     -d "fields=user{id,alias,profile_url},rank,score,timestamp,extra_data_base64"
     https://graph.oculus.com/leaderboard_remove_all_entries
   {"data":[{"id":"1074233745960170",
"user":{"id:865302060207175,"alias":"UnknownXuid","profile_url":"..."},"rank":25,"sco
re":12345,"timestamp":1456523020,"extra_data_base64":"T2N1bHVz"}, ...]
    "summary":{"total_count":45},
    "paging":{"next":"...","previous":"..."}}
```

Parameters

- api_name: Name of the leaderboard to query
- filter: (optional, default: "NONE")

  "NONE": Show all entries

  "FRIENDS": Show only entries from my friends. Note this option can only be used with User Access Tokens.
- start_at: (optional, default: "TOP")

  "TOP": Start at the first entry. This is the same as "OFFSET" with an "offset" of 0.

  "OFFSET": Starts at a particular entry. Use the "offset" parameter to specify which entry to start at.

  "CENTERED_ON_VIEWER": The first page will have the current user's entry in the center of the results. Only works with User Access Tokens. Will return an error if the current user has not posted on the leaderboard.
- offset: (optional) Specify when "start_at=OFFSET" to specify where to start. This is zero-based and the valid range is 0 to total_count -1. For example, if this is used with the friends filter, you might have 4 results with ranks of 75, 100, 125, and 150. In this example, "offset=2" would get the entries with rank 125 and 150.
- summary: (optional, default: false) Includes the summary node in the results, which gives you access to the total entries available for the current filter.
- limit: (optional, default: 100) Specifies the maximum number of entries to return. The max allowed value is 100.

Output

"data" contains at most "limit" entries. Available fields:

- user{id,alias,profile_url} – Gets the id, profile name, and url to the profile picture of the user who posted the entry.
- rank - The entry's rank relative to the current filter. The top entry on the leaderboard has rank 1.
- score - The entry's score.
- timestamp - Unix time for when the entry was posted
- extra_data_base64 - The entry's app-supplied metadata base64 encoded

If there are more entries available, the "paging" node in the output will contain a "next" field. That field is a URL that can be used to get the next page of entries. If you are not on the first page, there will also to be "previous" URL that can be used to page back.

In the "summary" node, "total_count" represents the total number of entries available for the current filter.

6. Submit a leaderboard entry

```
   $ curl -d "access_token=$APP_ACCESSTOKEN|$USER_ACCESSTOKEN" -d
"api_name=MY_NEW_LEADERBOARD"
```

```
      -d "score=12345" -d "extra_data_base64=T2N1bHVz" -d "force_update=true" -d
"user_id=865302060207175"
     https://graph.oculus.com/leaderboard_submit_entry
   {"success":true, "did_update":true}
```

Parameters

- api_name: Name of the leaderboard to post to
- score: The score being submitted
- extra_data_base64: (optional) Extra metadata to store on the row. This can be used to specify information about the score. For instance, in a driving game this might be what car was used. Decoded length can be at most 2048 bytes.
- force_update: (optional, default: false) By default, if you already have an entry on the leaderboard and post with a worse score than the existing entry, the existing entry will not be updated. You can use "force_update=true" to force the new entry even if it's worse than the old one.
- "user_id": When using an App Access Token this must be set to indicate which user you are posting on behalf of. That user must have an entitlement to your app. When using a User Access Token, this field must not be set.

Output

did_update indicates whether the entry was recorded or not. Entries will not be recorded if the user already has an entry on the leaderboard, the new score is worse than the old one, and force_update is false.

7. Delete a leaderboard entry. Note that the id parameter in the URL is the ID returned from the query leaderboard entries API above.

```
   $ curl -X DELETE -d "access_token=$APP_ACCESSTOKEN"
     https://graph.oculus.com/1074233745960170
   {"success":true}
```

Achievements

Achievement types

1. Configure achievement definitions

2. Query achievement definitions

3. Write achievements

4. Query achievements

5. Remove all achievements and progress from a user

In App Purchases

1. Verify the the user owns a specific IAP item

```
   $ curl -d "access_token=$USER_ACCESSTOKEN" -d "sku=some_sku"
https://graph.oculus.com/$APPID/verify_entitlement
   {"success":true}
```

2. Consume an IAP item

```
  $ curl  -d "access_token=$USER_ACCESSTOKEN" -d "sku=EXAMPLE1"
    https://graph.oculus.com/$APPID/consume_entitlement
  {"success":true}
```

After consumption the use is no longer entitled to the item:

```
   $ curl  -d "access_token=$USER_ACCESSTOKEN" -d "sku=EXAMPLE1"
       https://graph.oculus.com/$APPID/verify_entitlement
   {"success":false}
```

3. Query all IAP entitlements

```
   $ curl -G -d "access_token=$USER_ACCESSTOKEN"  -d "fields=id,item{sku}"
       https://graph.oculus.com/$APPID/viewer_purchases
   {"data":[{"id":"963119010431337","item":{"sku":"EXAMPLE1"}}]}
```

Rooms

1. Create a moderated room

```
   $ curl -d "access_token=$APP_ACCESSTOKEN"  -d "max_users=MY_MAX_USER_COUNT"
       https://graph.oculus.com/room_moderated_create
   {"id": 963119010431337}
```

2. Delete a moderated room

```
   $ curl  -X DELETE -d "access_token=$APP_ACCESSTOKEN"
       https://graph.oculus.com/MODERATED_ROOM_ID
   {"success": true}
```