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=$APP_ACCESSTOKEN" -d "api_name=MY_NEW_LEADERBOARD"
  -d "sort_order=HIGHER_IS_BETTER" -d "entry_write_policy=CLIENT_AUTHORITATIVE"
  -d "earliest_allowed_entry_time=1463875200" -d
"latest_allowed_entry_time=1464480000"
  https://graph.oculus.com/$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 queried with the client SDK.
- earliest_allowed_entry_time - (optional) Writes before this time are rejected. Can be used with latest_allowed_entry_time to create leaderboards that are active only during an event. The default is to allow entries to be posted at any time. If omitted on update, leaves the existing setting. Use 0 to disable the setting.
- latest_allowed_entry_time - (optional) Writes after this time are rejected. Works the same as earliest_allowed_entry_time.

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_AUTHORITATIVE","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

```
   $ 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_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

1. Create or modify an achievement definition

```
   $ curl -F "access_token=$APP_ACCESSTOKEN" -F "api_name=VISIT_3_CONTINENTS"
```

```
     -F "achievement_type=BITFIELD" -F
"achievement_write_policy=CLIENT_AUTHORITATIVE"
     -F "target=3" -F "bitfield_length=7" -F "is_archived=false" -F
"title=Achievement Title"
     -F "description=How to earn me" -F "unlocked_description_override=You did it"
     -F "is_secret=false" -F
"locked_image_file=@/path/to/locked_icon.png;type=image/png"
     -F  "unlocked_image_file=@/path/to/unlocked_icon.png;type=image/png"
     https://graph.oculus.com/$APPID/achievement_definitions
   {"id":"1074233745960170"}
```

Parameters

- api_name: The name used to refer to the achievement in this API and in the client SDK.
- achievement_type: Determines the behavior of the achievement. Allowed Values:

  "SIMPLE" – This is a simple binary achievement that is either earned or not.

  "COUNT" – This is an achievement that is unlocked when some target is reached. For instance, you might have an acheivement called "WALK_500_MILES" that unlocks when a counter hits 500.

  "BITFIELD" - This is an achievement used when there are a set of discrete things to accomplish. For instance, you might have an achievement "VISIT_3_CONTINENTS" and represent each continent in the world as a particular bit. This achievement would unlock when any 3 bits in the bitfield are set.
- achievement_write_policy: Determines who is allowed to write achievement progress

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

  "SERVER_AUTHORITATIVE" - Achievements 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 achievements. Achievement progress can still be queried with the client SDK.
- target (required for COUNT and BITFIELD achievements): For COUNT achievements, the threshold the progress counter must reach to unlock the achievement. In the above example this would be '500'. For BITFIELD achievements, the number of bits in the bitfield that must be set to unlock the achievement. In the above example this would be '3'.
- bitfield_length' (required for BITFIELD achievements): The size of the bitfield for this achievement. Since there are seven continents, in the above example this would be '7'.
- is_archived (optional): For data recoverability reasons, achievement definitions cannot be deleted. If one is created in error, you can set "is_archived=true" which will hide its existence from all clients. If an achievement is accidentally archived, it can be restored with "is_archived=false".
- title: The user-facing achievement title
- description: A description for the achievement. Often used to describe how to earn it.

- unlocked_description_override: A description to show in place of the above description when the user has earned the achievement. This may be used to hide spoilers from an achievement description until after it's been earned
- is_secret (optional): Users cannot see the existence of secret achievements, including the title, description, icon, and progress until they are earned. Default false.
- unlocked_image_file: The icon shown when the achievement is earned. Must be a 256x256 PNG.
- locked_image_file: The icon shown until the achievement is earned. Must be a 256x256 PNG. This image can be anything, but the best practice is for locked images to be grayscale, desaturated, or otherwise "greyed-out" versions of the unlocked image.

2. Query achievement definitions

```
   $ curl -G -d "access_token=$APP_ACCESSTOKEN|$USER_ACCESSTOKEN"
     -d 'api_names=["VISIT_3_CONTINENTS", "WALK_500_MILES"]' -d
"include_archived=true"
     -d
'fields=api_name,achievement_type,achievement_write_policy,target,bitfield_length,is_
archived,title,description,unlocked_description_override,is_secret,locked_image_uri,u
nlocked_image_uri'
     https://graph.oculus.com/$APPID/achievement_definitions
   {"data":[{"id":"1074233745960170", "api_name":"VISIT_3_CONTINENTS",
achievement_type":"BITFIELD", "achievement_write_policy":"CLIENT_AUTHORITATIVE",
"target":3, "bitfield_length":7, "is_archived":false, "title":"Achievement Title",
"description":"How to earn me", "unlocked_description_override":"You did it",
"is_secret":false, "locked_image_uri":"https://scontent.oculuscdn.com/...",
"unlocked_image_uri":"https://scontent.oculuscdn.com/...",}]}
```

See the create API for the definition of the fields. Note that the images are "locked_image_uri" and "unlocked_image_uri" instead of "locked_image_file" and "unlocked_image_file"

Parameters

- api_names (optional): The names of the achievement definitions to fetch. If omitted all achievement definitions are returned.
- include_archived (optional): True to include archived achievements. May only be used with APP_ACCESSTOKENs.

3. Write achievements

```
   $ curl -d "access_token=$APP_ACCESSTOKEN|$USER_ACCESSTOKEN" -d
"api_name=MY_ACHIEVEMENT"
      -d "add_bits=0011001" -d "add_count=25" -d "force_unlock=true"
     https://graph.oculus.com/$USERID/achievements
   {"id":"1074233745960170", "api_name":"MY_ACHIEVEMENT", "just_unlocked":true}
```

Note: The parameters use accumulation instead of setting new values directly. For instance, "add_count=25" instead of "set_count=100". This is so that conflicts that arise from updating achievements from multiple sources simultaneously or making progress from multiple devices in offline mode can be handled gracefully.

Parameters

- api_name: Name of the achievement to update
- add_count: Value to add to the progress counter for this achievement. Only valid for COUNT achievements.
- add_bits: Bits to add to the progress bitfield for this achievement. Bits are added to the existing bitfield with a logical OR operation. For example, if the previous bitfield was "0101" and "add_bits" is "1100", the new bitfield will be "1101". Only valid for BITFIELD achievements.
- force_unlock: (optional, default: false) Use true to unlock the achievement no matter what the current progress is. This must be used to unlock SIMPLE achievements.

Output

just_unlocked indicates if this operation caused the achievement to unlock.

4. Query achievements

```
   $ curl -G -d "access_token=$APP_ACCESSTOKEN|$USER_ACCESSTOKEN"
     -d 'api_names=["VISIT_3_CONTINENTS", "WALK_500_MILES"]'
     -d 'fields' =>
'definition{api_name,target},count_progress,bitfield_progress,is_unlocked,unlock_time
'
     https://graph.oculus.com/$USERID/achievements
   {"data":[{"id":"1074233745960170", "definition":{"api_name":"VISIT_3_CONTINENTS",
"target":3}, "count_progress":0, "bitfield_progress":"1001100", "is_unlocked":true,
"unlock_time":1459815726}]}
```

Parameters

- api_names (optional): The names of the achievements to fetch. If omitted all achievements that have progress are returned.

Output

"data" contains one entry for each achievement the user has any progress for. Available fields:

- definition{api_name,target} – Gets information from the achievement definition for this achievement. Any set of valid fields for achievement definitions may be used here.
- count_progress - For COUNT achievements, the current value of the counter. Will always be 0 for non-COUNT achievements.
- bitfield_progress - For BITFIELD achievements, a string of "1"s and "0"s representing the current state of the bitfield. Will always be empty for non-BITFIELD achievements.

- is_unlocked - A bool indicating if the achievement has been unlocked or not
- unlock_time - If the achievement is unlocked, a Unix time representing when the achievement was unlocked. 0 if the achievement has not been unlocked.

5. Remove all achievements and progress from a user

```
$ curl -d "access_token=$APP_ACCESSTOKEN" -d "user_id=$USERID"
  https://graph.oculus.com/achievement_remove_all
{"success":true}
```

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}
```