GXC API Library

Welcome to the GXC API Library! This is a marketplace asset containing functions for interacting with the GXC API. You can use this to submit challenge scores, retrieve information about the user's profile, challenges, scores and more.

This manual only contains information on using the functions given in this library. You can view the Opera GX section on the YoYo Games Helpdesk for tutorials on using GXC and creating challenges on GXC DevCloud.

Modules

Please look at the following sections for information on the different modules present in this library:

- General
- Challenge
- Profile

Miscellaneous

Please also read the following pages containing crucial information on the GXC library:

- Config Script
- Callbacks & Async Events
- Response Error Codes
- Library Errors

General Functions

This asset contains the following general functions:

- gxc_get_query_param

gxc_get_query_param

When your game is played on GXC, some extra parameters are passed into the URL of the game so they can be retrieved in-game. This function will return the value of the parameter specified in the first argument as a string.

The following parameter keys can be specified in the `key` argument:

Key	Value
game	The ID of the game
track	The ID of the track that is being played
challenge	The ID of the currently active challenge, or `undefined` if no challenge is active
username	The username of the current user
userId	The ID of the current user
avatarUrl	A URL to the current user's avatar image (can be used with `sprite_add()` to be displayed in-game)

Note that if the specified parameter is not present in the URL, this function will return `undefined`.

Syntax:

```
gxc_get_query_param(key);
```

Argument	Description
key	The parameter key to get the value of

Returns:

```
String (or undefined)
```

Example:

```
var _current_challenge = gxc_get_query_param("challenge");
var _highscore_challenge = "34esa3a1-e41e-4a9f-aaaa-4da7bd24ada2";

if (_current_challenge == _highscore_challenge)
{
    gxc_challenge_submit_score(global.highscore);
}
```

The above code retrieves the ID of the currently active challenge and checks whether it's equal to a specific challenge ID (meaning that challenge is active). In that case, it submits the current highscore as a new score to the challenge.

Here is an example where the profile data is retrieved and drawn in-game:

```
// Create event
username = gxc_get_query_param("username");

var _avatar_url = gxc_get_query_param("avatarUrl");

user_sprite = sprite_add(_avatar_url, 0, 0, 0, 0, 0);

// Draw event
if (sprite_exists(user_sprite))
{
    draw_sprite(user_sprite, 0, x, y);
    draw_text(x + 100, y, username);
}
```

The "Create event" code above retrieves the username and avatar URL for the current user, then uses `sprite_add()` to download the avatar image through the given URL and stores its new ID in a variable. In the Draw event, it checks whether the user sprite exists (which will be `true` after it has been downloaded) and if it does, draws the sprite along with the username.

> **NOTE**: Due to CORS, you will not be able to load the avatar image when the game is running locally. For it to work you are required to have uploaded your game to GXC and run it from there.

Challenges

Overview

A GXC game can have several challenges, only one of which can be active when the game is being played. The functions given below are used to interact with the currently active challenge (if there is one).

Note that the currently active challenge can be returned by calling `gxc_get_query_param("challenge")`.

Functions

The following functions are given for working with challenges:

- gxc_challenge_submit_score
- gxc_challenge_get_global_scores
- gxc_challenge_get_user_scores
- gxc_challenge_get_challenges

gxc_challenge_submit_score

Overview

This function is used to submit a new score to the currently active challenge, for the user that is currently logged in. You specify the score value to submit to the challenge, and an optional callback method which is called when a response arrives from the server. As explained on the Callbacks page, if you don't specify this argument, an Async Social event will be triggered when a response is received.

The function will always return a request ID, which can be used to identify the request in an Async Social event if a callback method is not specified.

Options

You can optionally specify a struct as the third argument, containing options for challenge identification. You can include the following variables in that struct:

Variable	Type	Value
challengeId	string	The challenge ID to use (if not specified, defaults to the currently active challenge)

See the example given at the bottom of this page for more information.

Callback Arguments

The callback method will receive two arguments: `_httpStatus` and `_result`, where the former is the status code for the HTTP response and the latter is a struct containing all returned data. The `_result` struct (or the `async_load` map in the Async Social event if a callback is not specified) will contain the following variables:

Variable	Type	Value
data	struct	A struct containing response data for the score submission request
errors	array	An array containing the errors that occurred for this request

It may have other variables too; see Callbacks & Async Events for more information.

The `data` struct contains the following variables:

Variable	Type	Value
newBestScore	boolean	Whether the submitted score was the new best score for the challenge

The `errors` array contains structs indicating errors with the request. Each error struct will have a `code` variable containing an error message, all of which are listed on this page.

Syntax:

```
gxc_challenge_submit_score(score, [callback], [options]);
```

Argument	Description
score	The new score value for the currently active challenge; should be an integer, if a decimal value is provided it will be rounded
callback	(Optional) A callback method which is called when an HTTP response is received
options	(Optional) A struct containing options for the submission

Returns:

```
Real (HTTP Request ID)
```

Example:

```
var _highscore_challenge = "34esa3a1-e41e-4a9f-aaaa-4da7bd24ada2";

if (gxc_get_query_param("challenge") == _highscore_challenge)
{
    gxc_challenge_submit_score(global.highscore);
}
```

The above code stores the ID of a challenge (retrieved from the GXC DevCloud website) and checks whether that challenge is currently active (by retrieving the `"challenge"` parameter from the URL). If that challenge is active, it submits the current highscore to the challenge, without a callback method (which is not required if all you need to do is submit a score).

If you need to confirm that the score was uploaded and then perform a certain action, you can make use of the callback function:

```
if (gxc_get_query_param("challenge") == _highscore_challenge)
{
    instance_create_layer(0, 0, "GUI", obj_loading_bar);

    gxc_challenge_submit_score(global.highscore, function (_status, _result)
    {
        if (_status == 200)
        {
            instance_destroy(obj_loading_bar);
            instance_create_layer(0, 0, "GUI", obj_upload_success);
        }
    });
}
```

The above code creates a "loading bar" instance before submitting the score, and uses a callback function to check whether the score was successfully uploaded to the server. In that case, it destroys the loading bar and creates an "upload success" object. (Note that ideally you would also want to handle what happens if an error is returned.)

The status code is being checked against `200` as that indicates that the request was successful.

If you need to submit a score to a specific challenge that is not the currently active challenge, you can use the `options` argument:

```
var _options = {
    challengeId: "cysdf906bh23rkhjm"; // Your challenge ID goes in the string
}

gxc_challenge_submit_score(global.otherScore, undefined, _options);
```

The above code submits a new score to the specified challenge. It uses no callback function (`undefined`) so no checking logic will be executed.

gxc_challenge_get_global_scores

Overview

This function is used to retrieve all the scores for the currently active challenge. Signing into GXC is not required for this function to work.

You specify an optional callback method which is called when a response arrives from the server. As explained on the **Callbacks** page, if you don't specify this argument, an Async Social event will be triggered when a response is received. You can also supply an optional struct as the second argument, containing parameters for pagination (returning only a specific amount of scores).

The function will always return a request ID, which can be used to identify the request in an Async Social event if a callback method is not specified.

Options

You can optionally specify a struct as the second argument, which allows you to apply pagination and target a specific challenge and track. You can include the following variables in that struct:

Variable	Type	Value
page	integer	The page number to return
pageSize	integer	The maximum number of scores to return
challengeId	string	The challenge ID to use (if not specified, defaults to the currently active challenge)
trackId	string	The track ID to use (if not specified, defaults to the currently active track)

See the example given at the bottom of this page for more information.

Callback Result

As explained on the **Callbacks & Async Events** page, the callback method will get a `_result` argument containing a `data` struct. The `data` struct for this function's response will contain the following variables:

Variable	Type	Value
scores	array	An array containing structs as the scores; these will be sorted from best to worst (so the best score will be the first entry in this array, and the worst will be the last)
challenge	struct	A struct containing information on the currently active challenge; for included variables see the table on gxc_challenge_get_challenges
pagination	struct	A struct containing information on pagination

The `_result.data.scores` array will contain a struct for each score returned, where each score struct will contain the following variables:

Variable	Type	Value
achievementDate	string	The date and time when this score was submitted
countryCode	string	The country code of the score submitter
player	struct	A struct containing the following information about the submitter:
- avatarUrl	string	A URL to the player's avatar
- userId	string	The player's user ID
- username	string	The player's username
score	integer	The score value
scoreId	integer	The ID of the submitted score

The `_result.data.pagination` struct will contain the following variables:

Variable	Type	Value
currPage	integer	The page number that was loaded
numPerPage	integer	The number of challenges that are being loaded per page
totalItems	integer	The total number of challenges for the game
totalPages	integer	The total number of pages that are available for the given page size

If pagination options are specified, the response may include pagination-specific errors that occur due to invalid values; please see Response Error Codes for information on such errors.

Syntax:

```
gxc_challenge_get_global_scores([callback, options])
```

Argument	Description
callback	(Optional) A callback method which is called when an HTTP response is received
options	(Optional) A struct containing options for pagination

Returns:

```
Real
```

Example:

```
var _options =
{
    page: 0,
    pageSize: 5
};

gxc_challenge_get_global_scores(function(_status, _result)
{
    if (_status == 200)
    {
        show_debug_message("Challenge: " + _result.data.challenge.name);

        var _scores_count = array_length(_result.data.scores);
        for (var i = 0; i < _scores_count; i ++)
        {
            var _score_data = _result.data.scores[i];
            show_debug_message("Score " + string(i) + " is " + _score_data.score);
        }
    }
}, _options);
```

The above code requests the top 5 global scores for the current challenge to be retrieved, and once they are successfully returned, loops through the array and prints the value of each score to the output log.

To load the scores for a different challenge (which is not being played at the moment), simply include its challenge ID in your options struct:

```
var _options =
{
    page: 0,
    pageSize: 5,
    challengeId: "cysdf906bh23rkhjm"
};

gxc_challenge_get_global_scores(function(_status, _result)
{
    // Callback
}, _options);
```

gxc_challenge_get_user_scores

Overview

This function is used to retrieve the current user's submitted scores for the currently active challenge. Signing into GXC is required for this function to work.

You specify an optional callback method which is called when a response arrives from the server. As explained on the **Callbacks** page, if you don't specify this argument, an Async Social event will be triggered when a response is received. You can also supply an optional struct as the second argument, containing parameters for pagination (returning only a specific amount of scores).

The function will always return a request ID, which can be used to identify the request in an Async Social event if a callback method is not specified.

Options

You can optionally specify a struct as the second argument, which allows you to apply pagination and target a specific challenge and track. You can include the following variables in that struct:

Variable	Type	Value
page	integer	The page number to return
pageSize	integer	The maximum number of scores to return
challengeId	string	The challenge ID to use (if not specified, defaults to the currently active challenge)
trackId	string	The track ID to use (if not specified, defaults to the currently active track)

See the example given at the bottom of this page for more information.

Callback Result

As explained on the **Callbacks & Async Events** page, the callback method will get a `_result` argument containing a `data` struct. The `data` struct for this function's response will contain the following variables:

Variable	Type	Value
scores	array	An array containing structs as the scores; these will be sorted from best to worst (so the best score will be the first entry in the array, and the worst will be the last)
challenge	struct	A struct containing information on the currently active challenge; for included variables see the table on gxc_challenge_get_challenges
pagination	struct	A struct containing information on pagination

The `_result.data.scores` array will contain a struct for each score returned, where each score struct will contain the following variables:

Variable	Type	Value
achievementDate	string	The date and time when this score was submitted
countryCode	string	The country code of the score submitter
player	struct	A struct containing the following information about the submitter:
- avatarUrl	string	A URL to the player's avatar
- userId	string	The player's user ID
- username	string	The player's username
score	integer	The score value
scoreId	integer	The ID of the submitted score

The `_result.data.pagination` struct will contain the following variables:

Variable	Type	Value
currPage	integer	The page number that was loaded
numPerPage	integer	The number of challenges that are being loaded per page
totalItems	integer	The total number of challenges for the game
totalPages	integer	The total number of pages that are available for the given page size

If pagination options are specified, the response may include pagination-specific errors that occur due to invalid values; please see Response Error Codes for information on such errors.

Syntax:

```
gxc_challenge_get_user_scores([callback, options])
```

Argument	Description
callback	(Optional) A callback method which is called when an HTTP response is received
options	(Optional) A struct containing options for pagination

Returns:

```
Real
```

Example:

```
var _options =
{
    page: 0,
    pageSize: 5
};

gxc_challenge_get_user_scores(function(_status, _result)
{
    if (_status == 200)
    {
        show_debug_message("Challenge: " + _result.data.challenge.name);

        var _scores_count = array_length(_result.data.scores);
        for (var i = 0; i < _scores_count; i ++)
        {
            var _score_data = _result.data.scores[i];
            show_debug_message("Score " + string(i) + " is " + _score_data.score);
        }
    }
}, _options);
```

The above code requests the top 5 scores of the user for the current challenge to be retrieved, and once they are successfully returned, loops through the array and prints the value of each score to the output log.

To load the scores for a different challenge (which is not being played at the moment), simply include its challenge ID in your options struct:

```
var _options =
{
    page: 0,
    pageSize: 5,
    challengeId: "cysdf906bh23rkhjm"
};

gxc_challenge_get_user_scores(function(_status, _result)
{
    // Callback
}, _options);
```

gxc_challenge_get_challenges

Overview

This function is used to retrieve all the challenges that have been created for the game on the currently active track (or the one specified in options).

You specify an optional callback method which is called when a response arrives from the server. As explained on the **Callbacks** page, if you don't specify this argument, an Async Social event will be triggered when a response is received. You can also supply an optional struct as the second argument, containing parameters for pagination (returning only a specific amount of challenges).

The function will always return a request ID, which can be used to identify the request in an Async Social event if a callback method is not specified.

Options

You can optionally specify a struct as the second argument, containing options for pagination. You can include the following variables in that struct:

Variable	Type	Value
page	integer	The page number to return
pageSize	integer	The maximum number of challenges to return
trackId	string	The track ID to use (if not specified, defaults to the currently active track)

See the example given at the bottom of this page for more information.

Callback Result

As explained on the **Callbacks & Async Events** page, the callback method will get a `_result` argument containing a `data` struct. The `data` struct for this function's response will contain the following variables:

Variable	Type	Value
challenges	array	An array containing structs for the challenges returned

pagination	struct	A struct containing information on pagination

The `_result.data.challenges` array will contain a struct for each challenge returned, where each challenge struct will contain the following variables:

Variable	Type	Value
challengeId	string	The ID of the challenge
coverArt	string	A link to the cover art for the challenge
creationDate	string	The date and time when the challenge was created
type	string	The type of the challenge (e.g., "duration")
criteria	string	The winning criteria for the challenge (e.g., "lowest_wins")
startsAt	string	The date and time when the challenge starts (if timed)
endsAt	string	The date and time when the challenge ends (if timed)
hasEnded	boolean	Has the challenge ended? (if timed)
hasStarted	boolean	Has the challenge started? (if timed)
isPublished	boolean	Is the challenge published?
isTimedChallenge	boolean	Is the challenge timed?
shortDescription	string	The short description of the challenge
longDescription	string	The long description of the challenge
name	string	The name of the challenge
players	integer	The amount of players who have submitted scores for this challenge

The `_result.data.pagination` struct will contain the following variables:

Variable	Type	Value
currPage	integer	The page number that was loaded
numPerPage	integer	The number of challenges that are being loaded per page
totalItems	integer	The total number of challenges for the game

totalPages	integer	The total number of pages that are available for the given page size

If pagination options are specified, the response may include pagination-specific errors that occur due to invalid values; please see Response Error Codes for information on such errors.

Syntax:

```
gxc_challenge_get_challenges([callback, options])
```

Argument	Description
callback	(Optional) A callback method which is called when an HTTP response is received
options	(Optional) A struct containing options for pagination

Returns:

```
Real
```

Example:

```
var _options =
{
    page: 0,
    pageSize: 10
};

gxc_challenge_get_challenges(function(_status, _result)
{
    if (_status == 200)
    {
        var _challenge_count = array_length(_result.data.challenges);
        for (var i = 0; i < _challenge_count; i ++)
        {
            var _challenge = _result.data.challenges[i];
            show_debug_message("Challenge " + string(i) + " is " + _challenge.name);
        }
    }
}, _options);
```

The above code requests the first 10 challenges for the current game to be retrieved, and once they are successfully returned, loops through the array and prints the name of each challenge to the output log.

Profile Functions

The following functions are given for working with profiles:

- gxc_profile_get_info

gxc_profile_get_info

Overview

This function is used to retrieve information about the current user's profile. You can specify an optional callback method which is called when a response arrives from the server. As explained on the **Callbacks** page, if you don't specify this argument, an Async Social event will be triggered when a response is received.

> **NOTE**: The data retrieved using this function is more readily available in the URL parameters, which can be retrieved using `gxc_get_query_param()`. (Note that URL parameters only work when the game is being played on GXC and not locally.)

The function will always return a request ID, which can be used to identify the request in an Async Social event if a callback method is not specified.

Callback Arguments

The callback method will receive two arguments: `_httpStatus` and `_result`, where the former is the status code for the HTTP response and the latter is a struct containing all returned data. This `_result` struct (or the `async_load` map in the Async Social event if a callback is not specified) will contain the following keys:

Key	Type	Value
data	struct	A struct containing information about the current user
errors	array	An array containing the errors that occurred for this request

The `data` struct will contain the following keys:

Key	Type	Value
username	string	The username of the current user
userId	string	The ID of the current user
avatarUrl	string	A URL to the current user's avatar image (can be used with `sprite_add()` to be displayed in-game)

The `errors` array contains structs indicating errors with the request. Each error struct will have a `code` variable containing an error message, all of which are listed on this page.

Syntax:

```
gxc_profile_get_info([callback]);
```

Argument	Description
callback	(Optional) A callback method which is called when an HTTP response is received

Returns:

```
Real (HTTP Request ID)
```

Example:

```
gxc_profile_get_info( function(_status, _result)
{
    if (_status == 200)
    {
        username = _result.data.username;

        var _avatar_url = _result.data.avatarUrl;

        user_sprite = sprite_add(_avatar_url, 0, 0, 0, 0, 0);
    }
});
```

The above code sends a request for the user's profile data, and in the callback function retrieves the username and the avatar URL. It then uses `sprite_add()` to download the avatar image through the given URL and stores its new ID in a variable.

Note that the status code is being checked against `200` as that indicates that the request was successful.

NOTE: Due to CORS, you will not be able to load the avatar image when the game is running locally. For it to work you are required to have uploaded your game to GXC and run it from there.

Config Script

The library contains a script called `gxc_api_config`, which holds any configuration options that are provided for altering the behaviour of the library.

The following macros are currently available in this script:

Macro	Description
`GXC_SUBMIT_SCORE_V2`	This controls whether the library uses the latest version of the `gxc_challenge_submit_score()` function (`true`), or whether it uses the legacy version (`false`). You should only use the legacy version if your game expects a response that the function provided in an earlier version of the library and updating it has broken game functionality.

Callbacks & Async Events

Callback Method

Several functions in this library accept an optional **callback method** as an argument. If specified, it will be called when a response is received for the submitted request, and it will get two arguments: `_httpStatus` and `_result`, where `_httpStatus` is the status of the HTTP response and `_result` is a struct containing all information returned from the response.

```
gxc_profile_get_info( function(_status, _result)
{
    // 200 means the request was successful
    if (_status == 200) show_debug_message("The user's name is " +
_result.data.username);
});
```

The `_result` struct will contain the following variables:

- **data**: This will be a struct containing all the data received as part of the response to your request. Note that it will be equal to `pointer_null` if the request resulted in one or more errors.
- **errors**: This will be an array containing any errors that resulted in your request not being completed; for more information please see **Response Error Codes**.
- **success**: This will be `true` if your request was successful, and `false` if it failed.

Async Social event

You also have the option to use the **Async Social** event instead of a callback method. To do so, you would simply not specify a callback argument in the function (or specify `undefined`) and then listen for a response in an Async Social event instead. (Note that if you do specify a valid callback method, then the Async Social event will **not** be fired.)

The `_httpStatus` argument is available in the Async Social event as the `"httpStatus"` key inside the `async_load` map, and the contents of the `_result` struct (which you would get in the callback method) are instead available directly inside the `async_load` map (so `_result.data` becomes `async_load[? "data"]`). The `"success"` key will also be available in this map.

You can check the return value of the original request call against the `"id"` key received in the Async Social event to confirm that the response is intended for a specific request:

```
// Create event
profile_request = gxc_profile_get_info();

// Async - Social event
if (async_load[? "id"] == profile_request && async_load[? "success"])
{
    var _data = async_load[? "data"]
    show_debug_message("The user's name is " + _data.username);
}
```

It is recommended to use a callback method as the Async event approach is only provided for ease of use with Drag And Drop™.

Response Error Codes

You may get one or more errors as part of your response for your Async request, for example after calling `gxc_challenge_submit_score()` or another similar function. You will receive an array with any errors that occurred, provided as part of the "data" object; so for callback methods, it will be available as `_result.data.errors` (where `_result` is the second argument of your method) and for Async Social events, it will be available as `async_load[? "data"].errors`. For more information on callback methods and Async Social events, please see **Callbacks & Async Events**.

The `errors` array may contain one or more structs, each containing just one variable: `code`. This `code` variable will be equal to a `gxc_error_*` constant which indicate the errors that resulted in your request not being completed. The `code` variable in an error struct will be equal to any one of the following constants:

Constant	Description
Game/Challenge	
`gxc_error_game_not_found`	The given game ID was invalid
`gxc_error_challenge_not_found`	The given challenge ID was invalid
`gxc_error_track_not_found`	The given track ID was invalid
`gxc_error_challenge_not_active`	The challenge is not active (it either has not started yet, or is already over)
`gxc_error_challenge_not_published`	The challenge has not been published on the selected track
`gxc_error_invalid_hash`	The given hash is incorrect
`gxc_error_negative_score`	The provided score is negative, which is not allowed
`gxc_error_sign_in_required`	Signing into GXC is required for this request
Pagination	
`gxc_error_page_invalid`	The provided page number is invalid

`gxc_error_page_less_than_0`	The provided page number is less than 0
`gxc_error_page_size_invalid`	The provided page size is invalid
`gxc_error_page_size_less_than_1`	The provided page size is less than 1 (as an empty page is not possible)
`gxc_error_page_size_too_high`	The provided page size is too high

This means that the `errors` structure will be laid out as shown below, in case of two errors:

```
errors:
[
    {
        code: gxc_error_game_not_found
    },
    {
        code: gxc_error_track_not_found
    }
]
```

Library Errors

Functions in this library may sometimes throw errors if they come across something problematic, all of which are described below:

Error	Description
`required query params not found`	A function was trying to retrieve a query parameter (such as `"game"` or `"challenge"`) which was not found in the game's current URL
`param 'callback' must be of type method`	A value of the wrong type was passed into a function's `callback` argument, which only accepts methods; please look at the function's reference page for an example
`function is deprecated`	A function you are calling has been deprecated and you must call a newer function instead