GML API

gdk_init REQUIRED

Usage

```
gdk_init(scid)
```

Description

Must be called before any other GDK extension function, recommend using a Controller persistent object that is created or placed in the first room and this call is in the create event.

Parameters

type	name	description
string	**scid**	The Service Configuration ID.

Code Sample

```
gdk_init("00000000-0000-0000-0000-000060ddb039");
```

gdk_update REQUIRED

Usage

`gdk_update()`

Description

This should be called each frame while GDK extension is active, recommend using a Controller persistent object that is created or placed in the first room and this call is in the step event.

Code Sample

```
gdk_update();
```

gdk_quit REQUIRED

Usage

`gdk_quit()`

Description

This should be called on close down and no GDK extension functions should be called after it, recommend using a Controller persistent object that is created or placed in the first room and this call is in the destroy event.

Code Sample

```
gdk_quit();
```

xboxone_show_account_picker TODO

Usage

```
xboxone_show_account_picker(arg0, arg1)
```

Description

This function launches the system's account picker which will associate the selected user with the specified pad. The "mode" argument is either 0 or 1 – if 0 is specified no guest accounts can be selected, while 1 allows guest accounts. This function returns an asynchronous event ID value, and when the account picker closes an asynchronous dialog event will be triggered with details of the result of the operation.

Parameters

type	name	description
string	**param1**	This is an important parameter
pointer	**param1**	This is an optional paramenter

Triggers

[Async Dialog Event]

type	name	description
string	**param1**	This is an important parameter
pointer	**param1**	This is an optional paramenter

Returns

type	description
string	This is an important parameter

Code Sample

```
show_debug_message("Hello world!");
```

xboxone_get_activating_user

Usage

```
xboxone_get_activating_user()
```

Description

With this function you can retrieve the user ID pointer for the user that launched the game from the console. Note that this is rarely what you want to do in a game, because this user can logout while the game is running.

Returns

type	description
pointer	The user ID pointer.

Code Sample

```
global.main_user = xboxone_get_activating_user();
```

xboxone_get_user_count

Usage

```
xboxone_get_user_count()
```

Description

With this function you can find the total number of users currently signed in to the system. The returned value will be an integer value.

Returns

type	description
real	The total number of users currently signed in to the system.

Code Sample

```
for (var i = 0; i < xboxone_get_user_count(); i++) {
    user_id[i] = xboxone_get_user(i);
}
```

xboxone_get_user

Usage

`xboxone_get_user(index)`

Description

Description With this function you can retrieve the user ID pointer for the indexed user. If the user does not exist, the function will return the constant pointer_null instead. You can find the number of users currently logged in with the function `xboxone_get_user_count()`.

Parameters

type	name	description
integer	**index**	The index to get the User ID from.

Returns

type	description
pointer	The user ID pointer.

Code Sample

```
for (var i = 0; i < xboxone_get_user_count(); i++) {
    user_id[i] = xboxone_get_user(i);
}
```

xboxone_stats_set_stat_real

Usage

```
xboxone_stats_set_stat_real(user_id, stat_name, stat_value)
```

Description

This function can be used to set the value of a stat for the given user ID. You supply the user ID as returned by the function `xboxone_get_user()`, then the stat string to set (if the stat string does not already exist then a new stat will be created and set to the given value) and a value (a real) to set the stat to. Note that the stat name must be alphanumeric only, with no symbols or spaces.

?> **Tip** When setting the stat value, any previous value will be overridden, therefore it is up to you to determine if the stat value should be updated or not (ie. check that the high score is actually the highest) by comparing to the current stat value with the new one before setting it.

Parameters

type	name	description
pointer	**user_id**	The user ID pointer.
string	**stat_name**	The statistic to set.
real	**stat_value**	The value to set the stat to.

Returns

type	description
real	-1 on error, any other value otherwise.

Code Sample

```
xboxone_stats_set_stat_real(p_user_id, "TestReal", 123.45);
```

xboxone_stats_set_stat_int

Usage

```
xboxone_stats_set_stat_int(user_id, stat_name, stat_value)
```

Description

This function can be used to set the value of a stat for the given user ID. You supply the user ID as returned by the function `xboxone_get_user()`, then the stat string to set (if the stat string does not already exist then a new stat will be created and set to the given value) and a value (an integer) to set the stat to. Note that the stat name must be alphanumeric only, with no symbols or spaces.

?> **Tip** When setting the stat value, any previous value will be overridden, therefore it is up to you to determine if the stat value should be updated or not (ie. check that the high score is actually the highest) by comparing to the current stat value with the new one before setting it.

Parameters

type	name	description
pointer	**user_id**	The user ID pointer.
string	**stat_name**	The statistic to set.
integer	**stat_value**	The value to set the stat to.

Returns

type	description
real	-1 on error, any other value otherwise.

Code Sample

```
xboxone_stats_set_stat_int(p_user_id, "TestInt", 22);
```

xboxone_stats_set_stat_string

Usage

```
xboxone_stats_set_stat_string(user_id, stat_name, stat_value)
```

Description

This function can be used to set the value of a stat for the given user ID. You supply the user ID as returned by the function `xboxone_get_user()`, then the stat string to set (if the stat string does not already exist then a new stat will be created and set to the given value) and a value (a string) to set the stat to. Note that the stat name must be alphanumeric only, with no symbols or spaces.

?> **Tip** When setting the stat value, any previous value will be overridden, therefore it is up to you to determine if the stat value should be updated or not (ie. check that the high score is actually the highest) by comparing to the current stat value with the new one before setting it.

Parameters

type	name	description
pointer	**user_id**	The user ID pointer.
string	**stat_name**	The statistic to set.
string	**stat_value**	The value to set the stat to.

Returns

type	description
real	-1 on error, any other value otherwise.

Code Sample

```
xboxone_stats_set_stat_string(p_user_id, "TestString", "YoYo Games");
```

xboxone_stats_delete_stat

Usage

```
xboxone_stats_delete_stat(user_id, stat_name)
```

Description

This function can be used to delete a stat from the stat manager for the given user ID. You supply the user ID as returned by the function `xboxone_get_user()`, then the stat string to delete. This clears the stat value and removed it from the stat manager, meaning it will no longer be returned by the functions `xboxone_stats_get_stat_names()` or `xboxone_stats_get_stat()`.

Parameters

type	name	description
pointer	**user_id**	The user ID pointer.
string	**stat_name**	The stat to be deleted.

Returns

type	description
real	-1 on error, any other value otherwise.

Code Sample

```
for (var i = 0; i < xboxone_get_user_count(); i++) {
    user_id[i] = xboxone_get_user(i);
    xboxone_stats_delete_stat(user_id[i], "highScore");
}
```

xboxone_stats_get_stat

Usage

```
xboxone_stats_get_stat(user_id, stat_name)
```

Description

This function can be used to retrieve a single stat value from the stat manager for a given user. You supply the user ID as returned by the function `xboxone_get_user()`, and then the stat string as defined when you created it using the one of the `xboxone_stats_set_stat_*` functions. The return value can be either a string or a real (depending on the stat being checked) or the GML constant undefined if the stat does not exist or there has been an error.

Parameters

type	name	description
pointer	**user_id**	The user ID pointer.
string	**stat_name**	The statistic to set.

Returns

type	description
real/string/undefined	the value for the given state.

Code Sample

```
if (game_over == true) {
    if (xboxone_stats_get_stat(p_user_id, "PercentDone") < 100) {
        var _val = (global.LevelsFinished / global.LevelsTotal) * 100;
        xboxone_stats_set_stat_real(p_user_id, "PercentDone", _val);
    }
}
```

xboxone_stats_get_stat_names

Usage

```
xboxone_stats_get_stat_names(user_id)
```

Description

This function can be used to retrieve all the defined stats from the stat manager for a given user. You supply the user ID as returned by the function `xboxone_get_user()`, and the function will returns an array of strings containing the statistics for the user. If an error occurs or the user has no stats the array will still be returned but will be empty.

Parameters

type	name	description
pointer	**user_id**	The user ID pointer.

Returns

type	description
array	Array with all the stat names.

Code Sample

```
var _stat_str = xboxone_stats_get_stat_names(user_id);
for (var i = 0; i < array_length(_stat_str); i++;) {
    xboxone_stats_delete_stat(user_id, _stat_str[i]);
}
```

xboxone_stats_add_user

Usage

```
xboxone_stats_add_user(user_id)
```

Description

This function can be used to add a given user to the statistics manager. This must be done before using any of the other stats functions to automatically sync the game with the xbox live server and retrieve the latest values. You supply the user ID as returned by the function `xboxone_get_user()`, and the function will return -1 if there was an error or the user ID is invalid, or any other value if the function was successfully called.

Parameters

type	name	description
pointer	**user_id**	The user ID pointer.

Returns

type	description
real	-1 on error, any other value otherwise.

Triggers

[Asynchronous Social Event]

type	name	description
constant	**id**	Will hold the constant `achievement_stat_event`.
string	**event**	Will hold the string `"LocalUserAdded"`.
pointer	**userid**	The user ID associated with the request.
real	**error**	0 if successful, some other value on error.
string	**errorMessage**	A string with an error message. OPTIONAL

Code Sample

```
for(var i = 0; i < xboxone_get_user_count(); i++)
{
    user_id[i] = xboxone_get_user(i);
    xboxone_stats_add_user(user_id[i]);
}
```

xboxone_stats_remove_user

Usage

```
xboxone_stats_remove_user(user_id)
```

Description

This function can be used to remove (unregister) a given user from the statistics manager, performing a flush of the stat data to the live server. According to the XBox documentation the game does not have to remove the user from the stats manager, as the XBox OS will periodically flush the stats anyway.

To use the function, you supply the user ID as returned by the function `xboxone_get_user()`, and the function will will return -1 if there was an error or the user ID is invalid, or any other value if the function was successfully called.

!> **Important** Removing the user can return an error if the statistics that have been set on the user are invalid (such as the stat names containing non-alpha numeric characters).

?> **Tip** If you want to flush the stats data to the live server at any time without removing the user, you can use the function `xboxone_stats_flush_user()`.

Parameters

type	name	description
pointer	**user_id**	The user ID pointer.

Returns

type	description
real	-1 on error, any other value otherwise.

Triggers

[Asynchronous Social Event]

type	name	description

type	name	description
constant	**id**	Will hold the constant `achievement_stat_event`.
string	**event**	Will hold the string `"LocalUserRemoved"`.
pointer	**userid**	The user ID associated with the request.
real	**error**	0 if successful, some other value on error.
string	**errorMessage**	A string with an error message. OPTIONAL

Code Sample

```
for (var i = 0; i < array_length(user_id); i++) {
    xboxone_stats_remove_user(user_id[i]);
}
```

xboxone_stats_flush_user

Usage

```
xboxone_stats_flush_user(user_id, high_priority)
```

Description

This function can be used to flush the stats data for a given user from the statistics manager, to the live server, ensuring that the server is up to date with the current values. To use the function, you supply the user ID as returned by the function `xboxone_get_user()`, and then give a priority value (0 for low priority and 1 for high priority). The function will will return -1 if there was an error or the user ID is invalid, or any other value if the function was successfully called.

?> **Important** According to XBox documentation, developers should be careful not to call this too often as the XBox will rate-limit the requests, and the XBox OS will also automatically flush stats approximately every 5 minutes automatically anyway.

Parameters

type	name	description
pointer	**user_id**	The user ID pointer.
boolean	**high_priority**	Whether or not flush is high priority.

Returns

type	description
real	-1 on error, any other value otherwise.

Triggers

[Asynchronous Social Event]

type	name	description
constant	**id**	Will hold the constant `achievement_stat_event`.

type	name	description
string	**event**	Will hold the string `"StatisticUpdateComplete"`.
pointer	**userid**	The user ID associated with the request.
real	**error**	0 if successful, some other value on error.
string	**errorMessage**	A string with an error message. OPTIONAL

Code Sample

```
for (var i = 0; i < array_length(user_id); i++) {
    xboxone_stats_flush_user(user_id[i], 0);
}
```

xboxone_stats_get_leaderboard

Usage

```
xboxone_stats_get_leaderboard(user_id, stat, num_entries, start_rank, start_at_user, ascending)
```

Description

This function can be used to retrieve a global leaderboard of ranks for a given statistic. You supply the user ID (as returned by the function `xboxone_get_user`), the stat string (as defined when you registered it as a "Featured Stat"), and then you specify a number of details about what leaderboard information you want to retrieve. Note that you can only retrieve a global leaderboard for int or real stats, but not for string stats.

!> **Important** Stats used in global leaderboards must be registered as "Featured Stats" in the XDP/Windows Dev Center otherwise an error will be returned. If you want local (social) leaderboards, then please see the function `xboxone_stats_get_social_leaderboard`.

Parameters

type	name	description
pointer	**user_id**	The user ID pointer.
string	**stat**	The stat to create the global leaderboard from.
real	**num_entries**	The number of entries from leaderboard to retrieve.
real	**start_rank**	The rank in the leaderboard to start from (0 if `start_at_user`).
boolean	**start_at_user**	Set to true to start at the user ID rank.
boolean	**ascending**	Set to true for ascending order and false for descending order.

Triggers

[Asynchronous Social Event]

type	name	description
constant	**id**	Will hold the constant `achievement_stat_event`.
string	**event**	Will hold the string `"LocalUserAdded"`.

type	name	description
pointer	**userid**	The user ID associated with the request.
real	**error**	0 if successful, some other value on error.
string	**errorMessage**	A string with an error message. OPTIONAL
string	**displayname**	The unique ID for the leaderboard as defined on the provider dashboard.
real	**numentries**	The number of entries in the leaderboard that you have received.
string	**PlayerN**	The name of the player, where "N" is position within the received entries list.
pointer	**PlayeridN**	The unique user id of the player, "N"
real	**RankN**	The rank of the player "N" within the leaderboard.
real	**ScoreN**	The score of the player "N".

Code Sample

The following is an extended example of how this function can be used. To start with you'd call it in some event like Room Start or Create:

```
xboxone_stats_get_leaderboard(user_id, "GlobalTime", 20, 1, false, true);
```

The above code would be called to get a list of all global leaderboard positions for the game, and will generate a Social Asynchronous Event call back which we would deal with in the following way:

```
if (async_load[? "id"] == achievement_stat_event) {
    if (async_load[? "event"] == "GetLeaderboardComplete") {
        global.numentries = async_load[? "numentries"];
        for (var i = 0; i < numentries; i++) {
            global.playername[i] = async_load[? "Player" + string(i)];
            global.playerid[i] = async_load[? "Playerid" + string(i)];
            global.playerrank[i] = async_load[? "Rank" + string(i)];
            global.playerscore[i] = async_load[? "Score" + string(i)];
        }
    }
}
```

The above code checks the returned ds_map in the Social Asynchronous Event and if its "id" matches the constant being checked, it then checks to see if the event has been triggered by returned leaderboard data before looping through the map and storing all the different values in a number of global arrays.

xboxone_stats_get_social_leaderboard

Usage

```
xboxone_stats_get_social_leaderboard(user_id, stat, num_entries, start_rank,
start_at_user, ascending, favourites_only)
```

Description

This function can be used to retrieve a social leaderboard of ranks for a given statistic. You supply the user ID (as returned by the function `xboxone_get_user`), the stat string (as defined when you created it using the `xboxone_stats_set_stat_*` functions), and then you specify a number of details about what leaderboard information you want to retrieve. Note that you can only retrieve a social leaderboard for int or real stats, but not for string stats, and that if you flag the "favourites_only" argument as true, then the results will only contain data for those friends that are marked by the user as "favourites".

?> **Tip** Stats used in social leaderboards do not need to be registered as "Featured Stats" in the XDP/Windows Dev Center.

Parameters

type	name	description
pointer	**user_id**	The user ID pointer.
string	**stat**	The stat to create the global leaderboard from.
real	**num_entries**	The number of entries from leaderboard to retrieve.
real	**start_rank**	The rank in the leaderboard to start from (0 if `start_at_user`).
boolean	**start_at_user**	Set to true to start at the user ID rank.
boolean	**ascending**	Set to true for ascending order and false for descending order.
boolean	**favourites_only**	Set to true to show only friends that are marked as "favourites".

Triggers

[Asynchronous Social Event]

type	name	description

type	name	description
constant	**id**	Will hold the constant `achievement_stat_event`.
string	**event**	Will hold the string `"LocalUserAdded"`.
pointer	**userid**	The user ID associated with the request.
real	**error**	0 if successful, some other value on error.
string	**errorMessage**	A string with an error message. OPTIONAL
string	**displayname**	The unique ID for the leaderboard as defined on the provider dashboard.
real	**numentries**	The number of entries in the leaderboard that you have received.
string	**PlayerN**	The name of the player, where "N" is position within the received entries list.
pointer	**PlayeridN**	The unique user id of the player, "N"
real	**RankN**	The rank of the player "N" within the leaderboard.
real	**ScoreN**	The score of the player "N".

Code Sample

The following is an extended example of how this function can be used. To start with you'd call it in some event like Room Start or Create:

```
xboxone_stats_get_social_leaderboard(user_id, "GlobalTime", 20, 1, false, true);
```

The above code would be called to get a list of all global leaderboard positions for the game, and will generate a Social Asynchronous Event call back which we would deal with in the following way:

```
if (async_load[? "id"] == achievement_stat_event) {
    if (async_load[? "event"] == "GetLeaderboardComplete") {
        global.numentries = async_load[? "numentries"];
        for (var i = 0; i < numentries; i++) {
            global.playername[i] = async_load[? "Player" + string(i)];
            global.playerid[i] = async_load[? "Playerid" + string(i)];
            global.playerrank[i] = async_load[? "Rank" + string(i)];
            global.playerscore[i] = async_load[? "Score" + string(i)];
        }
    }
}
```

The above code checks the returned ds_map in the Social Asynchronous Event and if its "id" matches the constant being checked, it then checks to see if the event has been triggered by returned leaderboard data before looping through the map and storing all the different values in a number of global arrays.