GOG Extension

Cross-platform services that make it easier and faster for developers to successfully launch, operate, and scale high-quality games.

See the official page for more documentation

Setup

Follow these guides to get yourself going on everything you need for your new game.

- Setup

Management

These functions provide general management features:

- GOG_GetError
- GOG_ProcessData REQUIRED

Modules

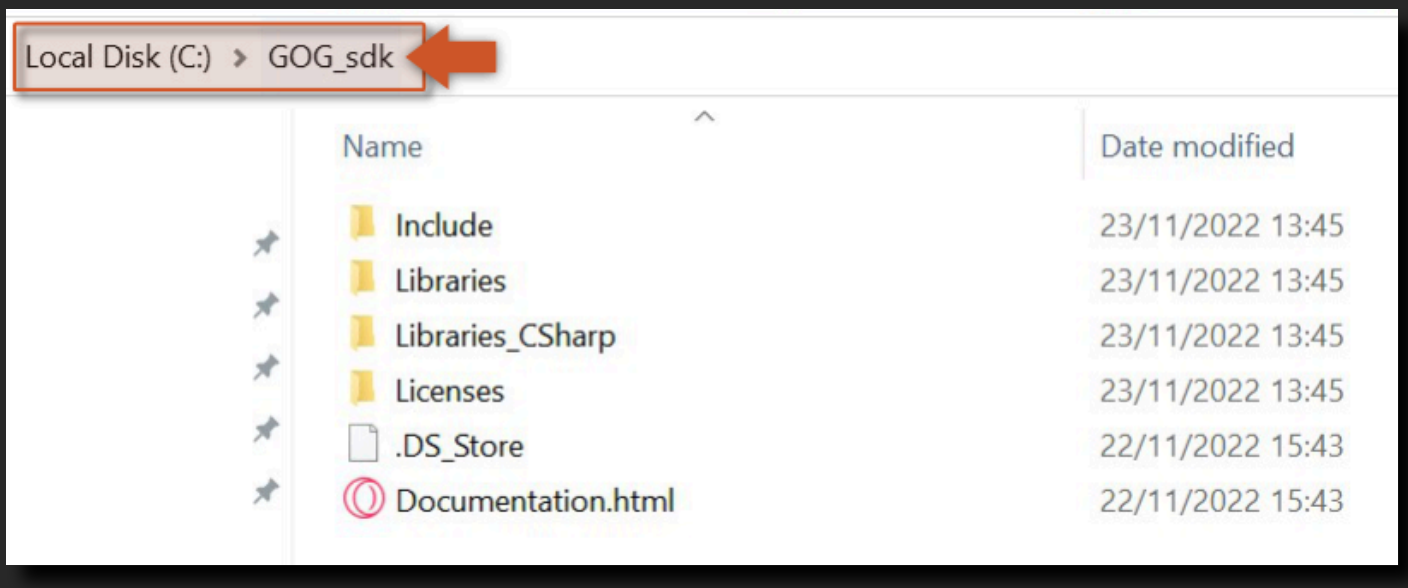
This extension API presents a variety of modules that can be used to push your game to the next level. These are the included modules:

- Apps
- Friends
- Stats
- Storage
- Telemetry
- User
- Utils

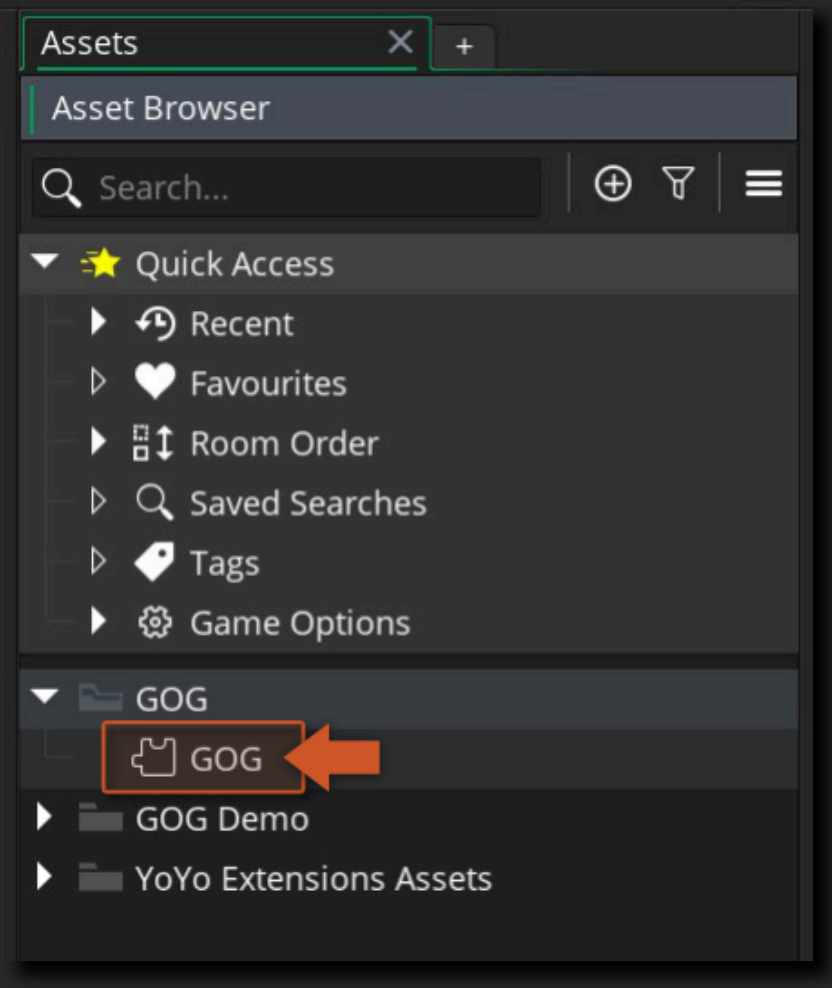
Setup Guide

To use the GOG API extension you should follow these steps:

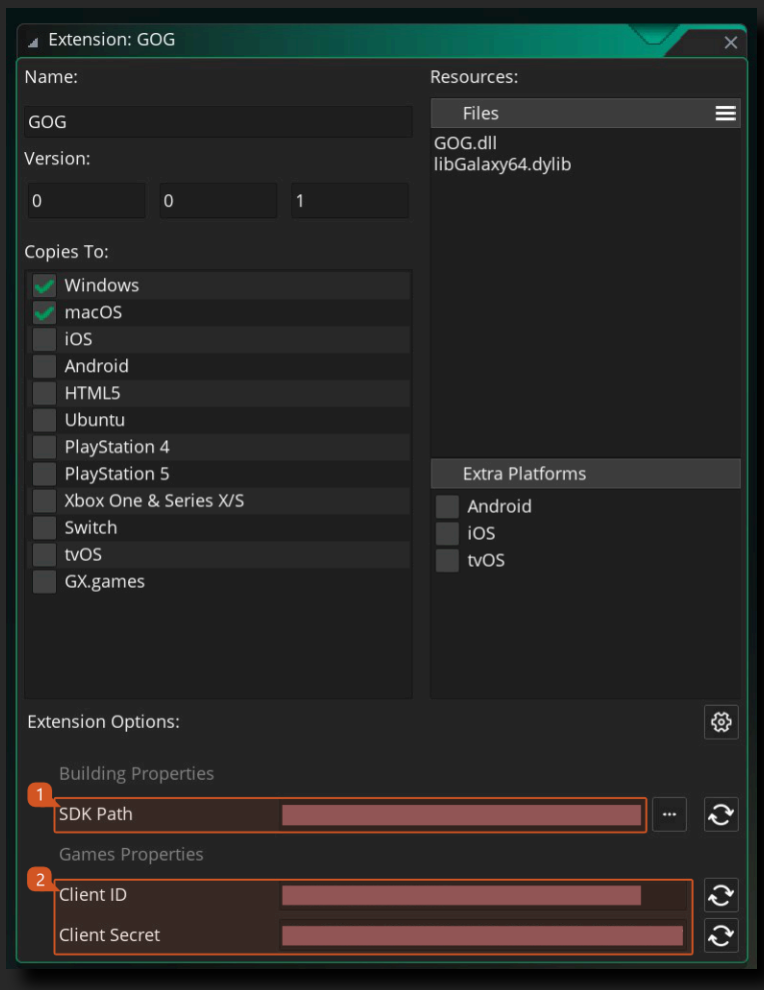
1. Import this GOG API extension into your project, if you haven't done that already.
2. The GOG Launcher needs to be **installed**, **running** and with an account **logged in** (**official site**).
3. Download GOG SDK (v1.150) from GOG's **developer portal** and extract the contents of the zip into a directory of your choice (e.g.: `C:\GOG_sdk`).



4. To set up your game properties, double click on the GOG extension in your Asset Browser in the IDE.



5. At the bottom of the extension widow you will find all the configurable options of the GOG extension.



6. The options are split in two sections **Building Properties** and **Game Properties**. The first one is a folder path that should point to the extracted folder of step 3, the second section will allow you to configure all the settings that are required for running and publishing a game to GOG Store.

GOG_GetError

Retrieves error connected with the last API call on the local thread.

NOTE If there was no error the return struct will be empty.

Syntax:

```
GOG_GetError();
```

Returns:

```
struct
```

Struct Member	Type	Description
Type	real	The type of the error.
Msg	string	The error message.
Name	string	The name of the error.

Example:

```
var error = GOG_GetError();
if (variable_struct_names_count(error) != 0)
{
    var type = error.Type;
    var message = error.Msg;
    var name = error.Name;
}
```

The code above provides a simple usage example.

GOG_ProcessData

This should be called each frame while GOG extension is active.

It's recommended using a Controller, persistent object that is created or placed in the first room and this call is in the Step Event.

Syntax:

```
GOG_ProcessData();
```

Returns:

```
undefined
```

Example:

```
GOG_ProcessData();
```

The code above provides a simple usage example.

Apps

This a module that manages application activities.

Functions

The following functions are provided to interact with this module:

- GOG_Apps_GetCurrentGameLanguage
- GOG_Apps_IsDlcInstalled

GOG_Apps_GetCurrentGameLanguage

Returns current game language for given product ID.

Syntax:

```
GOG_Apps_GetCurrentGameLanguage(productID)
```

Argument	Type	Description
productID	int64	The ID of the game or DLC to check.

Returns:

```
string
```

Example:

```
var language = GOG_Apps_GetCurrentGameLanguage(productID)
```

The code sample above will return the current game language for the provided product identifier.

GOG_Apps_IsDlcInstalled

Checks if specified DLC is installed.

Syntax:

```
GOG_Apps_IsDlcInstalled(productID)
```

Argument	Type	Description
productID	int64	The ID of the DLC to check.

Returns:

```
bool
```

Example:

```
if(GOG_Apps_IsDlcInstalled(productID))
{
   //Do something with this DLC
}
```

The code sample above will check if a given DLC is installed, the developer can then decide what should be done if it is.

Friends

This a module that manages social info and activities.

Functions

The following functions are provided to interact with this module:

- GOG_Friends_ClearRichPresence
- GOG_Friends_DeleteFriend
- GOG_Friends_DeleteRichPresence
- GOG_Friends_FindUser
- GOG_Friends_GetDefaultAvatarCriteria
- GOG_Friends_GetFriendAvatarImageID
- GOG_Friends_GetFriendAvatarImageRGBA
- GOG_Friends_GetFriendAvatarUrl
- GOG_Friends_GetFriendByIndex
- GOG_Friends_GetFriendCount
- GOG_Friends_GetFriendInvitationByIndex
- GOG_Friends_GetFriendInvitationCount
- GOG_Friends_GetFriendPersonaName
- GOG_Friends_GetFriendPersonaState
- GOG_Friends_GetPersonaName
- GOG_Friends_GetPersonaState
- GOG_Friends_GetRichPresence
- GOG_Friends_GetRichPresenceByIndex
- GOG_Friends_GetRichPresenceCount

- GOG_Friends_GetRichPresenceKeyByIndex
- GOG_Friends_IsFriend
- GOG_Friends_IsFriendAvatarImageRGBAAvailable
- GOG_Friends_IsUserInformationAvailable
- GOG_Friends_IsUserInTheSameGame
- GOG_Friends_RequestFriendInvitationList
- GOG_Friends_RequestFriendList
- GOG_Friends_RequestRichPresence
- GOG_Friends_RequestSentFriendInvitationList
- GOG_Friends_RequestUserInformation
- GOG_Friends_RespondToFriendInvitation
- GOG_Friends_SendFriendInvitation
- GOG_Friends_SendInvitation
- GOG_Friends_SetDefaultAvatarCriteria
- GOG_Friends_SetRichPresence
- GOG_Friends_ShowOverlayInviteDialog

Constants

These are the constants used by this module:

- AvatarType
- PersonaState

GOG_Friends_ClearRichPresence

Removes all rich presence data for the user.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_ClearRichPresence()
```

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description
type	string	`"GOG_Friends_ClearRichPresence"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Friends_ClearRichPresence()
```

The code sample above starts a clear rich presence task which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Friends_ClearRichPresence")
{
    if (ds_map_exists(async_load,"error"))
    {
```

```
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("ClearRichPresence SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_DeleteFriend

Removes a user from the friend list.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_DeleteFriend(userID)
```

Argument	Type	Description
userID	GalaxyID	The GalaxyID of the user to be removed from the friend list.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Friends_DeleteFriend"`
error	string	The error message; only if request failed OPTIONAL
userID	struct	GOG Galaxy user identifier

Example:

```
GOG_Friends_DeleteFriend(myFriendID)
```

The code sample above starts a friend deletion task which result can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Friends_DeleteFriend")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }
    var deletedFriend = async_load[?"userID"]
    show_debug_message("Friend Deleted")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_DeleteRichPresence

Removes the variable value under a specified name. If the variable doesn't exist method call has no effect.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_DeleteRichPresence(key)
```

Argument	Type	Description
key	string	The name of the variable to be removed.

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Friends_DeleteRichPresence"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Friends_DeleteRichPresence("playing")
```

The code sample above starts a rich presence delete task which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Friends_DeleteRichPresence")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
```

```
    }

    show_debug_message("DeleteRichPresence SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_FindUser

Finds a specified user.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_FindUser(userSpecifier)
```

Argument	Type	Description
userSpecifier	string	The specifier of the user.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Friends_FindUser"`
error	string	The error message; only if request failed OPTIONAL
userID	GalaxyID	The ID of the user.

Example:

```
GOG_Friends_FindUser(userSpecifier)
```

The code sample above starts a find user task which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Friends_FindUser")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }
    var user = async_load[?"userID"]
    show_debug_message("FindUser SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_GetDefaultAvatarCriteria

Returns the default avatar criteria which is a real with the bit sum of default AvatarType.

Syntax:

```
GOG_Friends_GetDefaultAvatarCriteria()
```

Returns:

```
real
```

Example:

```
var AvatarCriteria = GOG_Friends_GetDefaultAvatarCriteria()
```

The code above provides a simple usage example.

GOG_Friends_GetFriendAvatarImageID

Returns the ID of the avatar of a specified user.

REQUIREMENT Retrieve the avatar image first by calling GOG_Friends_RequestUserInformation with appropriate avatar criteria.

Syntax:

```
GOG_Friends_GetFriendAvatarImageID(userID, avatarType)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.
avatarType	AvatarType	The type of avatar.

Returns:

```
real
```

Example:

```
var ImageID = GOG_Friends_GetFriendAvatarImageID(userID, GOG_AVATAR_TYPE_SMALL)
```

The code above provides a simple usage example.

GOG_Friends_GetFriendAvatarImageRGBA

Copies the avatar of a specified user.

REQUIREMENT You might need to retrieve the data first by calling GOG_Friends_RequestUserInformation.

NOTE The size of the output buffer will be 4 * height * width (check AvatarType, for width and height values).

WARNING This function creates a new buffer everytime it is called you need to ensure you correctly delete the buffer when you don't need it anymore using the buffer_delete function. Failing to do so will result in memory leaks.

Syntax:

```
GOG_Friends_GetFriendAvatarImageRGBA(userID, avatarType)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.
AvatarType	AvatarType	The type of avatar.

Returns:

```
id.buffer
```

Example:

```
if(GOG_Friends_IsFriendAvatarImageRGBAAvailable(userID,GOG_AVATAR_TYPE_LARGE))
{
    var buff = GOG_Friends_GetFriendAvatarImageRGBA(userID,GOG_AVATAR_TYPE_LARGE)

    var size = buffer_get_size(buff)
    var L = sqrt(size/4)

    surf = surface_create(L,L)
    buffer_set_surface(buff,surf,0)
```

```
    buffer_delete(buff)
}
```

The code above provides a simple usage example.

GOG_Friends_GetFriendAvatarUrl

Returns the URL of the avatar of a specified user.

REQUIREMENT You might need to retrieve the data first by calling GOG_Friends_RequestUserInformation.

Syntax:

```
GOG_Friends_GetFriendAvatarUrl(userID, AvatarType)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.
avatarType	AvatarType	The type of avatar.

Returns:

```
string
```

Example:

```
var url = GOG_Friends_GetFriendAvatarUrl(userID, GOG_AVATAR_TYPE_SMALL)
```

The code above provides a simple usage example.

GOG_Friends_GetFriendByIndex

Returns the GalaxyID for a friend.

REQUIREMENT Retrieve the list of friends first by calling GOG_Friends_RequestFriendList.

Syntax:

```
GOG_Friends_GetFriendByIndex(index)
```

Argument	Type	Description
index	real	Index as an integer in the range of [0, number of friends).

Returns:

```
struct (GalaxyID)
```

Example:

```
for(var a = 0 ; a < GOG_Friends_GetFriendCount() ; a++)
{
    var friendID = GOG_Friends_GetFriendByIndex(a),GOG_AVATAR_TYPE_LARGE)
}
```

The code above provides a simple usage example.

GOG_Friends_GetFriendCount

Returns the number of retrieved friends in the user's list of friends.

REQUIREMENT Retrieve the list of friends first by calling GOG_Friends_RequestFriendList.

Syntax:

```
GOG_Friends_GetFriendCount()
```

Returns:

```
real
```

Example:

```
for(var a = 0 ; a < GOG_Friends_GetFriendCount() ; a++)
{
    var friendID = GOG_Friends_GetFriendByIndex(a),GOG_AVATAR_TYPE_LARGE)
}
```

The code above provides a simple usage example.

GOG_Friends_GetFriendInvitationByIndex

Reads the details of the friend invitation.

Syntax:

```
GOG_Friends_GetFriendInvitationByIndex(index)
```

Argument	Type	Description
index	real	Index as an integer in the range of [0, number of friend invitations).

Returns:

```
struct
```

Struct Member	Type	Description
userID	GalaxyID	The ID of the user who sent the invitation.
sendTime	real	The time at which the friend invitation was sent.

Example:

```
for(var i = 0 ; i < GOG_Friends_GetFriendInvitationCount() ; i++)
{
    var struct = GOG_Friends_GetFriendInvitationByIndex(i)
    var userID = struct.userID
    var sendTime = struct.sendTime
}
```

The code above provides a simple usage example.

GOG_Friends_GetFriendInvitationCount

Returns the number of retrieved friend invitations.

Syntax:

```
GOG_Friends_GetFriendInvitationCount()
```

Returns:

```
real
```

Example:

```
for(var i = 0 ; i < GOG_Friends_GetFriendInvitationCount() ; i++)
{
    var struct = GOG_Friends_GetFriendInvitationByIndex(i)
    var userID = struct.userID
    var sendTime = struct.sendTime
}
```

The code above provides a simple usage example.

GOG_Friends_GetFriendPersonaName

Returns the nickname of a specified user.

REQUIREMENT You might need to retrieve the data first by calling GOG_Friends_RequestUserInformation.

Syntax:

```
GOG_Friends_GetFriendPersonaName(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.

Returns:

```
string
```

Example:

```
var name = GOG_Friends_GetFriendPersonaName(GOG_User_GetGalaxyID())
```

The code above provides a simple usage example.

GOG_Friends_GetFriendPersonaState

Returns the state of a specified user, see PersonaState.

REQUIREMENT You might need to retrieve the data first by calling GOG_Friends_RequestUserInformation.

Syntax:

```
GOG_Friends_GetFriendPersonaState(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.

Returns:

```
real (PersonaState)
```

Example:

```
if (GOG_Friends_GetFriendPersonaState(myFriendID) == GOG_PERSONA_STATE_ONLINE)
{
    //My friend is online, do something
}
```

The code above provides a simple usage example.

GOG_Friends_GetPersonaName

Returns the user's nickname.

Syntax:

```
GOG_Friends_GetPersonaName()
```

Returns:

```
string
```

Example:

```
var name = GOG_Friends_GetPersonaName()
```

The code above provides a simple usage example.

GOG_Friends_GetPersonaState

Returns the user's state.

Syntax:

```
GOG_Friends_GetPersonaState (userID)
```

Returns:

```
real (PersonaState)
```

Example:

```
if(GOG_Friends_GetPersonaState() == GOG_PERSONA_STATE_ONLINE)
{
    // I'm online right now
}
```

The code above provides a simple usage example.

GOG_Friends_GetRichPresence

Returns the rich presence of a specified user.

REQUIREMENT Retrieve the rich presence first by calling GOG_Friends_RequestRichPresence.

Syntax:

```
GOG_Friends_GetRichPresence(key, userID)
```

Argument	Type	Description
key	string	The name of the property of the user's rich presence.
userID	GalaxyID	The ID of the user.

Returns:

```
string
```

Example:

```
var value = GOG_Friends_GetRichPresence("playing",GOG_User_GetGalaxyID())
```

The code above provides a simple usage example.

GOG_Friends_GetRichPresenceByIndex

Returns a property from the rich presence storage by index.

REQUIREMENT Retrieve the rich presence first by calling GOG_Friends_RequestRichPresence.

Syntax:

```
GOG_Friends_GetRichPresenceByIndex(index, userID)
```

Argument	Type	Description
index	real	Index as an integer in the range of [0, number of entries).
userID	GalaxyID	The ID of the user.

Returns:

```
struct
```

Struct Member	Type	Description
key	string	The name of the property of the user's rich presence.
value	string	The value of the property.

Example:

```
for(var i = 0 ; i < GOG_Friends_GetRichPresenceCount(GOG_User_GetGalaxyID()) ; i++)
{
    var struct = GOG_Friends_GetRichPresenceByIndex(i,GOG_User_GetGalaxyID())
    var key = struct.key
    var value = struct.value
}
```

The code above provides a simple usage example.

GOG_Friends_GetRichPresenceCount

Returns the number of retrieved properties in user's rich presence.

REQUIREMENT Retrieve the rich presence first by calling GOG_Friends_RequestRichPresence.

Syntax:

```
GOG_Friends_GetRichPresenceCount(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.

Returns:

```
real
```

Example:

```
for(var i = 0 ; i < GOG_Friends_GetRichPresenceCount(GOG_User_GetGalaxyID()) ; i++)
{
    var struct = GOG_Friends_GetRichPresenceByIndex(i,GOG_User_GetGalaxyID())
    var key = struct.key
    var value = struct.value
}
```

The code above provides a simple usage example.

GOG_Friends_GetRichPresenceKeyByIndex

Returns a key from the rich presence storage by index.

REQUIREMENT Retrieve the rich presence first by calling GOG_Friends_RequestRichPresence.

Syntax:

```
GOG_Friends_GetRichPresenceKeyByIndex(index, userID)
```

Argument	Type	Description
index	real	Index as an integer in the range of [0, number of entries).
userID	GalaxyID	The ID of the user.

Returns:

```
string
```

Example:

```
for (var i = 0; i < GOG_Friends_GetRichPresenceCount(GOG_User_GetGalaxyID()); i++)
{
    var key = GOG_Friends_GetRichPresenceKeyByIndex(i,GOG_User_GetGalaxyID())
}
```

The code above provides a simple usage example.

GOG_Friends_IsFriend

Checks if a specified user is a friend.

REQUIREMENT Retrieve the list of friends first by calling GOG_Friends_RequestFriendList.

Syntax:

```
GOG_Friends_IsFriend(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.

Returns:

```
bool
```

Example:

```
if (GOG_Friends_IsFriend(friendID))
{
    //Is my friend, do something
}
```

The code above provides a simple usage example.

GOG_Friends_IsFriendAvatarImageRGBAAvailable

Checks if a specified avatar image is available.

Syntax:

```
GOG_Friends_IsFriendAvatarImageRGBAAvailable(userID, avatarID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.
avatarID	AvatarType	The type of avatar.

Returns:

```
bool
```

Example:

```
if (GOG_Friends_IsFriendAvatarImageRGBAAvailable(userID,GOG_AVATAR_TYPE_LARGE))
{
    var buff = GOG_Friends_GetFriendAvatarImageRGBA(userID,GOG_AVATAR_TYPE_LARGE)

    var size = buffer_get_size(buff)
    var L = sqrt(size/4)

    surf = surface_create(L,L)
    buffer_set_surface(buff,surf,0)

    buffer_delete(buff)
}
```

The code above provides a simple usage example.

GOG_Friends_IsUserInformationAvailable

Checks if the information of specified user is available.

Syntax:

```
GOG_Friends_IsUserInformationAvailable(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.

Returns:

```
bool
```

Example:

```
if (GOG_Friends_IsUserInformationAvailable(GOG_User_GetGalaxyID()))
{
   var name = GOG_Friends_GetPersonaName(GOG_User_GetGalaxyID())
}
```

The code above provides a simple usage example.

GOG_Friends_IsUserInTheSameGame

Checks if a specified user is playing the same game.

REQUIREMENT Retrieve the rich presence first by calling GOG_Friends_RequestRichPresence.

Syntax:

```
GOG_Friends_IsUserInTheSameGame(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.

Returns:

```
bool
```

Example:

```
if(GOG_Friends_IsUserInformationAvailable(friendID))
{
   if(GOG_Friends_IsUserInTheSameGame(friendID))
   {
       //my friend is on same game, do something
   }
}
```

The code above provides a simple usage example.

GOG_Friends_RequestFriendInvitationList

Performs a request for the user's list of incoming friend invitations.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_RequestFriendInvitationList()
```

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Friends_RequestFriendInvitationList"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Friends_RequestFriendInvitationList()
```

The code sample above starts task for requesting friends invitation data which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Friends_RequestFriendInvitationList")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
```

```
        exit
    }

    show_debug_message("RequestFriendInvitationList SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_RequestFriendList

Performs a request for the user's list of friends.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_RequestFriendList()
```

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description
type	string	`"GOG_Friends_RequestFriendList"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Friends_RequestFriendList()
```

The code sample above starts task for requesting friends data which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Friends_RequestFriendList")
{
    if (ds_map_exists(async_load,"error"))
    {
```

```
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("RequestFriendList SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_RequestRichPresence

Performs a request for the user's rich presence.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_RequestRichPresence(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Friends_RequestRichPresence"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Friends_RequestRichPresence()
```

The code sample above starts task for requesting rich presence data which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Friends_RequestRichPresence")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("RequestRichPresence SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_RequestSentFriendInvitationList

Performs a request for the user's list of outgoing friend invitations.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_RequestSentFriendInvitationList()
```

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Friends_RequestSentFriendInvitationList"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Friends_RequestSentFriendInvitationList()
```

The code sample above starts task for requesting sent friendship invitation data which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Friends_RequestSentFriendInvitationList")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
```

```
        exit
    }

    show_debug_message("RequestSentFriendInvitationList SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_RequestUserInformation

Performs a request for information about specified user.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_RequestUserInformation(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Friends_RequestUserInformation"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Friends_RequestUserInformation()
```

The code sample above starts task for requesting user information data which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Friends_RequestUserInformation")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }
```

```
    show_debug_message("GOG_Friends_RequestUserInformation   SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_RespondToFriendInvitation

Responds to the friend invitation.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
Discord_Core_Create(userID, accept)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user who sent the friend invitation.
accept	bool	True when accepting the invitation, false when declining.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Friends_RespondToFriendInvitation"`
error	string	The error message; only if request failed OPTIONAL
userID	GalaxyID	The ID of the user.
accept	bool	true to accept invitation, false to refuse

Example:

```
GOG_Friends_RespondToFriendInvitation()
```

The code sample above starts task for responding to a pending invitation which results can be caught inside an **Async Social** event.

```
if (async_load[? "type"] == " GOG_Friends_RespondToFriendInvitation")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message(" RespondToFriendInvitation  SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_SendFriendInvitation

Sends a friend invitation.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_SendFriendInvitation(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Friends_SendFriendInvitation"`
error	string	The error message; only if request failed OPTIONAL
userID	GalaxyID	The ID of the user.

Example:

```
GOG_Friends_SendFriendInvitation (friendID)
```

The code sample above starts task for sending a friend invitation which results can be caught inside an Async Social event.

```
if(async_load[? "type"] == "GOG_Friends_SendFriendInvitation")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("SendFriendInvitationSUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_SendInvitation

Sends a game invitation without using the overlay.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_SendInvitation(userID, connectionString)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user.
connectionString	string	The string which contains connection info with the limit of 4095 bytes.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Friends_RespondToFriendInvitation"`
error	string	The error message; only if request failed OPTIONAL
userID	GalaxyID	The ID of the user.
connectionString	string	connectionString

Example:

```
GOG_Friends_SendInvitation(userID,connectionString)
```

The code sample above starts task for sending a play invitation which results can be caught inside an **Async Social** event.

```
if (async_load[? "type"] == " GOG_Friends_RespondToFriendInvitation")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("SendInvitation SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_SetDefaultAvatarCriteria

Sets the default avatar criteria.

Syntax:

```
GOG_Friends_SetDefaultAvatarCriteria(defaultAvatarCriteria)
```

Argument	Type	Description
defaultAvatarCriteria	real	The bit sum of default AvatarType.

Returns:

```
undefined
```

Example:

```
GOG_Friends_SetDefaultAvatarCriteria(GOG_AVATAR_TYPE_SMALL)
```

The code above provides a simple usage example.

GOG_Friends_SetRichPresence

Sets the variable value under a specified name.

There are three keys that can be used:

- `"status"` - The description visible in Galaxy Client with the limit of 3000 bytes.
- `"metadata"` - The metadata that describes the status to other instances of the game with the limit of 2048 bytes.
- `"connect"` - The string which contains connection info with the limit of 4095 bytes. It can be regarded as a passive version of GOG_Friends_SendInvitation because it allows friends that notice the rich presence to join a multiplayer game.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Friends_SetRichPresence(key, value)
```

Argument	Type	Description
key	string	The name of the property of the user's rich presence (see above).
value	string	The value of the property to set.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description

type	string	`"GOG_Friends_RespondToFriendInvitation"`
error	string	Only if request failed

Example:

```
GOG_Friends_RespondToFriendInvitation()
```

The code sample above starts task for setting the rich presence value which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == " GOG_Friends_SetRichPresence")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

   show_debug_message("SetRichPresence SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Friends_ShowOverlayInviteDialog

Shows game invitation dialog that allows to invite users to game.

NOTE For this call to work, the overlay needs to be initialized first. To check whether the overlay is initialized, call GOG_Utils_GetOverlayState.

Syntax:

```
GOG_Friends_ShowOverlayInviteDialog(connectionString)
```

Argument	Type	Description
connectionString	string	The string which contains connection info with the limit of 4095 bytes.

Returns:

```
undefined
```

Example:

```
GOG_Friends_ShowOverlayInviteDialog(connectionString)
```

The code above provides a simple usage example.

PersonaState

These constants represent state of a user, are used with the following function calls:

- GOG_Friends_GetFriendPersonaState

Persona State Constant	Description
`GOG_PERSONA_STATE_OFFLINE`	User is not currently logged on.
`GOG_PERSONA_STATE_ONLINE`	User is logged on.

AvatarType

These constants represent type of an user avatar, are used with the following function calls:

- GOG_Friends_GetFriendAvatarImageID

and are also used as a bit mask sum in the return value of GOG_Friends_GetDefaultAvatarCriteria.

Avatar Type Constant	Description
`GOG_AVATAR_TYPE_NONE`	No avatar type specified.
`GOG_AVATAR_TYPE_SMALL`	Avatar resolution size: 32x32.
`GOG_AVATAR_TYPE_MEDIUM`	Avatar resolution size: 64x64.
`GOG_AVATAR_TYPE_LARGE`	Avatar resolution size: 184x184.

Stats

This is a module for managing statistics, achievements and leaderboards.

Functions

The following functions are provided to interact with this module:

- GOG_Stats_ClearAchievement
- GOG_Stats_FindLeaderboard
- GOG_Stats_FindOrCreateLeaderboard
- GOG_Stats_GetAchievement
- GOG_Stats_GetAchievementDescription
- GOG_Stats_GetAchievementDisplayName
- GOG_Stats_GetLeaderboardDisplayName
- GOG_Stats_GetLeaderboardDisplayType
- GOG_Stats_GetLeaderboardEntryCount
- GOG_Stats_GetLeaderboardSortMethod
- GOG_Stats_GetRequestedLeaderboardEntry
- GOG_Stats_GetStatFloat
- GOG_Stats_GetStatInt
- GOG_Stats_GetUserTimePlayed
- GOG_Stats_IsAchievementVisible
- GOG_Stats_IsAchievementVisibleWhileLocked
- GOG_Stats_RequestLeaderboardEntriesAroundUser
- GOG_Stats_RequestLeaderboardEntriesGlobal
- GOG_Stats_RequestLeaderboards

- GOG_Stats_RequestUserStatsAndAchievements
- GOG_Stats_RequestUserTimePlayed
- GOG_Stats_ResetStatsAndAchievements
- GOG_Stats_SetAchievement
- GOG_Stats_SetLeaderboardScore
- GOG_Stats_SetLeaderboardScoreWithDetails
- GOG_Stats_SetStatFloat
- GOG_Stats_SetStatInt
- GOG_Stats_StoreStatsAndAchievements
- GOG_Stats_UpdateAvgRateStat

Constants

This module exposes the following constants:

- LeaderboardDisplayType
- LeaderboardSortMethod

GOG_Stats_ClearAchievement

Clears an achievement.

NOTE In order to make this and other changes persistent, call GOG_Stats_StoreStatsAndAchievements.

REQUIREMENT Retrieve the achievements first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_ClearAchievement(name)
```

Argument	Type	Description
name	string	The code name of the achievement.

Returns:

```
undefined
```

Example:

```
GOG_Stats_ClearAchievement("100kills")
```

The code above provides a simple usage example.

GOG_Stats_FindLeaderboard

Performs a request for definition of a specified leaderboard.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Stats_FindLeaderboard(name)
```

Argument	Type	Description
name	string	The name of the leaderboard.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Stats_FindLeaderboard"`
error	string	The error message; only if request failed OPTIONAL
name	string	Name of the leaderboard

Example:

```
GOG_Stats_FindLeaderboard("BestScoresLeaderboard")
```

The code sample above starts a task for finding a leaderboard which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Stats_FindLeaderboard")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }
    var name = async_load[?"name"]
    show_debug_message(" FindLeaderboard SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Stats_FindOrCreateLeaderboard

Performs a request for definition of a specified leaderboard, creating it if there is no such leaderboard yet.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Stats_FindOrCreateLeaderboard(name,displayName,sortMethod,displayType)
```

Argument	Type	Description
name	string	The name of the leaderboard.
displayName	string	The display name of the leaderboard.
sortMethod	LeaderboardSortMethod	The sort method of the leaderboard.
displayType	LeaderboardDisplayType	The display method of the leaderboard.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Stats_FindOrCreateLeaderboard"`
error	string	The error message; only if request failed OPTIONAL

name	string	Name of the leaderboard

Example:

```
GOG_Stats_FindOrCreateLeaderboard("BestScoresLeaderboard")
```

The code sample above starts a task for finding or creating a leaderboard which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Stats_FindOrCreateLeaderboard")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }
    var name = async_load[?"name"]
    show_debug_message(" FindOrCreateLeaderboard SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Stats_GetAchievement

Reads the state of an achievement of a specified user.

REQUIREMENT Retrieve the achievements first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_GetAchievement(name)
```

Argument	Type	Description
name	string	The code name of the achievement.

Returns:

```
struct
```

Struct Member	Type	Description
unlocked	bool	Indicates if the achievement has been unlocked.
unlockedTime	real	The time at which the achievement was unlocked.

Example:

```
var struct = GOG_Stats_GetAchievement("100kills")
var unlocked = struct.unlocked
if(unlocked)
{
    unlockTime = struct.unlockTime
    //Achievement unlocked, do something....
}
```

The code above provides a simple usage example.

GOG_Stats_GetAchievementDescription

Returns description of a specified achievement.

REQUIREMENT Retrieve the achievements first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_GetAchievementDescription(name)
```

Argument	Type	Description
name	string	The name of the achievement.

Returns:

```
string
```

Example:

```
var description = GOG_Stats_GetAchievementDescription("100kills")
```

The code above provides a simple usage example.

GOG_Stats_GetAchievementDisplayName

Returns display name of a specified achievement.

REQUIREMENT Retrieve the achievements first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_GetAchievementDisplayName(name)
```

Argument	Type	Description
name	string	The name of the achievement.

Returns:

```
string
```

Example:

```
var displayName = GOG_Stats_GetAchievementDisplayName("100kills")
```

The code above provides a simple usage example.

GOG_Stats_GetLeaderboardDisplayName

Returns display name of a specified leaderboard.

REQUIREMENT Retrieve definition of this particular leaderboard first by calling either GOG_Stats_FindLeaderboard or GOG_Stats_FindOrCreateLeaderboard, or definitions of all existing leaderboards by calling GOG_Stats_RequestLeaderboards.

Syntax:

```
GOG_Stats_GetLeaderboardDisplayName(name)
```

Argument	Type	Description
name	string	The name of the leaderboard.

Returns:

```
string
```

Example:

```
var displayName = GOG_Stats_GetLeaderboardDisplayName("100kills")
```

The code above provides a simple usage example.

GOG_Stats_GetLeaderboardDisplayType

Returns display type of a specified leaderboard.

REQUIREMENT Retrieve definition of this particular leaderboard first by calling either GOG_Stats_FindLeaderboard or GOG_Stats_FindOrCreateLeaderboard, or definitions of all existing leaderboards by calling GOG_Stats_RequestLeaderboards.

Syntax:

```
GOG_Stats_GetLeaderboardDisplayType(name)
```

Argument	Type	Description
name	string	The name of the leaderboard.

Returns:

```
real (LeaderboardDisplayType)
```

Example:

```
var displayType = GOG_Stats_GetLeaderboardDisplayType(name)
```

The code above provides a simple usage example.

GOG_Stats_GetLeaderboardEntryCount

Returns the leaderboard entry count for requested leaderboard.

REQUIREMENT In order to retrieve leaderboard entry count, first you need to call GOG_Stats_RequestLeaderboardEntriesGlobal.

Syntax:

```
GOG_Stats_GetLeaderboardEntryCount(name)
```

Argument	Type	Description
name	string	The name of the leaderboard.

Returns:

```
real
```

Example:

```
var count = GOG_Stats_GetLeaderboardEntryCount(name)
```

The code above provides a simple usage example.

GOG_Stats_GetLeaderboardSortMethod

Returns sort method of a specified leaderboard.

REQUIREMENT Retrieve definition of this particular leaderboard first by calling either GOG_Stats_FindLeaderboard or GOG_Stats_FindOrCreateLeaderboard, or definitions of all existing leaderboards by calling GOG_Stats_RequestLeaderboards.

Syntax:

```
GOG_Stats_GetLeaderboardSortMethod(name)
```

Argument	Type	Description
name	string	The name of the leaderboard.

Returns:

```
real (LeaderboardSortMethod)
```

Example:

```
GOG_Stats_GetLeaderboardSortMethod(name)
```

The code above provides a simple usage example.

GOG_Stats_GetRequestedLeaderboardEntry

Returns data from the currently processed request for leaderboard entries.

REQUIREMENT In order to retrieve leaderboards and get their count, first you need to call GOG_Stats_RequestLeaderboardEntriesGlobal..

Syntax:

```
GOG_Stats_GetRequestedLeaderboardEntry(index)
```

Argument	Type	Description
index	real	Index as an integer in the range of [0, number of entries fetched).

Returns:

```
struct
```

Struct Member	Type	Description
rank	bool	User's rank in the leaderboard.
score	real	User's score in the leaderboard.
userID	GalaxyID	The ID of the user.

Example:

```
var struct = GOG_Stats_GetRequestedLeaderboardEntry(index)
var rank = struct.rank
var score = struct.score
var userID = struct.userID
```

The code above provides a simple usage example.

GOG_Stats_GetStatFloat

Reads floating point value of a statistic of a specified user.

REQUIREMENT Retrieve the statistics first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_GetStatFloat(name, userID)
```

Argument	Type	Description
name	string	The code name of the statistic.
userID	GalaxyID	The ID of the user. It can be omitted when requesting for own data.

Returns:

```
real
```

Example:

```
var value = GOG_Stats_GetStatFloat(nameStat,GOG_User_GetGalaxyID())
```

The code above provides a simple usage example.

GOG_Stats_GetStatInt

Reads integer value of a statistic of a specified user.

REQUIREMENT Retrieve the statistics first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_GetStatInt(name, userID)
```

Argument	Type	Description
name	string	The code name of the statistic.
userID	GalaxyID	The ID of the user. It can be omitted when requesting for own data.

Returns:

```
real
```

Example:

```
var value = GOG_Stats_GetStatInt(name,GOG_User_GetGalaxyID())
```

The code above provides a simple usage example.

GOG_Stats_GetUserTimePlayed

Reads the number of seconds played by a specified user.

REQUIREMENT Retrieve the statistics first by calling GOG_Stats_RequestUserTimePlayed.

Syntax:

```
GOG_Stats_GetUserTimePlayed(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user. It can be omitted when requesting for own data.

Returns:

```
real
```

Example:

```
GOG_Stats_GetUserTimePlayed(GOG_User_GetGalaxyID())
```

The code above provides a simple usage example.

GOG_Stats_IsAchievementVisible

Returns visibility status of a specified achievement.

REQUIREMENT Retrieve the achievements first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_IsAchievementVisible(name)
```

Argument	Type	Description
name	string	The name of the achievement.

Returns:

```
bool
```

Example:

```
if(GOG_Stats_IsAchievementVisible(name))
{
    //Achievement is visible, do something...
}
```

The code above provides a simple usage example.

GOG_Stats_IsAchievementVisibleWhileLocked

Returns visibility status of a specified achievement before unlocking.

REQUIREMENT Retrieve the achievements first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_IsAchievementVisibleWhileLocked(name)
```

Argument	Type	Description
name	string	The name of the achievement.

Returns:

```
bool
```

Example:

```
if(GOG_Stats_IsAchievementVisibleWhileLocked(appId))
{
    //Achievement visible while locked
}
```

The code above provides a simple usage example.

GOG_Stats_RequestLeaderboardEntriesAroundUser

Performs a request for entries of a specified leaderboard for and near the specified user.

The specified numbers of entries before and after the specified user are treated as hints. If the requested range would go beyond the set of all leaderboard entries, it is shifted so that it fits in the set of all leaderboard entries and preserves its size if possible.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

REQUIREMENT Retrieve definition of this particular leaderboard first by calling either GOG_Stats_FindLeaderboard or GOG_Stats_FindOrCreateLeaderboard, or definitions of all existing leaderboards by calling GOG_Stats_RequestLeaderboards.

Syntax:

```
GOG_Stats_RequestLeaderboardEntriesAroundUser(name, countBefore, countAfter)
```

Argument	Type	Description
name	string	Name of the leaderboard
countBefore	real	The number of entries placed before the user's entry to retrieve (hint).
countAfter	real	The number of entries placed after the user's entry to retrieve (hint).

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description
type	string	`"GOG_Stats_RequestLeaderboardEntriesAroundUser"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Stats_RequestLeaderboardEntriesAroundUser("NearScores",1,10)
```

The code sample above starts a task that requests leadboard entries around user which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Stats_RequestLeaderboardEntriesAroundUser")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("DeleteRichPresence SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Stats_RequestLeaderboardEntriesGlobal

Performs a request for entries of a specified leaderboard in a global scope, i.e. without any specific users in the scope of interest.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

REQUIREMENT Retrieve definition of this particular leaderboard first by calling either GOG_Stats_FindLeaderboard or GOG_Stats_FindOrCreateLeaderboard, or definitions of all existing leaderboards by calling GOG_Stats_RequestLeaderboards.

Syntax:

```
GOG_Stats_RequestLeaderboardEntriesGlobal(name, rangeStart, rangeEnd)
```

Argument	Type	Description
name	string	Name of the leaderboard
rangeStart	real	The index position of the entry to start with.
rangeEnd	real	The index position of the entry to finish with.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Stats_RequestLeaderboardEntriesGlobal"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Stats_RequestLeaderboardEntriesGlobal("BestScores",1,10)
```

The code sample above starts a task that requests global leadboard entries which results can be caught inside an **Async Social** event.

```
if (async_load[? "type"] == "GOG_Stats_RequestLeaderboardEntriesGlobal")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("DeleteRichPresence SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Stats_RequestLeaderboards

Performs a request for definitions of leaderboards.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Stats_RequestLeaderboards()
```

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Stats_RequestLeaderboards"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Stats_RequestLeaderboards()
```

The code sample above starts a task that requests all leadboard definitions which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Stats_RequestLeaderboards")
{
    if (ds_map_exists(async_load,"error"))
    {
```

```
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("RequestLeaderboards SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Stats_RequestUserStatsAndAchievements

Performs a request for statistics and achievements of a specified user.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Stats_RequestUserStatsAndAchievements(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user. It can be omitted when requesting for own data.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Stats_RequestUserStatsAndAchievements"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Stats_RequestUserStatsAndAchievements(GOG_User_GetGalaxyID())
```

The code sample above starts a task that requests data for stats and achievements which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Stats_RequestUserStatsAndAchievements ")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("RequestUserStatsAndAchievements SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Stats_RequestUserTimePlayed

Performs a request for user time played.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Stats_RequestUserTimePlayed(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user. It can be omitted when requesting for own data.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Stats_RequestUserTimePlayed"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Stats_RequestUserTimePlayed (GOG_User_GetGalaxyID())
```

The code sample above starts a task that requests user's play time and which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Stats_RequestUserTimePlayed ")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }
    var userID = async_load[?"userID"]
    show_debug_message(" RequestUserTimePlayed SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Stats_ResetStatsAndAchievements

Resets all statistics and achievements.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

NOTE This is the same as setting statistics and achievements to their initial values and calling GOG_Stats_StoreStatsAndAchievements.

Syntax:

```
GOG_Stats_ResetStatsAndAchievements()
```

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Stats_ResetStatsAndAchievements "`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Stats_ResetStatsAndAchievements()
```

The code sample above starts a task for resetting stats and achievements and which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Stats_ResetStatsAndAchievements ")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }
    show_debug_message("ResetStatsAndAchievements SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Stats_SetAchievement

Unlocks an achievement.

NOTE In order to make this and other changes persistent, call GOG_Stats_StoreStatsAndAchievements.

REQUIREMENT Retrieve the achievements first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_SetAchievement(name)
```

Argument	Type	Description
name	string	The code name of the achievement.

Returns:

```
undefined
```

Example:

```
GOG_Stats_SetAchievement("100kills")
```

The code above provides a simple usage example.

GOG_Stats_SetLeaderboardScore

Updates entry for own user in a specified leaderboard.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

REQUIREMENT Retrieve definition of this particular leaderboard first by calling either GOG_Stats_FindLeaderboard or GOG_Stats_FindOrCreateLeaderboard, or definitions of all existing leaderboards by calling GOG_Stats_RequestLeaderboards.

NOTE For this call to work while the user is logged off, the definition of the leaderboard must have been retrieved at least once while the user was logged on.

Syntax:

```
GOG_Stats_SetLeaderboardScore(name, score, forceUpdate)
```

Argument	Type	Description
name	string	The name of the leaderboard.
score	real	The score to set.
forceUpdate	bool	If the update should be performed in case the score is worse than the previous score.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description

type	string	`"GOG_Stats_SetLeaderboardScore "`
error	string	The error message; only if request failed OPTIONAL
name	string	Name of the Leaderboard
score	real	Score submited

Example:

```
GOG_Stats_SetLeaderboardScore ("BestScores",992,false)
```

The code sample above starts a task for updating the score of a given leaderboard, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Stats_SetLeaderboardScore ")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }
    show_debug_message("SetLeaderboardScore SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Stats_SetLeaderboardScoreWithDetails

Updates entry with details for own user in a specified leaderboard.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

REQUIREMENT Retrieve definition of this particular leaderboard first by calling either GOG_Stats_FindLeaderboard or GOG_Stats_FindOrCreateLeaderboard, or definitions of all existing leaderboards by calling GOG_Stats_RequestLeaderboards.

NOTE For this call to work while the user is logged off, the definition of the leaderboard must have been retrieved at least once while the user was logged on.

Syntax:

```
GOG_Stats_SetLeaderboardScore(name, score, forceUpdate, details)
```

Argument	Type	Description
name	string	The name of the leaderboard.
score	real	The score to set.
forceUpdate	bool	If the update should be performed in case the score is worse than the previous score.
details	id.buffer	An extra game-defined information regarding how the user got that score with the limit of 3071 bytes.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description
type	string	`"GOG_Stats_SetLeaderboardScore "`
error	string	The error message; only if request failed OPTIONAL
name	string	Name of the Leaderboard
score	real	Score submited

Example:

```
GOG_Stats_SetLeaderboardScore ("BestScores",992,false)
```

The code sample above starts a task for updating the score (with extra details) of a given leaderboard, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Stats_SetLeaderboardScore ")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }
    show_debug_message("SetLeaderboardScore SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Stats_SetStatFloat

Updates a statistic with a floating point value.

NOTE In order to make this and other changes persistent, call GOG_Stats_StoreStatsAndAchievements.

REQUIREMENT Retrieve the statistics first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_SetStatFloat(name, value)
```

Argument	Type	Description
name	string	The code name of the statistic.
value	real	The value of the statistic to set.

Returns:

```
undefined
```

Example:

```
GOG_Stats_SetStatFloat("speed", 12.2)
```

The code above provides a simple usage example.

GOG_Stats_SetStatInt

Updates a statistic with an integer value.

NOTE In order to make this and other changes persistent, call GOG_Stats_StoreStatsAndAchievements.

REQUIREMENT Retrieve the statistics first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_SetStatInt(name, value)
```

Argument	Type	Description
name	string	The code name of the statistic.
value	real	The value of the statistic to set.

Returns:

```
undefined
```

Example:

```
GOG_Stats_SetStatInt("attack", 100)
```

The code above provides a simple usage example.

GOG_Stats_StoreStatsAndAchievements

Persists all changes in statistics and achievements.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Stats_StoreStatsAndAchievements()
```

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Stats_StoreStatsAndAchievements "`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_Stats_StoreStatsAndAchievements()
```

The code sample above starts a task to store stats and achievements data and which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Stats_StoreStatsAndAchievements ")
{
    if (ds_map_exists(async_load,"error"))
    {
```

```
        show_debug_message(async_load[?"error"])
        exit
    }
    var userID = async_load[?"userID"]
    show_debug_message(" StoreStatsAndAchievements SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Stats_UpdateAvgRateStat

Updates an average-rate statistic with a delta.

REQUIREMENT Retrieve the statistics first by calling GOG_Stats_RequestUserStatsAndAchievements.

Syntax:

```
GOG_Stats_UpdateAvgRateStat(name, countThisSession, sessionLenght)
```

Argument	Type	Description
name	string	The code name of the statistic.
countThisSession	real	The delta of the count.
sessionLenght	real	The delta of the session.

Returns:

```
undefined
```

Example:

```
GOG_Stats_UpdateAvgRateStat("coins", 100, 1000)
```

The code above provides a simple usage example.

LeaderboardDisplayType

These constants represent the display type of the leaderboard data, and are used with the following function calls:

- GOG_Stats_GetLeaderboardDisplayType
- GOG_Stats_FindOrCreateLeaderboard

Avatar Type Constant	Description
`GOG_LEADERBOARD_DISPLAY_TYPE_NONE`	No display type specified.
`GOG_LEADERBOARD_DISPLAY_TYPE_NUMBER`	Simple numerical score.
`GOG_LEADERBOARD_DISPLAY_TYPE_TIME_SECONDS`	The score represents time, in seconds.
`GOG_LEADERBOARD_DISPLAY_TYPE_TIME_MILLISECONDS`	The score represents time, in milliseconds.

LeaderboardSortMethod

These constants represent the method used to sort scores within a leaderboard, and are used with the following function calls:

- GOG_Stats_GetLeaderboardSortMethod
- GOG_Stats_FindOrCreateLeaderboard

Avatar Type Constant	Description
`GOG_LEADERBOARD_SORT_METHOD_NONE`	No sorting method specified.
`GOG_LEADERBOARD_SORT_METHOD_ASCENDING`	Top score is lowest number.
`GOG_LEADERBOARD_SORT_METHOD_DESCENDING`	Top score is highest number.

Storage

This is a module for managing of cloud storage files.

Functions

The following functions are provided to interact with this module:

- GOG_Storage_DownloadSharedFile
- GOG_Storage_FileDelete
- GOG_Storage_FileExists
- GOG_Storage_FileRead
- GOG_Storage_FileShare
- GOG_Storage_FileWrite
- GOG_Storage_GetDownloadedSharedFileByIndex
- GOG_Storage_GetDownloadedSharedFileCount
- GOG_Storage_GetFileCount
- GOG_Storage_GetFileNameByIndex
- GOG_Storage_GetFileSize
- GOG_Storage_GetFileTimestamp
- GOG_Storage_GetSharedFileName
- GOG_Storage_GetSharedFileOwner
- GOG_Storage_GetSharedFileSize
- GOG_Storage_SharedFileClose
- GOG_Storage_SharedFileRead

GOG_Storage_DownloadSharedFile

Downloads previously shared file.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_Storage_DownloadSharedFile(sharedFileID)
```

Argument	Type	Description
sharedFileID	int64	The ID of the shared file.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Storage_DownloadSharedFile"`
error	string	The error message; only if request failed OPTIONAL
sharedFileID	string	The ID of the shared file.
fileName	string	Name of the shared file

Example:

```
GOG_Storage_DownloadSharedFile(sharedFileID)
```

The code sample above starts a task for download a shared file, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Storage_DownloadSharedFile")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    var sharedFileID = async_load[?"sharedFileID"]
    var fileName = async_load[?"fileName"]

    show_debug_message("DownloadSharedFile SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Storage_FileDelete

Deletes the file.

REQUIREMENT The name that specifies the file has to be provided in the form of a relative path that uses slashes as separators. Every part of the path must not refer to any special or restricted name on any of the supported platforms. Backslashes are not allowed.

Syntax:

```
GOG_Storage_FileDelete(fileName)
```

Argument	Type	Description
fileName	string	The name of the file in the form of a path

Returns:

```
undefined
```

Example:

```
GOG_Storage_FileDelete (fileName)
```

The code above provides a simple usage example.

GOG_Storage_FileExists

Returns if the file exists.

REQUIREMENT The name that specifies the file has to be provided in the form of a relative path that uses slashes as separators. Every part of the path must not refer to any special or restricted name on any of the supported platforms. Backslashes are not allowed.

Syntax:

```
GOG_Storage_FileExists(fileName)
```

Argument	Type	Description
fileName	string	The name of the file in the form of a path

Returns:

```
bool
```

Example:

```
if(GOG_Storage_FileExists(fileName))
{
   //File exists, do something
}
```

The code above provides a simple usage example.

GOG_Storage_FileRead

Reads file content into the buffer.

REQUIREMENT The name that specifies the file has to be provided in the form of a relative path that uses slashes as separators. Every part of the path must not refer to any special or restricted name on any of the supported platforms. Backslashes are not allowed.

WARNING This function creates a new buffer everytime it is called you need to ensure you correctly delete the buffer when you don't need it anymore using the buffer_delete function. Failing to do so will result in memory leaks.

Syntax:

```
GOG_Storage_FileRead(fileName)
```

Argument	Type	Description
fileName	string	The name of the file in the form of a path

Returns:

```
id.buffer
```

Example:

```
var buffer = GOG_Storage_FileRead(fileName)
```

The code above provides a simple usage example.

GOG_Storage_FileShare

Uploads the file for sharing.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

REQUIREMENT The name that specifies the file has to be provided in the form of a relative path that uses slashes as separators. Every part of the path must not refer to any special or restricted name on any of the supported platforms. Backslashes are not allowed.

Syntax:

```
GOG_Storage_FileShare(fileName)
```

Argument	Type	Description
fileName	string	The name of the file in the form of a path

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_Storage_FileShare"`
error	string	The error message; only if request failed OPTIONAL
sharedFileID	string	The ID of the shared file.
fileName	string	Name of the shared file

Example:

```
GOG_Storage_FileShare(fileName)
```

The code sample above starts a task for sharing a file, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_Storage_FileShare")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    var sharedFileID = async_load[?"sharedFileID"]
    var fileName = async_load[?"fileName"]

    show_debug_message("FileShare SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_Storage_FileWrite

Writes data into the file.

REQUIREMENT The name that specifies the file has to be provided in the form of a relative path that uses slashes as separators. Every part of the path must not refer to any special or restricted name on any of the supported platforms. Backslashes are not allowed.

NOTE The files created using this method will be stored in GOG Galaxy internal directory and should be accessed only via Galaxy SDK methods.

Syntax:

```
GOG_Storage_FileWrite(fileName, buffer)
```

Argument	Type	Description
fileName	filename	Name of the file
buffer	id.buffer	Buffer with data

Returns:

```
undefined
```

Example:

```
GOG_Storage_FileWrite(fileName, buffer)
```

The code above provides a simple usage example.

GOG_Storage_GetDownloadedSharedFileByIndex

Returns the ID of the open downloaded shared file.

Syntax:

```
GOG_Storage_GetDownloadedSharedFileByIndex(index)
```

Argument	Type	Description
index	real	Index as an integer in the range of [0, number of open downloaded shared files).

Returns:

```
real
```

Example:

```
for(var i = 0 ; i < GetDownloadedSharedFileCount() ; i++)
{
   var sharedFileID= GOG_Storage_GetDownloadedSharedFileByIndex(index)
   var fileName = GOG_Storage_GetSharedFileName(sharedFileID)
   var fileSize = GOG_Storage_GetSharedFileSize(sharedFileID)
   var owner = GOG_Storage_GetSharedFileOwner( sharedFileID )
   var buffer = GOG_Storage_SharedFileRead(sharedFileID )
}
```

The code above provides a simple usage example.

GOG_Storage_GetDownloadedSharedFileCount

Returns the number of open downloaded shared files.

Syntax:

```
GetDownloadedSharedFileCount()
```

Returns:

```
real
```

Example:

```
for(var i = 0 ; i < GetDownloadedSharedFileCount() ; i++)
{
   var sharedFileID= GOG_Storage_GetDownloadedSharedFileByIndex(index)
   var fileName = GOG_Storage_GetSharedFileName(sharedFileID)
   var fileSize = GOG_Storage_GetSharedFileSize(sharedFileID)
   var owner = GOG_Storage_GetSharedFileOwner(sharedFileID)
   var buffer = GOG_Storage_SharedFileRead(sharedFileID)
}
```

The code above provides a simple usage example.

GOG_Storage_GetFileCount

Returns number of the files in the storage.

Syntax:

```
GOG_Storage_GetFileCount()
```

Returns:

```
real
```

Example:

```
for(i = 0 ; i < GOG_Storage_GetFileCount() ; i++)
{
   var filename = GOG_Storage_GetFileNameByIndex(i)
   var fileSize = GOG_Storage_GetFileSize(filename)
   var fileTimestamp = GOG_Storage_GetFileTimestamp(filename)
   var fileSize = GOG_Storage_GetSharedFileSize(filename)
}
```

The code above provides a simple usage example.

GOG_Storage_GetFileNameByIndex

Returns name of the file.

Syntax:

```
GOG_Storage_GetFileNameByIndex(index)
```

Argument	Type	Description
index	int64	The index as an integer in the range of [0, number of files).

Returns:

```
string
```

Example:

```
for(i = 0 ; i < GOG_Storage_GetFileCount() ; i++)
{
   var filename = GOG_Storage_GetFileNameByIndex(i)
   var fileSize = GOG_Storage_GetFileSize(filename)
   var fileTimestamp = GOG_Storage_GetFileTimestamp(filename)
   var fileSize = GOG_Storage_GetSharedFileSize(filename)
}
```

The code above provides a simple usage example.

GOG_Storage_GetFileSize

Returns the size of the file.

REQUIREMENT The name that specifies the file has to be provided in the form of a relative path that uses slashes as separators. Every part of the path must not refer to any special or restricted name on any of the supported platforms. Backslashes are not allowed.

Syntax:

```
GOG_Storage_GetFileSize(filename)
```

Argument	Type	Description
filename	string	The name of the file in the form of a path.

Returns:

```
real
```

Example:

```
for(i = 0 ; i < GOG_Storage_GetFileCount() ; i++)
{
   var filename = GOG_Storage_GetFileNameByIndex(i)
   var fileSize = GOG_Storage_GetFileSize(filename)
   var fileTimestamp = GOG_Storage_GetFileTimestamp(filename)
   var fileSize = GOG_Storage_GetSharedFileSize(filename)
}
```

The code above provides a simple usage example.

GOG_Storage_GetFileTimestamp

Returns the timestamp of the last file modification.

REQUIREMENT The name that specifies the file has to be provided in the form of a relative path that uses slashes as separators. Every part of the path must not refer to any special or restricted name on any of the supported platforms. Backslashes are not allowed.

Syntax:

```
GOG_Storage_GetFileTimestamp(filename)
```

Argument	Type	Description
filename	string	The name of the file in the form of a path.

Returns:

```
real
```

Example:

```
for(i = 0 ; i < GOG_Storage_GetFileCount() ; i++)
{
   var filename = GOG_Storage_GetFileNameByIndex(i)
   var fileSize = GOG_Storage_GetFileSize(filename)
   var fileTimestamp = GOG_Storage_GetFileTimestamp(filename)
   var fileSize = GOG_Storage_GetSharedFileSize(filename)
}
```

The code above provides a simple usage example.

GOG_Storage_GetSharedFileName

Gets name of downloaded shared file.

Syntax:

```
GOG_Storage_GetSharedFileName(sharedFileID)
```

Argument	Type	Description
sharedFileID	int64	The ID of the shared file.

Returns:

```
string
```

Example:

```
for(var i = 0 ; i < GetDownloadedSharedFileCount() ; i++)
{
   var sharedFileID = GOG_Storage_GetDownloadedSharedFileByIndex(index)
   var fileName = GOG_Storage_GetSharedFileName(sharedFileID)
   var fileSize = GOG_Storage_GetSharedFileSize(sharedFileID)
   var owner = GOG_Storage_GetSharedFileOwner(sharedFileID)
   var buffer = GOG_Storage_SharedFileRead(sharedFileID)
}
```

The code above provides a simple usage example.

GOG_Storage_GetSharedFileOwner

Gets the owner (GalaxyID) of downloaded shared file.

Syntax:

```
GOG_Storage_GetSharedFileOwner(sharedFileID)
```

Argument	Type	Description
sharedFileID	int64	The ID of the shared file.

Returns:

```
struct (GalaxyID)
```

Example:

```
for(var i = 0 ; i < GetDownloadedSharedFileCount() ; i++)
{
   var sharedFileID = GOG_Storage_GetDownloadedSharedFileByIndex(index)
   var fileName = GOG_Storage_GetSharedFileName(sharedFileID)
   var fileSize = GOG_Storage_GetSharedFileSize(sharedFileID)
   var owner = GOG_Storage_GetSharedFileOwner(sharedFileID)
   var buffer = GOG_Storage_SharedFileRead(sharedFileID)
}
```

The code above provides a simple usage example.

GOG_Storage_GetSharedFileSize

Gets size of downloaded shared file.

Syntax:

```
GOG_Storage_GetSharedFileSize(sharedFileID)
```

Argument	Type	Description
sharedFileID	int64	The ID of the shared file.

Returns:

```
real
```

Example:

```
for(var i = 0 ; i < GetDownloadedSharedFileCount() ; i++)
{
   var sharedFileID = GOG_Storage_GetDownloadedSharedFileByIndex(index)
   var fileName = GOG_Storage_GetSharedFileName(sharedFileID)
   var fileSize = GOG_Storage_GetSharedFileSize(sharedFileID)
   var owner = GOG_Storage_GetSharedFileOwner(sharedFileID)
   var buffer = GOG_Storage_SharedFileRead(sharedFileID)
}
```

The code above provides a simple usage example.

GOG_Storage_SharedFileClose

Closes downloaded shared file and frees the memory.

Syntax:

```
GOG_Storage_SharedFileClose(sharedFileID)
```

Argument	Type	Description
sharedFileID	int64	The ID of the shared file.

Returns:

```
undefined
```

Example:

```
GOG_Storage_SharedFileClose(sharedFileID)
```

The code above provides a simple usage example.

GOG_Storage_SharedFileRead

Reads downloaded shared file content into the buffer.

WARNING This function creates a new buffer everytime it is called you need to ensure you correctly delete the buffer when you don't need it anymore using the **buffer_delete** function. Failing to do so will result in memory leaks.

Syntax:

```
GOG_Storage_SharedFileRead(sharedFileID)
```

Argument	Type	Description
sharedFileID	int64	The ID of the shared file.

Returns:

```
id.buffer
```

Example:

```
for(var i = 0 ; i < GetDownloadedSharedFileCount() ; i++)
{
   var sharedFileID = GOG_Storage_GetDownloadedSharedFileByIndex(index)
   var fileName = GOG_Storage_GetSharedFileName(sharedFileID)
   var fileSize = GOG_Storage_GetSharedFileSize(sharedFileID)
   var owner = GOG_Storage_GetSharedFileOwner(sharedFileID)
   var buffer = GOG_Storage_SharedFileRead(sharedFileID)
}
```

The code above provides a simple usage example.

Telemetry

This is a module for handling telemetry.

Functions

The following functions are provided to interact with this module:

- GOG_Telemetry_AddArrayParam
- GOG_Telemetry_AddBoolParam
- GOG_Telemetry_AddFloatParam
- GOG_Telemetry_AddIntParam
- GOG_Telemetry_AddObjectParam
- GOG_Telemetry_AddStringParam
- GOG_Telemetry_ClearParams
- GOG_Telemetry_CloseParam
- GOG_Telemetry_GetVisitID
- GOG_Telemetry_ResetVisitID
- GOG_Telemetry_SendAnonymousTelemetryEvent
- GOG_Telemetry_SendTelemetryEvent
- GOG_Telemetry_SetSamplingClass

GOG_Telemetry_AddArrayParam

Adds an array parameter to be applied next time you call GOG_Telemetry_SendTelemetryEvent or GOG_Telemetry_SendAnonymousTelemetryEver

Syntax:

```
GOG_Telemetry_AddArrayParam(name)
```

Argument	Type	Description
name	string	The name of the parameter

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_ClearParams()
GOG_Telemetry_AddStringParam("hello","world")

GOG_Telemetry_AddArrayParam (name)
GOG_Telemetry_AddIntParam("int",123)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_AddObjectParam(name)
GOG_Telemetry_AddBoolParam("bool",true)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_SendTelemetryEvent(eventType)
```

The code above provides a simple usage example.

GOG_Telemetry_AddBoolParam

Adds a boolean parameter to be applied next time you call GOG_Telemetry_SendTelemetryEvent or GOG_Telemetry_SendAnonymousTelemetryEver

Syntax:

```
GOG_Telemetry_AddBoolParam(name,value)
```

Argument	Type	Description
name	string	The name of the parameter or **empty string** when adding a value to an array.
value	bool	The value of the parameter.

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_ClearParams()
GOG_Telemetry_AddStringParam("hello","world")

GOG_Telemetry_AddArrayParam (name)
GOG_Telemetry_AddIntParam("int",123)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_AddObjectParam(name)
GOG_Telemetry_AddBoolParam("bool",true)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_SendTelemetryEvent(eventType)
```

The code above provides a simple usage example.

GOG_Telemetry_AddFloatParam

Adds a float parameter to be applied next time you call GOG_Telemetry_SendTelemetryEvent or GOG_Telemetry_SendAnonymousTelemetryEver

Syntax:

```
GOG_Telemetry_AddFloatParam(name,value)
```

Argument	Type	Description
name	string	The name of the parameter or **empty string** when adding a value to an array.
value	float	The value of the parameter.

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_ClearParams()
GOG_Telemetry_AddStringParam("hello","world")

GOG_Telemetry_AddArrayParam (name)
GOG_Telemetry_AddIntParam("int",123)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_AddObjectParam(name)
GOG_Telemetry_AddBoolParam("bool",true)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_SendTelemetryEvent(eventType)
```

The code above provides a simple usage example.

GOG_Telemetry_AddIntParam

Adds an integer parameter to be applied next time you call GOG_Telemetry_SendTelemetryEvent or GOG_Telemetry_SendAnonymousTelemetryEver

Syntax:

```
GOG_Telemetry_AddIntParam(name,value)
```

Argument	Type	Description
name	string	The name of the parameter or **empty string** when adding a value to an array.
value	int	The value of the parameter.

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_ClearParams()
GOG_Telemetry_AddStringParam("hello","world")

GOG_Telemetry_AddArrayParam (name)
GOG_Telemetry_AddIntParam("int",123)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_AddObjectParam(name)
GOG_Telemetry_AddBoolParam("bool",true)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_SendTelemetryEvent(eventType)
```

The code above provides a simple usage example.

GOG_Telemetry_AddObjectParam

Adds an object parameter to be applied next time you call GOG_Telemetry_SendTelemetryEvent or GOG_Telemetry_SendAnonymousTelemetryEver

Syntax:

```
GOG_Telemetry_AddObjectParam(name)
```

Argument	Type	Description
name	string	The name of the parameter or **empty string** when adding a value to an array.

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_ClearParams()
GOG_Telemetry_AddStringParam("hello","world")

GOG_Telemetry_AddArrayParam (name)
GOG_Telemetry_AddIntParam("int",123)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_AddObjectParam(name)
GOG_Telemetry_AddBoolParam("bool",true)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_SendTelemetryEvent(eventType)
```

The code above provides a simple usage example.

GOG_Telemetry_AddStringParam

Adds a string parameter to be applied next time you call GOG_Telemetry_SendTelemetryEvent or GOG_Telemetry_SendAnonymousTelemetryEver

Syntax:

```
GOG_Telemetry_AddStringParam(name,value)
```

Argument	Type	Description
name	string	The name of the parameter or **empty string** when adding a value to an array.
value	bool	The value of the parameter.

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_ClearParams()
GOG_Telemetry_AddStringParam("hello","world")

GOG_Telemetry_AddArrayParam (name)
GOG_Telemetry_AddIntParam("int",123)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_AddObjectParam(name)
GOG_Telemetry_AddBoolParam("bool",true)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_SendTelemetryEvent(eventType)
```

The code above provides a simple usage example.

GOG_Telemetry_ClearParams

Clears all parameters that may have been set so far at any level.

Syntax:

```
GOG_Telemetry_ClearParams()
```

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_ClearParams()
GOG_Telemetry_AddStringParam("hello","world")

GOG_Telemetry_AddArrayParam (name)
GOG_Telemetry_AddIntParam("int",123)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_AddObjectParam(name)
GOG_Telemetry_AddBoolParam("bool",true)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_SendTelemetryEvent(eventType)
```

The code above provides a simple usage example.

GOG_Telemetry_CloseParam

Closes an object or array parameter and leaves its scope.

Syntax:

```
GOG_Telemetry_CloseParam()
```

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_ClearParams()
GOG_Telemetry_AddStringParam("hello","world")

GOG_Telemetry_AddArrayParam (name)
GOG_Telemetry_AddIntParam("int",123)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_AddObjectParam(name)
GOG_Telemetry_AddBoolParam("bool",true)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_SendTelemetryEvent(eventType)
```

The code above provides a simple usage example.

GOG_Telemetry_GetVisitID

Retrieves current VisitID.

NOTE Visit ID is used to link subsequent telemetry events that corresponds to the same action (e.x. game session).

Syntax:

```
GOG_Telemetry_GetVisitID()
```

Returns:

```
string
```

Example:

```
var VisitID = GOG_Telemetry_GetVisitID()
```

The code above provides a simple usage example.

GOG_Telemetry_ResetVisitID

Resets current VisitID.

NOTE Visit ID is used to link subsequent telemetry events that corresponds to the same action (e.x. game session).

Syntax:

```
GOG_Telemetry_ResetVisitID()
```

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_ResetVisitID()
```

The code above provides a simple usage example.

GOG_Telemetry_SendAnonymousTelemetryEvent

Sends an anonymous telemetry event.

Syntax:

```
GOG_Telemetry_SendAnonymousTelemetryEvent(eventType)
```

Argument	Type	Description
eventType	string	The type of the event.

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_ClearParams()
GOG_Telemetry_AddStringParam("hello","world")

GOG_Telemetry_AddArrayParam (name)
GOG_Telemetry_AddIntParam("int",123)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_AddObjectParam(name)
GOG_Telemetry_AddBoolParam("bool",true)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_SendAnonymousTelemetryEvent(eventType)
```

The code above provides a simple usage example.

GOG_Telemetry_SendTelemetryEvent

Sends a telemetry event.

Syntax:

```
GOG_Telemetry_SendTelemetryEvent(eventType)
```

Argument	Type	Description
eventType	string	The type of the event.

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_ClearParams()
GOG_Telemetry_AddStringParam("hello","world")

GOG_Telemetry_AddArrayParam (name)
GOG_Telemetry_AddIntParam("int",123)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_AddObjectParam(name)
GOG_Telemetry_AddBoolParam("bool",true)
GOG_Telemetry_AddFloatParam("float", 987.765)
GOG_Telemetry_CloseParam()

GOG_Telemetry_SendTelemetryEvent(eventType)
```

The code above provides a simple usage example.

GOG_Telemetry_SetSamplingClass

Sets a sampling class to be applied next time you call GOG_Telemetry_SendTelemetryEvent or GOG_Telemetry_SendAnonymousTelemetryEver

Syntax:

```
GOG_Telemetry_SetSamplingClass(name)
```

Argument	Type	Description
name	string	The name of the sampling class.

Returns:

```
undefined
```

Example:

```
GOG_Telemetry_SetSamplingClass(name)
```

The code above provides a simple usage example.

User

This is a module for handling the user account.

Functions

The following functions are provided to interact with this module:

- GOG_User_DeleteUserData
- GOG_User_GetAccessToken
- GOG_User_GetGalaxyID
- GOG_User_GetRefreshToken
- GOG_User_GetSessionID
- GOG_User_GetUserData
- GOG_User_GetUserDataByIndex
- GOG_User_GetUserDataCount
- GOG_User_IsLoggedOn
- GOG_User_IsUserDataAvailable
- GOG_User_ReportInvalidAccessToken
- GOG_User_RequestUserData
- GOG_User_SetUserData
- GOG_User_SignedIn
- GOG_User_SignInAnonymous
- GOG_User_SignInAnonymousTelemetry
- GOG_User_SignInCredentials
- GOG_User_SignInGalaxy

- GOG_User_SignInLauncher
- GOG_User_SignInServerKey
- GOG_User_SignInToken
- GOG_User_SignOut

Structures

This module exposes the following data structures:

- GalaxyID

GOG_User_DeleteUserData

Clears a property of user data storage.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

REQUIREMENT Retrieve the user data first by calling GOG_User_RequestUserData.

Syntax:

```
GOG_User_DeleteUserData(key)
```

Argument	Type	Description
key	string	The name of the property of the user data storage.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_User_DeleteUserData"`
error	string	The error message; only if request failed OPTIONAL
name	string	Name of the leaderboard

Example:

```
GOG_User_DeleteUserData("BestScoresLeaderboard")
```

The code sample above starts a task for deleting user data, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_User_DeleteUserData")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }
    var name = async_load[?"name"]
    show_debug_message(" FindLeaderboard SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_User_GetAccessToken

Returns the access token for current session.

Syntax:

```
GOG_User_GetAccessToken()
```

Returns:

```
string
```

Example:

```
var accessToken = GOG_User_GetAccessToken()
```

The code above provides a simple usage example.

GOG_User_GetGalaxyID

Returns the ID of the user, provided that the user is signed in.

Syntax:

```
GOG_User_GetGalaxyID()
```

Returns:

```
struct (GalaxyID)
```

Example:

```
var myUsedID = GOG_User_GetGalaxyID()
```

The code above provides a simple usage example.

GOG_User_GetSessionID

Returns the ID of current session.

Syntax:

```
GOG_User_GetSessionID()
```

Returns:

```
real
```

Example:

```
var sessionID = GOG_User_GetSessionID()
```

The code above provides a simple usage example.

GOG_User_GetRefreshToken

Returns the refresh token for the current version.

Syntax:

```
GOG_User_GetRefreshToken()
```

Returns:

```
string
```

Example:

```
var refreshToken = GOG_User_GetRefreshToken()
```

The code above provides a simple usage example.

GOG_User_GetUserData

Returns an entry from the data storage of current user.

REQUIREMENT Retrieve the user data first by calling GOG_User_RequestUserData.

Syntax:

```
GOG_User_GetUserData(key, userID)
```

Argument	Type	Description
key	string	The name of the property of the user data storage.
userID	GalaxyID	The ID of the user. It can be omitted when reading own data.

Returns:

```
string
```

Example:

```
Discord_Core_Create(appId)
```

The code above provides a simple usage example.

GOG_User_GetUserDataByIndex

Returns a property from the user data storage by index.

REQUIREMENT Retrieve the user data first by calling GOG_User_RequestUserData.

Syntax:

```
GOG_User_GetUserDataByIndex(index, userID)
```

Argument	Type	Description
index	real	Index as an integer in the range of [0, number of entries).
userID	GalaxyID	The ID of the user. It can be omitted when reading own data.

Returns:

```
struct
```

Struct Member	Type	Description
key	string	The name of the property of the user data storage.
value	string	The value of the property of the user data storage.

Example:

```
for(var a = 0 ; a < GOG_User_GetUserDataCount(GOG_User_GetGalaxyID()) ; a ++ )
    draw_text(100,220+a*30,GOG_User_GetUserDataByIndex(a,GOG_User_GetGalaxyID()))
```

The code above provides a simple usage example.

GOG_User_GetUserDataCount

Returns the number of entries in the user data storage.

REQUIREMENT Retrieve the user data first by calling GOG_User_RequestUserData.

Syntax:

```
GOG_User_GetUserDataCount(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user. It can be omitted when reading own data.

Returns:

```
real
```

Example:

```
for(var a = 0 ; a < GOG_User_GetUserDataCount(GOG_User_GetGalaxyID()) ; a ++ )
    draw_text(100,220+a*30,GOG_User_GetUserDataByIndex(a,GOG_User_GetGalaxyID()))
```

The code above provides a simple usage example.

GOG_User_IsLoggedOn

Checks if the user is logged on to Galaxy backend services.

Syntax:

```
GOG_User_IsLoggedOn()
```

Returns:

```
bool
```

Example:

```
if(GOG_User_IsLoggedOn())
{
   //User is logged, do something...
}
```

The code above provides a simple usage example.

GOG_User_IsUserDataAvailable

Checks if user data exists.

REQUIREMENT Retrieve the user data first by calling GOG_User_RequestUserData.

Syntax:

```
GOG_User_IsUserDataAvailable(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user. It can be omitted when reading own data.

Returns:

```
bool
```

Example:

```
if(GOG_User_IsUserDataAvailable())
{
   //User data is available, do something
}
```

The code above provides a simple usage example.

GOG_User_ReportInvalidAccessToken

Reports current access token as no longer working.

Syntax:

```
GOG_User_ReportInvalidAccessToken(accesstToken, info)
```

Argument	Type	Description
accessToken	string	The invalid access token.
info	string	Additional information.

Returns:

```
undefined
```

Example:

```
GOG_User_ReportInvalidAccessToken(accesstToken,"A hacker wanna be...")
```

The code above provides a simple usage example.

GOG_User_RequestUserData

Retrieves/Refreshes user data storage.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_User_RequestUserData(userID)
```

Argument	Type	Description
userID	GalaxyID	The ID of the user. It can be omitted when reading own data.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_User_RequestUserData"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_User_RequestUserData("BestScoresLeaderboard")
```

The code sample above starts a task for requesting user data, which results can be caught inside an **Async Social** event.

```
if (async_load[? "type"] == "GOG_User_RequestUserData")
{
    show_debug_message("RequestUserData SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_User_SetUserData

Creates or updates an entry in the user data storage.

REQUIREMENT Retrieve the user data first by calling GOG_User_RequestUserData.

Syntax:

```
GOG_User_SetUserData(key, value)
```

Argument	Type	Description
key	string	The name of the property of the user data storage with the limit of 1023 bytes.
value	string	The value of the property to set with the limit of 4095 bytes.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_User_SetUserData"`
userID	GalaxyID	The ID of the user.

Example:

```
GOG_User_SetUserData("weapon", "sword");
```

The code sample above starts a task for setting user data, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_User_SetUserData")
{
    show_debug_message("User data was set for user {0}", async_load[? "userID"]);
}
```

This code sample provides an example of handling the returned callback data.

GOG_User_SignedIn

Checks if the user is signed in to Galaxy.

The user should be reported as signed in as soon as the authentication process is finished.

If the user is not able to sign in or gets signed out, there might be either a networking issue or a limited availability of Galaxy backend services.

After loosing authentication the user needs to sign in again in order for the Galaxy Peer to operate.

Syntax:

```
GOG_User_SignedIn()
```

Returns:

```
bool
```

Example:

```
if(GOG_User_SignedIn())
{
   //user signed in, do something
}
```

The code above provides a simple usage example.

GOG_User_SignInAnonymous

Authenticates the Galaxy Game Server anonymously.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_User_SignInAnonymous()
```

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description
type	string	`"GOG_User_SignInAnonymous"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_User_SignInAnonymous()
```

The code sample above starts a task for signing in anonymously, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_User_SignInAnonymous")
{
    if (ds_map_exists(async_load,"error"))
    {
```

```
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message(" SignInAnonymous SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_User_SignInAnonymousTelemetry

Authenticates the Galaxy Peer anonymously.

This authentication method enables the peer to send anonymous telemetry events.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_User_SignInAnonymousTelemetry()
```

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_User_SignInAnonymousTelemetry"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_User_SignInAnonymousTelemetry()
```

The code sample above starts a task for signing in anonymously (with anonymous telemetry), which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_User_SignInAnonymousTelemetry")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message(" SignInAnonymousTelemetry SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_User_SignInCredentials

Authenticates the Galaxy Peer with specified user credentials.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_User_SignInCredentials(login, password)
```

Argument	Type	Description
login	string	The user's login.
password	string	The user's password.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_User_SignInCredentials"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_User_SignInServerKey(refreshToken)
```

The code sample above starts a task for signing in with credentials, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_User_SignInCredentials")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("SignInToken SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_User_SignInGalaxy

Authenticates the Galaxy Peer based on Galaxy Client authentication.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_User_SignInGalaxy(requireOnline)
```

Argument	Type	Description
requireOnline	bool	Indicates if sing in with Galaxy backend is required.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_User_SignInGalaxy"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_User_SignInGalaxy()
```

The code sample above starts a task for signing in with galaxy client authentication, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_User_SignInGalaxy")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message(" SignInGalaxy SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_User_SignInLauncher

Authenticates the Galaxy Peer based on launcher authentication.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_User_SignInLauncher()
```

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_User_SignInLauncher"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_User_SignInLauncher()
```

The code sample above starts a task for signing in with launcher authentication, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_User_SignInLauncher")
{
    if (ds_map_exists(async_load,"error"))
    {
```

```
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("SignInLauncher SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_User_SignInServerKey

Authenticates the Galaxy Peer with a specified server key.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_User_SignInServerKey(key)
```

Argument	Type	Description
key	string	The server key.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_User_SignInServerKey"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_User_SignInServerKey(key)
```

The code sample above starts a task for signing in using a server key, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_User_SignInServerKey")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("SignInServerKey SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_User_SignInToken

Authenticates the Galaxy Peer with refresh token.

This is an asynchronous function that will trigger the an Async Social when the task is finished.

Syntax:

```
GOG_User_SignInToken(refreshToken)
```

Argument	Type	Description
refreshToken	string	The refresh token obtained from Galaxy login page.

Returns:

```
undefined
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"GOG_User_SignInToken"`
error	string	The error message; only if request failed OPTIONAL

Example:

```
GOG_User_SignInServerKey(refreshToken)
```

The code sample above starts a task for signing in using a refresh token, which results can be caught inside an Async Social event.

```
if (async_load[? "type"] == "GOG_User_SignInToken")
{
    if (ds_map_exists(async_load,"error"))
    {
        show_debug_message(async_load[?"error"])
        exit
    }

    show_debug_message("SignInToken SUCCESS")
}
```

This code sample provides an example of handling the returned callback data.

GOG_User_SignOut

Signs the Galaxy Peer out.

Syntax:

```
GOG_User_SignOut()
```

Returns:

```
undefined
```

Example:

```
GOG_User_SignOut()
```

The code above provides a simple usage example.

GalaxyID

This structure represents a GOG ID (User or Lobby) and is used all across the GOG API as either a return value or function argument.

WARNING These values shouldn't be used or changed and are to be passed back into the extension to perform the desired actions.

Key	Type	Description
ID	real	The real ID value.
IDType	real	The type of the ID.

Utils

This is a module for managing various aspects of the framework (ie.: overlay, images, services...).

Functions

The following functions are provided to interact with this module:

- GOG_Utils_DisableOverlayPopups
- GOG_Utils_GetGogServicesConnectionState
- GOG_Utils_GetImageRGBA
- GOG_Utils_GetImageSize
- GOG_Utils_GetOverlayState
- GOG_Utils_IsOverlayVisible
- GOG_Utils_ShowOverlayWithWebPage

Constants

This module exposes the following constants:

- OverlayState
- ServicesConnectionState

GOG_Utils_DisableOverlayPopups

Disable overlay pop-up notifications.

Hides overlay pop-up notifications based on the group specified below:

- `"chat_message"` - New chat messages received.
- `"friend_invitation"` - Friend request received, new user added to the friend list.
- `"game_invitation"` - Game invitation sent or received.

NOTE For this call to work, the overlay needs to be initialized first. To check whether the overlay is initialized successfully, call GOG_Utils_GetOverlayState.

Syntax:

```
GOG_Utils_DisableOverlayPopups(popupGroup)
```

Argument	Type	Description
popupGroup	string	The name of the notification pop-up group.

Returns:

```
undefined
```

Example:

```
GOG_Utils_DisableOverlayPopups("chat_message")
```

The code above provides a simple usage example.

GOG_Utils_GetGogServicesConnectionState

Return current state of the connection to GOG services, see ServicesConnectionState.

Syntax:

```
GOG_Utils_GetGogServicesConnectionState()
```

Returns:

```
real (ServicesConnectionState)
```

Example:

```
GOG_Utils_GetGogServicesConnectionState()
```

The code above provides a simple usage example.

GOG_Utils_GetImageRGBA

Reads the image of a specified ID, into a buffer.

NOTE The size of the output buffer will be 4 * height * width (see function GOG_Utils_GetImageSize).

WARNING This function creates a new buffer everytime it is called you need to ensure you correctly delete the buffer when you don't need it anymore using the buffer_delete function. Failing to do so will result in memory leaks.

Syntax:

```
GOG_Utils_GetImageRGBA(imageID)
```

Argument	Type	Description
imageID	real	The ID of the image.

Returns:

```
Id.Buffer
```

Example:

```
var buff = GOG_Utils_GetImageRGBA(imageID)

var size = buffer_get_size(buff)
var L = sqrt(size/4)

surf = surface_create(L,L)
buffer_set_surface(buff,surf,0)

buffer_delete(buff)
```

The code above provides a simple usage example.

GOG_Utils_GetImageSize

Reads width and height of the image of a specified ID.

Syntax:

```
GOG_Utils_GetImageSize(imageID)
```

Argument	Type	Description
imageID	real	The ID of the image.

Returns:

```
struct
```

Struct Member	Type	Description
width	real	The width of the image.
height	real	The height of the image.

Example:

```
var struct = GOG_Utils_GetImageSize(imageID)
var width = struct.width
var height = struct.height
```

The code above provides a simple usage example.

GOG_Utils_GetOverlayState

Return current state of the overlay.

Syntax:

```
GOG_Utils_GetOverlayState()
```

Returns:

```
real (OverlayState)
```

Example:

```
if(GOG_Utils_GetOverlayState() == GOG_OVERLAY_STATE_INITIALIZED)
{
   //Overlay initialized, do something
}
```

The code above provides a simple usage example.

GOG_Utils_IsOverlayVisible

Return current visibility of the overlay.

Syntax:

```
GOG_Utils_IsOverlayVisible()
```

Returns:

```
bool
```

Example:

```
if(GOG_Utils_IsOverlayVisible())
{
   //Overlay visible, do something
}
```

The code above provides a simple usage example.

GOG_Utils_ShowOverlayWithWebPage

Shows web page in the overlay.

Syntax:

```
GOG_Utils_ShowOverlayWithWebPage(url)
```

Argument	Type	Description
url	string	The URL address of a specified web page with the limit of 2047 bytes.

Returns:

```
undefined
```

Example:

```
GOG_Utils_ShowOverlayWithWebPage("https://gamemaker.io")
```

The code above provides a simple usage example.

OverlayState

These constants represent the currect overlay state, and are used with the following function calls:

- GOG_Utils_GetOverlayState

Overlay State Constant	Description
`GOG_OVERLAY_STATE_UNDEFINED`	Overlay state is undefined.
`GOG_OVERLAY_STATE_NOT_SUPPORTED`	Overlay is not supported.
`GOG_OVERLAY_STATE_DISABLED`	Overlay is disabled by the user in the Galaxy Client.
`GOG_OVERLAY_STATE_FAILED_TO_INITIALIZE`	Galaxy Client failed to initialize the overlay or inject it into the game.
`GOG_OVERLAY_STATE_INITIALIZED`	Overlay has been successfully injected into the game.

ServicesConnectionState

These constants represent the current state of the services connection, are used with the following function calls:

- GOG_Utils_GetGogServicesConnectionState

Avatar Type Constant	Description
`GOG_SERVICES_CONNECTION_STATE_UNDEFINED`	Connection state is undefined.
`GOG_SERVICES_CONNECTION_STATE_CONNECTED`	Connection to the GOG services has been established.
`GOG_SERVICES_CONNECTION_STATE_DISCONNECTED`	Connection to the GOG services has been lost.
`GOG_SERVICES_CONNECTION_STATE_AUTH_LOST`	Connection to the GOG services has been lost due to lose of peer authentication.