Ludosity's

Steamworks Wrapper

A fully managed .NET wrapper for Steamworks

What is this?

Ludosity's Steamworks Wrapper is a fully managed .NET wrapper for the Steamworks API developed by Ludosity AB. The purpose is to make it easier to implement Steamworks features in games using Unity as well as XNA / Monogame.

The library aims to be a complete a 1-to-1 mapping of the Steamworks C++ API. This means that the usage pattern is more or less the same as if C++ where used. This also means that the official help resources can be used as most things will work the same. The main differences are how callbacks are handled.

Where to start?

Start by reading the Setup instructions. If you are updating from an older version, check out the Change log.

Then take a look at the Steam class and the example scripts included in the package (Steamworks.cs, Stats.cs). To run the example, just attach these scripts to any GameObject and press Play.

Refer to Common errors if something goes wrong.

What is included?

The entire Steamworks API are exposed at the moment, this enables support for:

- Apps
- Cloud
- Stats
- Achievements
- Friends
- Leaderboards
- Matchmaking
- MatchmakingServers
- GameServer
- GameServerStats
- Networking
- Screenshots
- Utils
- HTTP
- Big Picture Mode
- UGC
- SteamVR (Not available in Linux/Mac, the SDK does not yet include support for this)
- Steam Controller (Temporarily not available in Linux/Mac)

Feel free to contact us if you have any specific feature needs and we will try to make them a priority.

The library is not designed to be thread safe.

Almost all method comments are (for the moment) simply copy-pasted from the regular Steamworks C++ headers. So any names mentioned in these comments refer to the C++ names, but these names are almost the same as the names used in this library.

Micro Transactions

Micro transactions are handled a bit different than the rest of the API, we may provide an easy-to-use interface in the future, but for now you'll have to use the ISteamHTTP interface and make requests to valve's web api.

Warning! When using the micro transaction part of the web API you'll need something called a publisher key (which you can request from valve). It is very important that this publisher key only is used on the server-side of you solution, this is never meant for the client to see.

SteamVR

The SteamVR is only available in Windows but the VR is still just experimental in the SDK and requires a bit of work to get going. The way to access it in code is through SteamInterface::Hmd.

A Valve employee explains how to get Steam to start in VR-mode here:
http://steamcommunity.com/app/250820/discussions/0/630802344730488459/

Once you have followed those steps, you should have a folder called SteamVR in your <steam_install_path>\steamapps\. Copy that folder into a folder called vr in the same directory as your exe to get started.

Keep in mind that this is still in experimental mode, during testing I had problems launching my test-application due to Steam not letting it start.

Software requirements

This library requires access to the Steamworks SDK to work (currently version 1.28).

To be able to use this library on a Windows computer the Microsoft Visual C++ 2010 Redistributable Package (x86) needs to be installed. This applies to both developers and end users. This means that the redist package needs to be included with the final game.

This library is built against Mac OS X 10.6 and have been tested on Mac OS X 10.7.4.

This library is built against Ubuntu 12.04 32-bit version. It has not been properly tested on other Linux distributions yet. **Currently we only support 32-bit versions of Linux.** 64-bit is being worked on, stay tuned for an announcement on an official relase.

Known bugs

When the library is used from within the editor, the Steam client will think that you are playing the game until the editor is closed, even if play mode is stopped. This will however not affect the usage of this library as it is designed to handle this if used as in the example.

The Game overlay will not always work when running the game from the Unity editor. To make the overlay display correctly you may have to publish a built version of your game to a local content server and run the game via the steam client. Refer to the official Steamworks documentation on how to setup a local content server.

The Steam Controller and SteamVR is not available in Linux/Mac, some of the functions are still available, but will not result in anything.

Support

If you have any bug reports, feature requests or questions regarding the wrapper itself, please send an email to **steamworks@ludosity.com**.

Please include the version of the library that you are using (Version Info) and the invoice number from the Unity Asset Store when contacting support. This will enable us to verify your purchase and can significantly speed up turnaround times as we will be able to send you development builds to verify that bug fixes work.

We can unfortunately only help with things that have to do directly with the wrapper, questions about the usage of the Steamworks API itself needs to be directed to the official Steamworks mailing list.

Setup instructions

Windows

Download and install the appropriate version of the Steamworks SDK (currently 1.8).

Create a file named steam_appid.txt and place it on the same level as the Assets folder in the Unity project. For example, if you create a Unity project called Bob, Unity will create a folder called Bob that contains the Assets, Library and Temp folders. Place the steam_appid.txt file directly in the Bob folder.

Enter you AppID (assigned to you by Valve) into the steam_appid.txt file and save it. Make sure that you save the file in regular ANSI encoding, or there may be problems reading the file.

When you build the game for testing, you will have to manually copy the steam_appid.txt to the same folder where the .exe file is located.

To be able to use the library from the editor, you have to copy the **steam_api.dll** and **SteamworksNative.dll** files into the project folder at the same place where you placed the steam_appid.txt file. To be able to build the project you will also have to copy **SteamworksManaged.dll** into your assets folder.

When you build the game you will also have to manually copy the **steam_api.dll** file to the same folder where the built .exe file is placed. The folder stucture should be like this:

- Game_Data/
- Game.exe
- steam_api.dll
- steam_appid.txt
- SteamworksNative.dll

When using the 64-bit version of the Windows wrapper, which can be found in the .unitybundle, you'll need to use the DLLs in the **Windows 64-bit** folder. You'll also need to use **steam_api64.dll** instead of the **steam_api.dll.**

Mac

Download and install the appropriate version of the Steamworks SDK (currently 1.26a)

Create a file named steam_appid.txt and pace it on the same level as the Assets folder in the Unity project. For example, if you create a Unity project called Bob, Unity will create a folder called Bob that contains the Assets, Library and Temp folders. Place the steam_appid.txt file directly in the Bob folder.

Enter you AppID (assigned to you by Valve) into the steam_appid.txt file and save it. Make sure that you save the file in regular ANSI encoding, or there may be problems reading the file.

To be able to use the library from the editor, you have to copy **SteamworksNative.bundle** to a folder named **Plugins** in your Assets folder, then copy **libsteam_api.dylib** from the Steamworks SDK to Steamworks.bundle/Contents/MacOS. You will also need to copy **SteamworksManaged.dll** to your Assets folder. So in the end you're project folder should look something like this:

- Assets/
    - Plugins/
        - SteamworksNative.bundle/
            - Contents/

        - MacOS/
            - SteamAPIWrap
            - Libsteam_api.dylib
        - Info.plist
    - SteamworksManaged.dll
    - YourCode.cs
- steam_appid.txt

Linux

Currently the Unity editor is not available in Linux, but you can still export and play your games in Linux. Refer to previous setup instructions for Windows and Mac to set them up to export to Linux.

To be able to run the game the user will need to have **libsteam_api.so** in a place where Linux will look for library-files, for example in the same directory as your game, or /usr/lib/.

The file structure will look very similar to how it does in windows, the differences is that you will need l**ibsteam_api.so** instead of **steam_api.dll** and also need **libSteamworksNative.so** alongside **libsteam_api.so** instead of **SteamworksNative.dll**, so it should be something like:

- Game_Data/
- Game.x86
- libsteam_api.so
- steam_appid.txt
- libSteamworksNative.so
-

Common errors

- **DllNotFoundException** is thrown even if the SteamworksNative.dll / SteamworksNative.bundle / libSteamworksNative.so are in the right folder.
    - **On Windows:** Make sure that you have copied steam_api.dll to the Unity project folder. See the Windows setup instructions for more detailed information. The error can also occur if the Microsoft Visual C++ 2010 Redistributable Package (x86) is not installed.
    - **On Mac:** Make sure that you have copied libsteam_api.dylib into the bundle. See the Mac setup instructions for detailed information.

    - **On Linux:** Make sure that you have libsteam_api.so in a valid location. SteamworksNative.dll. See the Linux setup instructions for detailed information.

- **Linux executable can't be opened,** this is the result of a problem with building the Linux executable on windows, they don't get the right permissions. Fix this typing **sudo chmod +x ./<your_execulable>** in the terminal.

- **CallbackStructSizeMismatchException** This is a result of the struct in C# not matching the one in C++ from Valve, and this is an error on our side. This error is quite common around new versions of the wrapper, when Valve change the structs and we don't change ours. When you receive this error you are prompted to report it (we may automate this in the future), make sure to also say what version of the plugin you're using and on what operating system this occurs.

-

Frequently Asked Questions

How do i do <insert question here> in Steamworks?

If you're wondering how to implement something that is related to Steamworks, you should first consult the Spacewar project provided by Valve in the SDK folder that you download from them. Spacewar is a C++ project where they try to demonstrate as many relevant techniques as possible.

If you don't find your answer in the Spacewar project you can always visit https://partner.steamgames.com and search for your answer or join their mailing list. Since we strive towards being an 1:1 mapping of Valves Steamworks, any of their official documentation should work with our library as well.

Can I access the source code?

Yes! We offer access to the source code for $300 (or $250 if you've already bought it on the Asset Store) which will grant you access to our Bitbucket repository. With source access, you can extend or modify the wrapper as you please, and use the results it in your commercial products.

Email **steamworks@ludosity.com** for more info on how to get source access!

Any other questions?

If you don't find help in any of the options above you can always mail us at **steamworks@ludosity.com** and we'll try to help you as best as we can.

Change log

Version 1.8.128

- Added the SteamController and Hmd (SteamVR), based on the matching classes in the SDK.
- Linked agains Steamworks SDK 1.28.
- New functions: Utils::IsSteamRunningInVR().
- A couple of functions changed to match the SDK version of the function.
- Made a couple of functions in SteamGameServer obsolete, due to them soon beeing removed from the SDK.
- Temporarily removed 64-bit support for Windows applications. This should be available again very soon.

Version 1.7.126a

- Added the UGC interface, based on the ISteamUGC class (use the UGC interface instead of Cloud as often as possible since Valve will probably stop supporting, or remove that part of Cloud soon).
- Linked against Steamworks SDK 1.26a.
- New function: Friends::GetPlayerNickname().
- More data in the ValidateAuthTicketResponse and GSClientApprove callbacks, they now contain the SteamID of the owner of the current game.
- RestartAppIfNecessary is now a static functions that can, and should, be called before Initialize.
- Some changes have been made to AuthTicket functions in GameServer to be more like the ones in User.
- Added support for 64-bit Windows applications.

Version 1.6.125

- Updated Cloud::GetPublishedFileDetails to match Steamworks SDK 1.25 update.
- New function: User::GetBadgeLevel(), User::GetPlayerSteamLevel() and Steam::RestartAppIfNecessary().
- New ways to handle friend actions through the overlay. Friend can now be added or removed and friend requests can be accepted or ignored directly though Friends::ActivateGameOverlayToUser().
- Fixes for lots of StructSizeMismatchExceptions.
- Linked against Steamworks SDK 1.25.

Version 1.5.123a

- Finished the HTTP interface, based on the ISteamHTTP class.
- Finished the Screenshots interface, based on the ISteamScreenshots class.
- Added Linux support.

- Linked against Steamworks SDK 1.23a.

Version 1.4.123

- Finished the User interface, based on the ISteamUser class.
- Added the Apps interface, based on the ISteamApps class.
- Linked against Steamworks SDK 1.23.
- Added Cloud::UGCDownloadToLocation (this is one of the functions 1.23 added, we can't add the other one yet as we haven't exposed ISteamScreenshot).
- Fixed Big Picture mode input support.
- Added more example code.
- Fixed intellisense documentation, last version was lacking in documentation since intellisense was broken and oxygen was removed.
- Made sure that every function we've exposed had an alternative version where we don't use the 'out' keyword, but returns a struct.

Version 1.3.122

- Linked against Steamworks SDK 1.22 (No support for Big Picture gamepad input yet).
- Added the Utilities interface, based on the ISteamUtils class.
- The game overlay is now fully functional.
- Removed the automatic documentation as it was broken.

Version 1.2.119

- Linked against Steamworks SDK 1.19.
- Renamed the dll/bundle from SteamAPIWrap to SteamworksNative
- Moved all constants to the Constants class
- Added all the remaining features from the ISteamRemoteStorage class to the Cloud interface
- Enabled all the remaining features from the ISteamFriends class to the Friends interface
- Renamed things to better match the C++ API, might cause some compile errors. Sorry, will avoid it in the future.
- Added some missing items to the LeaderboardEntry struct.
- Documentation updates.
- Added the Matchmaking interface, based on the ISteamMatchMaking class
- Added the GameServer interface, based on the ISteamGameServer class

Version 1.1.452

- The game overlay can now be used to open things like the achievements list, the stats page and arbitrary web pages.

Version 1.0.427

- Fixed internal marshaling error when unlocking achievements.
- Added better handling of errors in native code.

Version 1.0.403

- All the structures that wrap handles can now use .Equals() and == to test for equality.
- Added an example of how to use Stats.

Version 1.0.352

- Initial release