

Developer

/ Documentation

/ Unreal Engine ▾

/ Unreal Engine 5.4 Documentation

/ Programming and Scripting

/ Online Subsystems and Services

/ Online Subsystem

/ Online Subsystem Steam

Online Subsystem Steam

An overview of Online Subsystem Steam, including how to set up your project for distribution on Valve's Steam platform.



Choose your operating system

 Windows  macOS  Linux

Contributors: Valve

The **Online Subsystem Steam API** enables you to ship Unreal Engine (UE) applications to [Valve's Steam platform](#). The main purpose of the **Steam** module is to help you distribute your application with a set of features (such as matchmaking and leaderboards) to Steam users. Additionally, the Steam module implements several of the interfaces being exposed by the [Online Subsystem](#), supporting most of what is offered by the Steamworks Software Development Kit (SDK).

Some of the available Steam Interfaces include:

- Matchmaking (Lobbies and GameServer APIs)
- Leaderboards
- Achievements
- Voice
- UserCloud

- SharedCloud
- External UI



Reference our [Online Subsystem Steam API Reference](#) for a more complete listing of currently available Steam Interfaces.

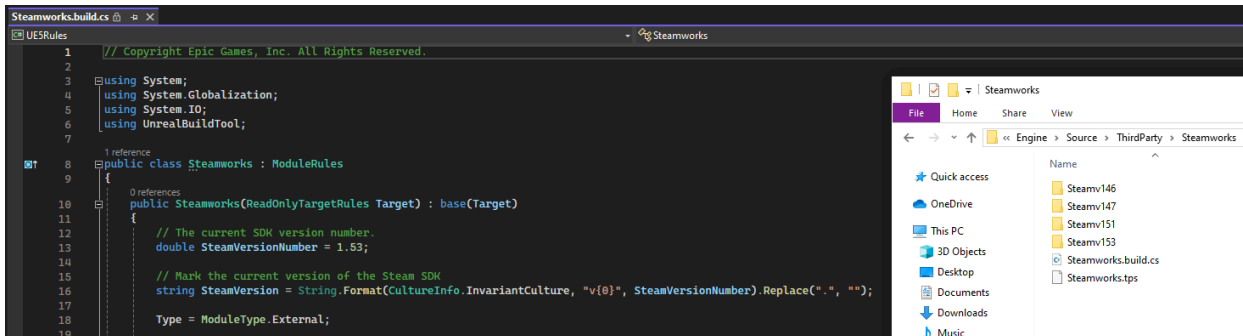
Meeting Valve's Requirements

The Steam Subsystem requires additional setup through [Valve Steamworks](#). Please contact [Valve](#) and reference their [Steamworks SDK Documentation](#) to make sure that your application meets Valve's requirements before attempting to use Steam with UE.

Downloading Steamworks

If your application meets Valve's requirements, go ahead and download the latest version of the [Steamworks SDK](#). Once you have downloaded your desired SDK version, unzip and copy the SDK to `../Engine/Source/ThirdParty/Steamworks/Steam<VERSION>/sdk` where `<VERSION>` is the SDK version number. For example, if you are using Steamworks Version 1.53, the `<VERSION>` is `v153`.

If you're updating your project's Steamworks SDK, make sure to update the `SteamVersionNumber` in your `Steamworks.build.cs` file, which is located in the project directory. Updating this value also updates the path for your Steamworks SDK so Unreal Engine uses the correct SDK version.



Click image to expand.

Setting up the Steamworks SDK

Using Steam against the precompiled version of the engine should only require copying some of the dynamically linked libraries from Valve's SDK into the appropriate folders. If you intend to recompile the engine against the source, putting the SDK in the right place is required as well. Now, copy the relevant redistributable files from the `/redistributable_bin/` directory of the SDK to the following locations:

Some of the 64 bit libraries can be found in your Steam client directory (at the time of this writing, Valve didn't include all of the libraries in the SDK).

`/YourUnrealEnginePath/Engine/Binaries/ThirdParty/Steamworks/Steam[CurrentVersion]/Win64` `steam_api64.dll` `steamclient64.dll` `tier0_s64.dll`
`vstdlib_s64.dll`

`/YourUnrealEnginePath/Engine/Binaries/ThirdParty/Steamworks/Steam[CurrentVersion]/Win32` `steam_api.dll` `steamclient.dll` `tier0_s.dll` `vstdlib_s.dll`

Both `tier0_s.dll` and `vstdlib_s.dll` files are only required when explicitly linking against them using the `"-force_steamclient_link"` flag for dedicated server builds. Client builds

never need these files.

In UE version 4.22 and earlier:

`/YourUnrealEnginePath/EngineOrGameFolder/Binaries/Mac/YourGame.app/Contents/MacOS/libsteam_api.dylib` (from `/redistributable_bin/osx32` - the [dynamic library](#) (.dylib) has both 32 and 64 bit support)

Starting in UE version 4.23, Mac users must also run the following command:

`/YourUnrealEnginePath/Engine/Binaries/ThirdParty/Steamworks/Steam[CurrentVersion]/Mac/` * `libsteam_api.dylib`, generated by running the following command:

`install_name_tool -id @rpath/libsteam_api.dylib /redistributable_bin/osx32/libsteam_api.dylib`



Linux users must link and ship the following files alongside the executable.

`/YourUnrealEnginePath/Engine/Binaries/ThirdParty/Steamworks/Steam[CurrentVersion]/i686-unknown-linux-gnu/libsteam_api.so` *

`redistributable_bin/linux32/libsteam_api.so`

`/YourUnrealEnginePath/Engine/Binaries/ThirdParty/Steamworks/Steam[CurrentVersion]/x86_64-unknown-linux-gnu/libsteam_api.so` *

`redistributable_bin/linux64/libsteam_api.so`

Steam App ID

All games using the Steam Online Subsystem must have a valid application ID because the Steamworks API won't initialize if it doesn't know your application's Steam App ID. Before initializing Steam, UE will generate `steam_appid.txt` (during a graceful shutdown of the engine, UE deletes this file). It's important to note that `steam_appid.txt` must be located in the same directory as your application's executable file because Steam will look for the text file in the current working directory. Additionally, the file should not be included in any Steam images.

If you open `steam_appid.txt`, you'll see a **SteamDevAppId** entry, which is a field that hints your application's ID to Steam. This makes it unnecessary to launch the game using the Steam client (although it must be running).



If you want to test your application, you can use a **SteamDevAppId** of `480`, which is a test App ID that is shared by all developers. Although you can test most Steam Interfaces with the aforementioned test App ID, your application will need a Steam App ID before being shipped.

Shipping Builds

In shipping builds, the engine will check to make sure that the logged-in user is properly subscribed to the game and will shutdown if the engine's test returns false (this is one way to help secure the game). Additionally, using Steam DRM (see the Steamworks SDK) should further protect the game from being tampered with.

Configuring your Application's Settings



If you're creating a new project using the **Unreal Project Browser**, **Online Subsystem** settings shouldn't exist in `DefaultEngine.ini`; however, if you're modifying one of our Sample Projects, the **Online Subsystem** settings might already exist.

Now that you've set up the Steamworks SDK for your application (along with setting up the Steam App ID), you're ready to configure your application's settings to use Online Subsystem Steam.

Steps

1. Open your project's `DefaultEngine.ini` file and add the following setting:

```
1 [/Script/Engine.GameEngine]
2 +NetDriverDefinitions=
   (DefName="GameNetDriver",DriverClassName="OnlineSubsystemSteam.SteamNetDri
```

 Copy full snippet

NetDriverDefinitions describes the net drivers available to UE with the following properties being defined:

- **DefName** is the unique name of this net driver definition.
- **DriverClassName** is the class name of the primary net driver.
- **DriverClassNameFallback** is the class name of the fallback net driver if the primary net driver class fails to initialize.


2. To tell UE to use Online Subsystem Steam, add the following setting:

```
1 [OnlineSubsystem]
2 DefaultPlatformService=Steam
```

 Copy full snippet

3. Now that you've told UE that you want your application to use the Steam Online Subsystem, you'll need to configure the **OnlineSubsystemSteam** module by adding the following settings:

```
1 [OnlineSubsystemSteam]
2 bEnabled=true
3 SteamDevAppId=480
```

 Copy full snippet

4. Specify the Steam class to the Net Driver for the application's connections:

```
1 [/Script/OnlineSubsystemSteam.SteamNetDriver]
2 NetConnectionClassName="OnlineSubsystemSteam.SteamNetConnection"
```

 Copy full snippet

After this, the next steps depend on whether you are using **Sessions** or **Lobbies** for your game.

Using Sessions

If you are using Sessions in your game, add the following under `OnlineSubsystemSteam`:

```
bInitServerOnClient=true
```

 Copy full snippet

If you are using Lobbies, you do not need to set this value.



You need to enable the `bInitServerOnClient` setting for users to be able to create and join sessions. If you do not enable this setting, the Steam online subsystem will not initialize correctly. The Create Session Blueprint node and equivalent C++ functions will not work, and you will not be able to join sessions.

End Result

At the end of this brief guide, your application's `DefaultEngine.ini` file should look like the following setting block. If you want to see how other projects set up and use the Online Subsystem, please refer to our library of Sample Projects.

Finished Settings

DefaultEngine.ini

```
1 [/Script/Engine.GameEngine]
2 +NetDriverDefinitions=
   (DefName="GameNetDriver",DriverClassName="OnlineSubsystemSteam.SteamNetDriver
3
4 [OnlineSubsystem]
5 DefaultPlatformService=Steam
6
7 [OnlineSubsystemSteam]
8 bEnabled=true
9 SteamDevAppId=480
10
11 ; If using Sessions
12 ; bInitServerOnClient=true
13
```

```
14 [/Script/OnlineSubsystemSteam.SteamNetDriver]
15 NetConnectionClassName="OnlineSubsystemSteam.SteamNetConnection"
```

 Copy full snippet

Servers and Lobbies


Steam supports peer-to-peer matchmaking (for both dedicated and listen server games) through lobbies, and provides the ability to run dedicated servers. Lobbies give users the ability to learn information about game servers, and are usually used to convey game-specific information about listen server games, such as what map or mode is being played. Users can also see the number of other users connected to a lobby without joining, through the matchmaking system. Dedicated servers are separate binaries from lobbies, as their task is to run and host the server-side portion of the game. For more information about Steam's features, implementation, or developer interface, please refer to the [partner documentation](#) on the Steam site.

Lobby Details

Lobbies are essentially chat rooms that exist as peer-to-peer instances on Steam's backend service. Unlike servers, no information about a lobby is available before joining, such as ping times or current number of other users. They are typically used for listen servers. To set up a lobby, set the `bUsesPresence` flag and the `bUseLobbiesIfAvailable` flag to `true`. You can set these in the `FOnlineSessionSettings` object passed to the `IOnlineSession::CreateSession` method in your C++ code.

Server Details

To set up a server instance, make sure the `bUsesPresence` flag is `false`. Dedicated servers require the following three macros, and all three must match the values from Steam's dedicated server tool page, found on the partner panel. The Steam network will not show the dedicated server if these values do not match:

UE Macro	Steam Name	Description
<code>UE_PROJECT_STEAMPRODUCTNAME</code>	<code>STEAMPRODUCTNAME</code>	<p>Valve recommends that this be a build ID, although it is typical to use your AppID (as a string), or a light product name without any extra symbols.</p> <div>  <p>Changing this field, such as when updating your build ID, will cause Steam to ignore all active servers with the old name for matchmaking purposes, and will require you to modify your tool information in the partner backend.</p> </div>
<code>UE_PROJECT_STEAMGAMEDIR</code>	<code>STEAMGAMEDIR</code>	This is usually the folder name of your game, and cannot contain spaces or symbols. It is not required for this to be a folder name at all, as long as it is a name without spaces or symbols.
<code>UE_PROJECT_STEAMGAMEDESC</code>	<code>STEAMGAMEDESC</code>	Valve recommends that you set this as the full name of your product.
<code>UE_PROJECT_STEAMSHIPPINGID</code>	N/A	To launch outside of Steam in UE version 4.22 and higher, this macro must contain the product's Steam ID.



In UE version 4.22, you can specify these values in your game's `Target.cs` file. You can also write the Steam ID into a build by defining `UE_PROJECT_STEAMSHIPPINGID`. All build configurations require this, and you cannot launch outside of Steam without it.

For versions before 4.22, edit `OnlineSessionAsyncServerSteam.cpp` so that it contains your game's values.

For information about distributing a dedicated server build, please refer to [Steam's partner documentation](#).

Steam Online Authentication

Steam features a special authentication system that controls access to some of the unique server-related functionality the platform offers, such as advertising servers and player counts, retrieving and reacting to ban lists (both publisher and Valve Anti-Cheat bans), and performing license checks. Unreal Engine provides the ability to interface with this functionality through the `FOnlineAuthSteam` class. Once your application is set up to use the Steam Online Subsystem, it is ready to take advantage of these features by enabling the **SteamAuth** packet handler component.

Enabling SteamAuth

To enable SteamAuth, add the following to "DefaultEngine.ini", or to the platform-specific engine .ini file (such as "WindowsEngine.ini", "LinuxEngine.ini", or "MacEngine.ini") for each platform that is intended to support Steam Online Authentication:

```
1 [PacketHandlerComponents]
2 +Components=OnlineSubsystemSteam.SteamAuthComponentModuleInterface
```

 Copy full snippet

Once enabled, the Steam Online Subsystem interface (SteamOSS) function, `GetAuthInterface`, can be used to access SteamAuth functionality.



Enabling SteamAuth will cause the application to run authentication procedures (against the Steam service) for all joining players. By default, SteamAuth will kick players who fail this check, but this behavior can be overridden.

SteamAuth Delegates

There are two Delegates in the SteamAuth system that developers may wish to override:

`OverrideFailureDelegate` and `OnAuthenticationResultDelegate`.

`OverrideFailureDelegate` is called with the `FUniqueNetId` of a player when that player attempts to join the server without Steam authentication, or when that player loses Steam authentication during the session. By default, SteamAuth will kick the player from the game. However, if this Delegate is bound, the default behavior will be suspended, so developers will have to kick the player manually if that behavior is still desired.


`OnAuthenticationResultDelegate` handles responses from Steam's authentication service, providing the `FUniqueNetId` of the player and a boolean indicating whether the authentication attempt succeeded.

Additional Notes

Using IPNetDriver

By default, UE's Steam OSS uses Steam Networking as the default socket subsystem. In 4.22, you can disable this behavior by setting `OnlineSubsystemSteam.bUseSteamNetworking` to `false`. To do this, add the following to "DefaultEngine.ini", or to the platform-specific engine .ini file (such as "WindowsEngine.ini", "LinuxEngine.ini", or "MacEngine.ini") for each platform you support:

```
1 [OnlineSubsystemSteam]
2 bUseSteamNetworking=false
```

 Copy full snippet

In versions prior to 4.22, modify `SocketSubsystemSteam.cpp` where the `RegisterSocketSubsystem` function is called and change the `bool` parameter to be `false`. You will also need to change the netdrivers in your project configuration files.

Module Setup

Make sure to include the Unreal Engine Steam module as part of your project (see [Unreal Build Tool Target Files](#) for additional help). Specifically, adding the following line in the

constructor for `mygame.build.cs` should be enough to make sure that the Steam module is built along with your game.

```
DynamicallyLoadedModuleNames.Add("OnlineSubsystemSteam");
```

 Copy full snippet

Steam Overlay on Mac

The Steam Overlay on Mac requires launching your games through the Steam client. For this, you first need to add the game to your library using the "Add a Non-Steam Game to My Library" option from Steam's "Games" menu.