Upgrading the EOS SDK

Learn how to upgrade the EOS SDK for use in Unreal Engine.

Overview

The **Epic Online Services (EOS) Software Development Kit (SDK)** is a suite of tools that allow you to access EOS in your game. Multiple **Unreal Engine (UE) Plugins** depend on the EOS SDK to function, including:

- Online Subsystem EOS Plugin (OSS EOS)
- Online Services EOS Plugin (OSSv2 EOS)
- EOS Voice Chat Plugin

This page explains how to:

- Locate the EOS SDK files in the UE directory structure.
- Upgrade the EOS SDK.
- Find additional resources about the EOS SDK.

For more information about Epic Online Services, see the Additional Resources section of this page.

Location of the EOS SDK

The EOS SDK module is located in the following directory:

```
UNREAL_ENGINE_ROOT/Engine/Source/ThirdParty/EOSSDK
```

Copy full snippet

Plugins and modules can depend on the EOS SDK by adding this EOS SDK module as a dependency.

Modules Related to the EOS SDK Module

Several platform-specific extensions of this module exist, which contain the platform-specific EOS SDK, in the following directories:

```
UNREAL_ENGINE_ROOT/Engine/Platforms/<PLATFORM>/Source/ThirdParty/EOSSDK
```

Copy full snippet

The EOS Shared Plugin exposes common EOS functionality and is responsible for initialization and shutdown of the EOS SDK runtime library. The EOS Shared plugin is located in the following directory:

```
UNREAL_ENGINE_ROOT/Engine/Plugins/Online/EOSShared
```

Copy full snippet

Upgrade the EOS SDK

There are three methods for upgrading the EOS SDK:

Method	Description
Full Upgrade of Libraries and Headers	Choose this method if you wish to take advantage of new APIs and bug fixes in more recent SDK versions.

Method	Description
Libraries-only Upgrade	Choose this method if you wish to take advantage of bug fixes in more recent SDK versions, but not new APIs.
Project Override Upgrade	Choose this method if you wish to compile project modules against a different version of the EOS SDK than the one you compile engine modules against.

Full Upgrade of Libraries and Headers

This method upgrades the entire SDK, including libraries and header files. This allows you to rebuild the engine and plugins against the new headers, so you can use any new APIs and bug fixes available in recent SDK releases.

Steps

To perform a full upgrade of the EOS SDK:

1. Download your desired EOS SDK version from the EOS Developer Portal.
2. In the `Engine/Source/ThirdParty/EOSSDK/` directory, do the following:
   - Do not delete the `EOSSDK.Build.cs` or any other loose files located in this directory.
   - Delete all sub-directories of `Engine/Source/ThirdParty/EOSSDK/`.
3. Extract the `.zip` file for the new SDK into `Engine/Source/ThirdParty/EOSSDK/`. This creates a folder named `SDK` at the following location: `Engine/Source/ThirdParty/EOSSDK/SDK`.
4. Repeat the steps 1-3 for any additional platform-specific SDKs .
   - **For Android and iOS:** extract the `.zip` file into `Engine/Source/ThirdParty/EOSSDK/` over top of the base SDK.
   - **For other platforms:** extract the `.zip` files into their respective platform extensions in their respective directories `Engine/Platforms/<PLATFORM>/Source/ThirdParty/EOSSDK/`.
5. Move the file `Engine/Source/ThirdParty/EOSSDK/SDK/Bin/libEOSSDK-Mac-Shipping.dylib` to the following location `Engine/Binaries/ThirdParty/EOSSDK/Mac/libEOSSDK-Mac-Shipping.dylib`.

EOS SDK updates might include breaking or mandatory code changes. We advise you to consult the EOS SDK Release Notes for more information about required code changes.

API Version in EOS Engine Plugins

Many EOS engine plugins have migrated to setting a specific API version in EOS options structs rather than using `API_LATEST`.

Legacy Behavior

Previously, EOS engine plugins used `API_LATEST` as the version number in EOS options structs. As a result, when you upgraded the EOS SDK using the full upgrade method, and rebuilt the engine plugins, the SDK sometimes made breaking changes. These breaking changes occurred because the EOS SDK used the behavior contract defined in the new API version, but the engine plugins that used the EOS SDK expected the contract defined in the older API version from the headers the plugins were originally compiled against. It was then your responsibility to ensure that all the engine plugins were updated to use the new contract.

Current Behavior

Now, instead of `API_LATEST`, the API version specified in the headers the plugins were written against is used. This means that when you perform a full upgrade, engine code still uses the API version from the old headers. As a result, the EOS SDK continues to honor the contract the plugins were written against. This makes engine plugins operate as if you had performed a library-only upgrade.

The intention is to make engine plugins that use EOS less susceptible to breaking changes caused by SDK upgrades. Occasionally, the SDK might change the names of fields in options structs, or remove them altogether. When the SDK changes field names or removes them, compilation may fail with the new headers. If compilation fails, copy-paste the options struct definition from the old headers into the code so it can continue to use the old definition.

Example

Consider a case where the EOS SDK added APIs for setting the network state of the program. The default state in the new SDK became "Offline", a breaking change from the previous default of "Online". Because the engine plugins were passing `API_LATEST`, when the engine plugins were rebuilt against the new SDK, the SDK would be "Offline".

Because support for this new API is not present in UE, nothing is setting the network state to "Online". The SDK is left in the "Offline" state and becomes non-functional.

Had the old API version been passed (the value of `API_LATEST` in the headers the code was written against), the SDK would have known that it was being called by something written against headers pre-dating the addition of network state APIs, and would have continued working as if a DLL-only upgrade had been performed.

API Version Warnings

A common pattern in EOS plugins has previously been to add `static_asserts` next to usages of EOS options structs. These asserts fired when the value of `API_LATEST` for the options struct changed. This alerted you to the change and prompted you to take any required action.

Unfortunately, using asserts also meant that when you performed full (libraries and headers) upgrades of the EOS SDK, you might have encountered compilation failures when any such API version bump occurred. To fix this, we have replaced the `static_asserts` with a deprecation-like mechanism which instead emits warnings at compile time.

Silence Warnings

To silence API version warnings, add the following section to any of the engine configuration files, such as `DefaultEngine.ini`:

```
[EOSShared]
bEnableApiVersionWarnings=false;
```

Copy full snippet

Libraries-only Upgrade

This method involves upgrading only the library files, but not the header files. This enables you to take advantage of bug fixes in more recent SDK releases, or to swap your SDK for a version built against a different platform SDK version. This method does not require a rebuild of the engine or plugins since the headers have not changed. The SDK should continue to operate in the same manner as it did in the SDK version it was written against (the SDK version of the headers).

Steps

To perform a libraries-only upgrade of the EOS SDK:

1. Replace any of the following file types in their respective locations:
   - `.dll`
   - `.so`
   - `.dylib`
   - Other platform specific binary files in their respective locations.
2. Do the following depending on the platform you're using:
   - **For Mac:** note that the binary lives under `Engine/Binaries/ThirdParty/EOSSDK` unlike other platforms which all live under `Engine/Source/ThirdParty/EOSSDK` or `Engine/Platforms/<PLATFORM>/Source/ThirdParty/EOSSDK`. You just need to update the binary at `Engine/Binaries/ThirdParty/EOSSDK/Mac` in this case.
   - **For Android:** leave the `SDK/Bin/Android/Include` folder alone and update just the other subfolder or subfolders of `SDK/Bin/Android` containing `.aar` and `.lib` files.
   - **For IOS:** ignore the `SDK/Bin/IOS/EOSSDK.framework/Headers` folder and just update the other contents of the `SDK/Bin/IOS/EOSSDK.framework` folder.

Project Override

The project override method enables your project to provide its own EOS SDK. With this method, engine modules can continue to compile against the EOS SDK contained in `Engine/Source/ThirdParty/EOSSDK`, but project modules can choose to compile against SDK headers from either the engine or the project. Everything, including engine modules, is then run against the project provided EOS SDK binaries, because the EOS Shared Plugin prefers to load project-provided EOS SDK binaries at runtime instead of engine-provided ones.

Steps

To perform an EOS SDK project override:

1. Create your own `EOSSDK<PROJECT>` module that contains the EOS SDK for your project.
    - Make sure the `.Build.cs` file adds a `RuntimeDependencies` entry to copy your EOS SDK binary to `<PROJECT>/Binaries/<PLATFORM>`. Alternatively, you can check in the binary to `<PROJECT>/Binaries/<PLATFORM>` directly.
2. Set `EOSSDK_USE_PROJECT_BINARY=1` in your project's `.Target.cs` files.
    - This ensures monolithic builds do not link against the Engine EOS SDK binaries – only your project provided ones.
3. For project modules, choose whether to add the EOS SDK (the engine module) or your project's EOS SDK override module as a dependency.
    - This specifies which set of headers your project modules compile against.

Product Name

You can set the EOS SDK product name in engine configuration files, such as your project's `DefaultEngine.ini`. The EOS SDK product name is the name that appears in the Presence information in the EOS overlay. If the EOS SDK product name is not set in engine configuration, it defaults to your project name.

Set EOS SDK Product Name

To set your product name, add the following in your project's `DefaultEngine.ini`:

```
[EOSSDK]
ProductName="<PRODUCT_NAME>"
```

Copy full snippet

where `<PRODUCT_NAME>` is the name of your product. For example, to add "MyProduct" as the name of your product, add the following to your project's `DefaultEngine.ini`:

```
[EOSSDK]
ProductName="MyProduct"
```

Copy full snippet

Additional Resources

The EOS SDK and Unreal Engine are separate products. Unreal Engine contains a build of the EOS SDK for convenience, but the latest version of Unreal Engine might not include the latest version of the EOS SDK. For more information about the EOS SDK, see the following Epic Developer Resources:

- Getting Started with EOS: Steps to set up an Epic Games Account and Organization, as well as directions for downloading the EOS SDK and requesting access to the EOS SDK for consoles.
- EOS SDK Key Information: Fundamental EOS SDK information including the various SDK downloads available, naming conventions, error handling, and more.
- EOS SDK API Reference: Full API reference for the EOS SDK.
- EOS SDK Release Notes: Release notes for each EOS SDK release including new APIs and bug fixes.