GeForce NOW

Software Development Kit Unreal Engine Integration Guide

Document History

Version	Date	Description of Change
1.0	*05/15/2024*	*Initial release revision*

Introduction 4
Audience 4
SDK Overview 4
Unreal Engine Overview 5
1. Existing GFN SDK Integration in UE5 5
2. Interface Design 5
3. Sample API Calls 8
4. Building the UE Module 10
5. Making Calls to the SDK Wrapper Interface 11
6. Conclusion and Next Steps 12

Introduction

NVIDIA GeForce NOW (GFN) allows users to enjoy their library of PC games on many internet-enabled devices via the power of cloud gaming. This includes PCs, Macs, Smart TVs, and mobile devices such as smartphones and tablets.

GFN has the ability to onboard your Windows and Linux games and applications as-is for your users to enjoy. For tighter integration of these applications with GFN, NVIDIA provides the GeForce NOW Runtime Software Development Kit (GFN SDK) to developers so that their applications can perform various tasks; from knowing when the application is running in GFN to getting information about the GFN session or the user's client system.

This document is provided as part of the SDK to aid in getting the developer up to speed with the SDK quickly, as well as providing source code pointers for the most common of integration scenarios.

Audience

This document is directed towards developers using Epic's Unreal Engine for their applications and games, and that wish to integrate those titles into the GFN ecosystem to provide their users a seamless experience between their application and GFN.

For the most complete understanding of the SDK, please first read the SDK's Quick Start Guide found in ./doc folder of the SDK package.

SDK Overview

The GFN Runtime SDK consists of an ever-growing collection of C-based native APIs that support both Windows and Linux applications. Along with the C source files that define the APIs, the SDK also includes various documentation to aid in development, as well as a collection of samples in source code form that provide example execution of the APIs in various scenarios.

The SDK is designed with ease of use in mind, with API design meant to allow function calls and data consumption to be straightforward. In addition, the SDK is designed to support backwards compatibility as a core requirement, which allows developers to move to new versions of the SDK without the worry of code breakage related to existing API calls due to API definition alterations or deprecation.

Unreal Engine Overview

Unreal Engine (UE) is a source engine developed by Epic Games that can be licensed by developers for their title development. It provides a multitude of features that allow developers the ability to bring their titles to market much faster than developing their own baseline source engine. Epic Games has made the source code publicly available at https://github.com/EpicGames/UnrealEngine.

NVIDIA and Epic Games have partnered together for a number of years for various features, including use of the GFN SDK in titles built on the Unreal Engine.

For the scope of this document, discussion will be focused on Unreal Engine 5, commonly known as UE5.

1. Existing GFN SDK Integration in UE5

Epic Games has integrated the GFN SDK into current versions of both UE4 and UE5 to provide themselves an easy integration path for their own titles, such as Fortnite.
This integration can be found via the **IGeForceNOWWrapperModule** interface defined in IGeForceNOWWrapperModule.h header.

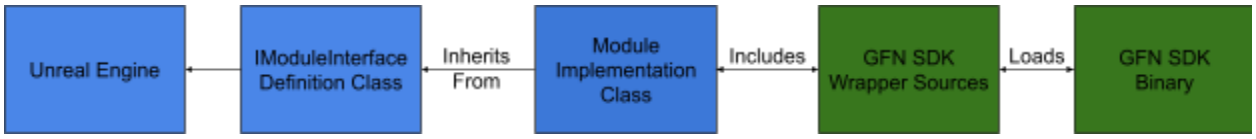
However, this interface wrapper does not provide a complete integration of all GFN SDK functionality. Epic Games has stated that the interface was designed for their use, and not public reuse. Therefore, it is best for 3rd party developers to create their own integration interface instead.

This document will focus on the steps necessary to create that interface, and for simplicity, will focus on that interface in the scope of UE5.

2. Interface Design

For 3rd party code integrations into UE titles, UE relies on the concept of “modules”. If developers are not familiar with this concept, they should first read https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-modules.

While the existing GFN SDK integration in the UE5 source code should not be used directly by developers, its overall design can be leveraged for a developer’s own integration as it follows the design for UE Module. This existing GFN SDK module design is visualized in the following diagram:



The first layer of a module is defined by inheriting from the UE's IModuleInterface via an interface class. Detailed information on the IModuleInterface can be found as part of UE's developer documentation. For the purpose of GFN SDK integration, this class can be defined as:

```
C/C++
#include "Modules/ModuleInterface.h"

/**
 * Interface for the GeForce NOW SDK Module.
 */
class IGFNSDKWrapperModule : public IModuleInterface
{
};
```

With that interface class defined, the implementation class can be created that inherits from it. For example, using Epic's naming standards:

```
C/C++
class FGFNSDKWrapperModule : public IGFNSDKWrapperModule
{
public:
	virtual void StartupModule() override
	{
		const GfnError GfnResult = GeForceNOWWrapper::Initialize();
		if (GFN_Succeeded(GfnResult)
		{
			// GFN SDK successfully initialized
			// any other SDK API calls to be run on startup goes here
		}
		else
		{
```

```
                    // Always log failures for easier debugging!
                    UE_LOG(LogGFN, Log, TEXT("GFNSDK init failed: %d"),
        GfnResult);
                }
        }
        virtual void ShutdownModule() override
        {
        // Always shutdown the SDK on module shutdown for proper cleanup
        GeForceNOWWrapper::Shutdown();
        }
};
```

Finally, do not forget to declare the module at the end:

```
C/C++
// Expose the module to the project
IMPLEMENT_MODULE(FGFNSDKWrapperModule, GFNSDKWrapper);
```

Once this module definition class is created, then the Wrapper class can be defined as well. This class's role is to expose module functions to the engine and bridge them to the GFN SDK APIs.

As the GFN SDK is designed to have a single invocation for the lifetime of the calling process, the Wrapper class should make use of the singleton design pattern. The singleton skeleton can be setup as follows:

```
C/C++
/**
 * Singleton wrapper class for calling GeForce NOW SDK APIs
 */
class GFNSDKWRAPPER_API GFNSDKWrapper
{
private:

        /** Singleton accessors and destructors. */
        GFNWrapper(){}
```

```
        ~GFNWrapper() {};

        static GFNSDKWrapper* Singleton;

public:
        /** Loads and Initializes the GFN SDK */
        static GfnError Initialize();

        /** Unload the GFNSDK */
        static GfnError Shutdown();

        /** Allows access to pointer to singleton object */
        static GFNSDKWrapper& Get();
};

/**
 * Global singleton class object
 */
GFNSDKNWrapper* GFNSDKWrapper::Singleton = nullptr;
```

With the singleton wrapper class defined, the rest of the work is to define public class functions that wrap the GFN SDK APIs that will be used. These calls can either be:

1. Manually loading the SDK binary and directly from the binary exports
2. Using the GFN SDK's Wrapper sources (GfnRuntimeSdk_Wrapper.*)

The second option is preferred as the wrappers handle signature checking of the SDK binary to prevent tampering or spoofing, as well as binary lifetime handling for you.

For specific information on how either method is used to call SDK APIs, please refer to the SDK's Quick Start Guide in the ./doc folder of the GFN SDK's image, or the source code of the samples included with the GFN SDK. Understanding the SDK's APIs and their general calling patterns will help developers better understand their integration into UE.

3. Sample API Calls

This section of the document provides code snippets for how the singleton Wrapper class can make calls into the SDK. As stated above, the preferred method is to use the GFN SDK's Wrapper sources for the benefits of SDK binary lifetime management, and so the code examples provided in this section will leverage those sources.

For brevity, this section will not continue snippets for every SDK API function, but will continue example code for the most used API functions as well as provide enough information to extend to any API function not covered in this section.

Initializing the SDK

```
C/C++
GfnError GFNSDKWrapper::Initialize() {
	if (!Singleton) {
		Singleton = new GFNSDKWrapper();
	}
	// Build path to the GFN SDK DLL
	FString GFNDllPath = FPaths::Combine(FPaths::EngineDir(),
TEXT("Binaries/ThirdParty/NVIDIA/GeForceNOW"),
FPlatformProcess::GetBinariesSubdirectory(), TEXT("GfnRuntimeSdk.dll"));
	FString GFNDllFullPath =
IFileManager::Get().ConvertToAbsolutePathForExternalAppForRead(*GFNDllPath);
	GFNDllFullPath.ReplaceInline(TEXT("/"), TEXT("\\"),
ESearchCase::CaseSensitive);
	const GfnError ErrorCode = GfnInitializeSdkFromPathW(gfnDefaultLanguage,
*GFNDllFullPath);
	return ErrorCode;
}
```

Note that the paths above are examples based on Epic's existing integration packaging, and should be changed to match paths used in your build.

Basic Query if Running GeForceNOW Ecosystem

```
C/C++
bool GFNSDKWrapper::IsRunningInGFN(){
	bool bLocalIsRunningInCloud = false;
	GfnIsRunningInCloud(&bLocalIsRunningInCloud);
	return bLocalIsRunningInCloud;
}
```

Registering for a Callback

```
C/C++
GfnError GFNSDKWrapper::RegisterClientInfoCallback(ClientInfoCallbackSig
ClientInfoCallback, void* Context) const {
	return GfnRegisterClientInfoCallback(ClientInfoCallback, Context);
}
```

If there are more than one SDK callbacks that will be used, there will be a Singleton Wrapper function and callback function defined for each one.

Handling a Callback

```
C/C++
GfnError GFNSDKWrapper::ClientInfoCallback(ClientInfoCallbackSig
ClientInfoCallback, void* Context) const {
	// Read client info here as needed
	return GfnSuccess;
}
```

Shutting Down the SDK

```
C/C++
GfnError GFNSDKWrapper::Shutdown(){
	if (!!Singleton) {
		// destroy the singleton object in case a new one is to be created
		delete Singleton;
		Singleton = nullptr;
	}
	return GfnShutdownSdk();
}
```

4. Building the UE Module

The instructions for build definition and building the new GFN SDK module are defined as part of the UE documentation at: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-modules. Overall, your project will include a Build.cs with some basic information. From the perspective of the GFN SDK, this CS file will include a link to the GFN SDK sources. For example:

```
Unset
using UnrealBuildTool;
using System;

public class GFNSDKWrapper : ModuleRules
{
	public GFNSDKWrapper (ReadOnlyTargetRules Target) : base(Target)
	{
		PublicSystemIncludePaths.Add(
			Target.UEThirdPartySourceDirectory +
"NVIDIA/GFNSDK/include"
		);

		PublicDependencyModuleNames.AddRange(
			new string[] {
				"GFNSDK",
				"Slate"
			}
		);

		PrivateDependencyModuleNames.AddRange(
			new string[] {
				"Core",
				"SlateCore"
			}
		);
	}
}
```

Note that this path will need to be tailored to where you include the GFN SDK's sources in your overall UE build.

5. Making Calls to the SDK Wrapper Interface

Now that the GFN SDK Wrapper module has been created, and can make calls into the SDK's functions, the final step is to make a function call from the engine into the GFN SDK Wrapper Module.

There are several ways to do this, depending on if the calls will be from other dependency modules, or from blueprints. In order to not duplicate Unreal Engine documentation that is

subject to change each release, it is preferred the developer refer to the Unreal Engine documentation starting with https://dev.epicgames.com/documentation/en-us/unreal-engine/programming-with-cplusplus-in-unreal-engine.

6. Conclusion and Next Steps

This document is meant to provide an overview of the GFN SDK's use in UE5, and provide integration paths for most-common GFN SDK API methods. The code samples provided here are meant to provide enough code to get started, and are not meant to be complete drop-in code for every type of use case. For the sake of brevity, not all error cases are handled by the sample code provided, nor is everything logged, whereas production-level code must do so.

For complete information about the APIs, refer to the API documentation in the ./doc folder, as well as the working code supplied with the samples in ./samples. If there are still questions about the SDK, then please email the SDK development team at geforcenow-sdk-support@nvidia.com.