Peter Gustafson writing sample
Epic Games
Feb 21, 2021

[Getting started with the C# SDK](https://dev.epicgames.com/docs/services/en-US/CSharp/GettingStarted/index.html)

`C#` SDK Introduction
Epic Online Services (EOS) offers a `C#` SDK in addition to the `C` SDK.

While the core functionality remains the same regardless of which SDK you use, the `C#` SDK differs in design in the following ways:

- The `C#` SDK adheres to `C#` best practices and follows an object-oriented approach such as using handle objects rather than C-style handles to manage asynchronous operations.

- Naming conventions match typical `C#` patterns. For example, the`EOS_Auth_Login` in the `C` SDK is accessible in the `C#` SDK as`Epic.OnlineServices.Auth.AuthInterface.Login`.

- Data structures in the `C` SDK require macro-based API version numbers to ensure compatibility. These values are pre-populated in the `C#` SDK.
- At this time, the `C#` SDK supports only desktop platforms.

Download the `C#` SDK

 1. Download the C# SDK[LINK TO C# SDK DOWNLOAD PAGE].
 2. Set-up a product on Epic's [Developer Portal](https://dev.epicgames.com/docs/services/en-US/DevPortal/index.html).
 3. Create values for the following fields in the [Platform Interface](https://dev.epicgames.com/docs/services/en-US/Interfaces/Platform/index.html) (this is required to access all of the [EOS SDK interfaces](https://dev.epicgames.com/docs/services/en-US/Interfaces/index.html)).

- `Product ID`

- `Sandbox ID`

- `Deployment ID`

- `Client ID`

- `Client secret`

General Integration
1. Include the `C#` SDK source files in your project.

2. Select the appropriate library binary file for your target platform.

3. For example, a Windows x64 integration uses `EOSSDK-Win64-Shipping.dll`.

4. The `C#` SDK determines which library binary name to target in `Epic.OnlineServices.Config`. Several platforms have been preconfigured to point to the correct name.

5. If your target platform has not been preconfigured, update the configuration and your project symbols to avoid a build error.

6. For example, set `PLATFORM_64BITS` if targeting Windows x64, or `PLATFORM_32BITS` if targeting Windows x86.

Unity Integration

1. Copy the `C#` SDK source files into the `Assets` folder. Include both the `Core` and `Generated` folders.

2. Copy the library binary file into the `Assets` folder for your target platform. For example, projects targeting Windows x64 would include `EOSSDK-Win64-Shipping.dll`.

3. The `C#` SDK determines which library binary name to target in `Epic.OnlineServices.Config`.

4. If your target platform has not been preconfigured, make the appropriate changes to the configuration and your project symbols to avoid build errors.

5. Create a new script to control the SDK. The screenshot below uses the name `EOSSDKComponent`.

 ![New script Component](https://dev.epicgames.com/docs/services/Images/CSharp/GettingStarted/Unity-2.jpg)

6. Add the component to an entity so it's available when the game needs it. The screenshot below uses the [Main Camera](https://docs.unity3d.com/ScriptReference/Camera-main.html).

 ![Add Component](https://dev.epicgames.com/docs/services/Images/CSharp/GettingStarted/Unity-3.jpg)

7. Write your EOS SDK code in this component. View the [sample code](https://dev.epicgames.com/docs/services/en-US/CSharp/GettingStarted/

index.html#samplecode).

Implementing the SDK

After setting up your product(s) and the `C#` SDK, begin writing your code.

Don't forget to initialize the SDK, by making a call to its `Tick` method regularly so the SDK can execute and callbacks can run. Remember to shut it down when your application finishes.

Editor Environments
Because these editor environments use one continuous instance of the EOS SDK, operations that begin right before the end of a play session may finish and trigger callbacks in the following session.

Managing the SDK's Lifecycle
There are three main functions of the SDK's lifecycle:

 - Initialization
 - Ticking
 - Shutdown

The [Platform Interface](https://dev.epicgames.com/docs/services/en-US/Interfaces/Platform/index.html) is the entry point to the SDK. You need an instance of this interface to use EOS.

Initialization

 - Initialize by making a call to `Epic.OnlineServices.Platform.PlatformInterface.Create` with the values you have obtained from the Developer Portal to get an `Epic.OnlineServices.Platform.PlatformInterface` instance. Store this instance; you need it to interact with the SDK.

 - You can only initialize the SDK once. Subsequent calls to `Epic.OnlineServices.Platform.PlatformInterface.Initialize` will return the `AlreadyInitialized` error code.

 - We recommend initializing the SDK when your application starts up. Don't release it until the application shuts down.

Ticking
Ticking is the normal operation of your game.

 - In order for the SDK to operate, make a call to `Epic.OnlineServices.Platform.PlatformInterface.Tick` on your Platform Interface instance regularly.

 - These calls don't need to happen in every frame, but should occur often. One call every 100 milliseconds is good. Feel free to adjust the ticking frequency to your gaming needs.

 - SDK callbacks can only run when you call `Tick`.

Shutdown

Shutdown the EOS SDK by making a call to `Epic.OnlineServices.Platform.PlatformInterface.Release` to release your Platform Interface instance, and then `Epic.OnlineServices.Platform.PlatformInterface.Shutdown` to complete the shutdown process.

 - This process is final and puts the SDK into a finalized state.

 - Some editor environments, like Unity, load external libraries during the editor start up. In this cases, don't make a call to `Epic.OnlineServices.Platform.PlatformInterface.Release` or to`Epic.OnlineServices.Platform.PlatformInterface.Shutdown` at the end of an in-editor play session. You won't be able to initialize the SDK successfully in any future sessions without restarting the editor.

Problems setting up your account? Visit the [**developer help forum**](https://www.epicgames.com/help/en-US/).

Results

Most callback data structures (and some return values) use `Epic.OnlineServices.Result` to convey the results of SDK operations. Make sure to use this to handle errors so operations perform as expected.

Logging

The SDK outputs useful debugging information through an internal interface.

 1. Enable logging by setting up your `Epic.OnlineServices.Logging.LoggingInterface`.

 2. Be smart and set-up logging as early as possible. Preferably after initializing the SDK.

 3. Make a call to `Epic.OnlineServices.Logging.LoggingInterface.SetLogLevel` with the parameters for your level of detail.
 4. Next, make a call to `Epic.OnlineServices.Logging.LoggingInterface.SetCallback` with a callback function to receive log information.

 5. Once enabled, logging provides insight into internal operations and helps identify the causes of unexpected behaviors.

Unmanaged Memory

 - The SDK uses objects with an underlying type of `Epic.OnlineServices.Handle`.

 - In some cases, these objects act as accessors to unmanaged data in the SDK cache, with application-controlled lifetimes.

 - These objects have `Release` methods that you must call to prevent memory leaks.

 - For example, the `Epic.OnlineServices.Presence.PresenceModification` object that you get from `Epic.OnlineServices.Presence.PresenceInterface.CreatePresenceModification` has a `Release` method that you must call when you no longer need the object.

Threading

The C# SDK is not currently thread safe. All calls made to the SDK must come from your application's main thread. At this time, we recommend against using `async`, `await`, `Thread`, `Task`, or similar patterns.

Sample Projects

The `C#` SDK ships with sample projects that demonstrate various features and can help you get started faster.

Sample code

The sample code below helps familiarize you with the `C#` SDK.

Setting up the platform

```
// Set these values as appropriate. For more information, see the Developer Portal documentation.
string productName = "MyApplication";
string productVersion = "1.0";
string productId = "";
string sandboxId = "";
string deploymentId = "";
string clientId = "";
string clientSecret = "";
```

```
var initializeOptions = new
Epic.OnlineServices.Platform.InitializeOptions()
{
    ProductName = productName,
    ProductVersion = productVersion
};

var initializeResult =
Epic.OnlineServices.Platform.PlatformInterface.Initialize(initializeOp
tions);
if (initializeResult != Epic.OnlineServices.Result.Success)
{
    throw new System.Exception("Failed to initialize platform: " +
initializeResult);
}

// The SDK outputs lots of information that is useful for debugging.
// Make sure to set up the logging interface as early as possible:
after initializing.
Epic.OnlineServices.Logging.LoggingInterface.SetLogLevel(Epic.OnlineSe
rvices.Logging.LogCategory.AllCategories, LogLevel.VeryVerbose);
Epic.OnlineServices.Logging.LoggingInterface.SetCallback((Epic.OnlineS
ervices.Logging.LogMessage logMessage) =>
{
    Console.WriteLine(logMessage.Message);
});

var options = new Options()
{
    ProductId = productId,
    SandboxId = sandboxId,
    DeploymentId = deploymentId,
    ClientCredentials = new ClientCredentials()
    {
        ClientId = clientId,
        ClientSecret = clientSecret
    }
};

var platformInterface = PlatformInterface.Create(options);
if (platformInterface == null)
{
    throw new System.Exception("Failed to create platform");
}
```

Login

```
var loginCredentialType =
```

```
Epic.OnlineServices.Auth.LoginCredentialType.AccountPortal;
/// These fields correspond to <see
cref="Epic.OnlineServices.Auth.Credentials.Id" /> and <see
cref="Epic.OnlineServices.Auth.Credentials.Token" />,
/// and their use differs based on the login type. For more
information, see <see cref="Epic.OnlineServices.Auth.Credentials" />
/// and the Auth Interface documentation.
var loginCredentialId = null;
var loginCredentialToken = null;

var authInterface = platformInterface.GetAuthInterface();
if (authInterface == null)
{
    throw new Exception("Failed to get auth interface");
}

var loginOptions = new Epic.OnlineServices.Auth.LoginOptions()
{
    Credentials = new Epic.OnlineServices.Auth.Credentials()
    {
        Type = loginCredentialType,
        Id = loginCredentialId,
        Token = loginCredentialToken
    }
};

// Ensure platform tick is called on an interval, or this will not
callback.
authInterface.Login(loginOptions, null,
(Epic.OnlineServices.Auth.LoginCallbackInfo loginCallbackInfo) =>
{
    if (loginCallbackInfo.ResultCode ==
Epic.OnlineServices.Result.Success)
    {
        Console.WriteLine("Login succeeded");
    }
    else
    {
        Console.WriteLine("Login returned " +
loginCallbackInfo.ResultCode);
    }
});
```

Unity EOS SDK Component

```
// This code is provided for demonstration purposes and is not
intended to represent ideal Unity practices.
using Epic.OnlineServices;
```

```
using Epic.OnlineServices.Auth;
using Epic.OnlineServices.Logging;
using Epic.OnlineServices.Platform;
using UnityEngine;

public class EOSSDKComponent : MonoBehaviour
{
    // Set these values as appropriate. For more information, see the
Developer Portal documentation.
    public string m_ProductName = "MyApplication";
    public string m_ProductVersion = "1.0";
    public string m_ProductId = "";
    public string m_SandboxId = "";
    public string m_DeploymentId = "";
    public string m_ClientId = "";
    public string m_ClientSecret = "";

    public LoginCredentialType m_LoginCredentialType =
LoginCredentialType.AccountPortal;
    /// These fields correspond to <see cref="Credentials.Id" /> and
<see cref="Credentials.Token" />,
    /// and their use differs based on the login type. For more
information, see <see cref="Credentials" />
    /// and the Auth Interface documentation.
    public string m_LoginCredentialId = null;
    public string m_LoginCredentialToken = null;

    private static PlatformInterface s_PlatformInterface;
    private const float c_PlatformTickInterval = 0.1f;
    private float m_PlatformTickTimer = 0f;

    void Start()
    {
        var initializeOptions = new InitializeOptions()
        {
            ProductName = m_ProductName,
            ProductVersion = m_ProductVersion
        };

        var initializeResult = PlatformI
```

Need help? Visit the [**developer help forum**](https://
www.epicgames.com/help/en-US/).