# EOS Tech Writer Coding Exercise - Implementing EOS Achievements

**Written By James Ryan**

## Prerequisites (Copied From Exercise Writeup)

Before you start, make sure you have the following:

- The project you are going to work on, downloaded and unzipped.
    - Download it from Box.com at epicgames.box.com/~.
- A Windows PC with Visual Studio 2022 (visualstudio.microsoft.com) installed.
- An Epic Games account.
    - You set up an Epic Games account in the Epic Games Developer Portal (dev.epicgames.com/portal).
    - As part of this you must set up an organization in the Epic Games Developer Portal. The organization can be whatever name you choose. You do not need to configure the organization.
    - If you already have an Epic Games account, you can reuse it, you do not need to set up a new one.
- A product in the Epic Games Developer Portal (dev.epicgames.com/portal).
    - You do not need to configure the product.
    - You will need the various IDs associated with your product and organization for EOS SDK configuration.
    - These IDs are available from your product's Product Settings page in the Developer Portal. (Documentation: Product Management.)
- The EOS SDK in C downloaded from the Epic Games Developer Portal (dev.epicgames.com/portal).
    - The project you are going to work on uses EOS SDK v1.16.4.

## I. Create A Means To Tick The EOS Platform Using The EOS Platform Interface

### Step 1: Initialize The Project

- Open the solutions file, ***StubGame.sln***, in Visual Studio 2022. This will open the ***PlatformHandler.cpp*** file.
- Press "run" at the top to generate the different directories of the project. This returns an error that can be dealt with by copying and pasting a file from one directory to another, as outlined in the writeup.
    - Copy the file ***EOSSDK-Win64-Shipping.dll*** from ***StubGame/StubGame/sdk/Bin*** into the ***StubGame/x64/Debug*** directory.
- Press "run" at the top again and see that abort() is being called in line 35.

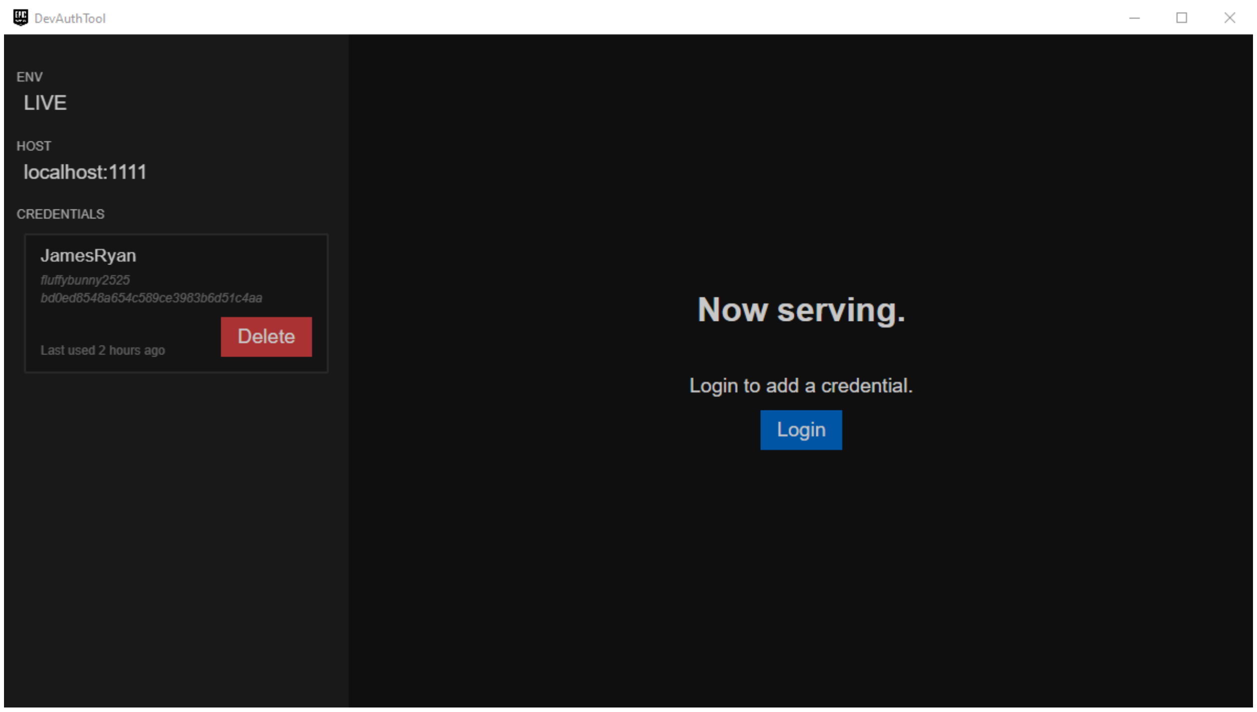
    - This line is an assert statement to see that the EOS has successfully been initialized. An abort() call suggests that it hasn't been properly set up yet. There is still work to be done here!

## Step 2: Edit And Personalize The *SDKConfig.h* File

- Referencing the Platform Interface documentation, ensure that all required components are defined in the solutions file.
    - This is where the product set up earlier through the Epic Games Developer Portal (in the prerequisites section) comes in handy!
- Fill in the ***SDKConfig.h*** file (***StubGame/StubGame/SDKConfig.h***) with the information from the Dev portal.
    - Note: This is the only instance of actually editing code from the project files in this section, everything else for this section is on the online Dev portal.
- ***"ProductId"***, "***SandboxId"***, and "***DeploymentId"*** are all found on the Product Settings page and can be directly entered into the corresponding sections of the ***SDKConfig.h*** file as is.
- Create a Client for the "***ClientId"*** with a policy type of ***"GameClient /w UnlockAchievements"***
- On the Product Settings page, click **"*Use credentials in header file"*** with ***"LiveDeployment"*** for Deployment and the created client for Client.
    - Update the ***SDKConfig.h*** header file with the values returned here, specifically ***"ClientCredentialsId"*** and ***"ClientCredentialsSecret"***.
    - Save the changes to the ***SDKConfig.h*** file now that it has been filled out with more specificity.
- Now the only empty fields in this header file are ***"CredentialName[]"*** and ***"Port[]"***, both of which come from the DevAuthTool.

### Step 2.1 - Set Up the DevAuthTool

- Run the DevAuthTool found in the SDK download (from the prerequisites section), under ***SDK\Tools\EOS_DevAuthTool-win32-x64-1.2.1*** (this is what I opened on my Windows machine while running this - you may need to open a different folder based on your hardware and OS).
- Unzip the folder then run the executable file housed within.
- Sign in with Epic Games and copy this information to the ***"CredentialName[]"*** and ***"Port[]"*** fields in the ***SDKConfig.h*** file, respectively.
    - ***"CredentialName"*** is whatever you named your sign-in, and ***"Port"*** is ***"localhost:####"*** where the numbers are the port you chose.
    - Below is an example of my environment after setting up DevAuthTool for port 1111:



ENV
LIVE

HOST
localhost:1111

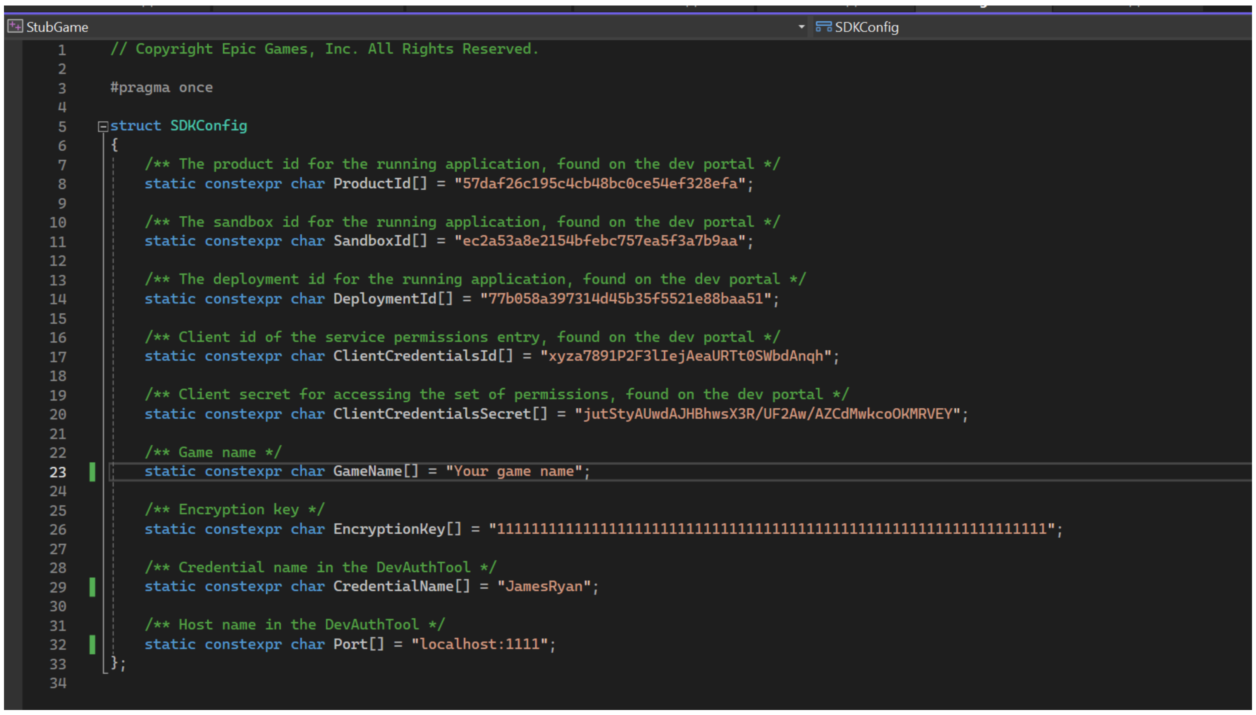
CREDENTIALS

JamesRyan
*fluffybunny2525*
*bd0ed8548a654c589ce3983b6d51c4aa*
Last used 2 hours ago

**Now serving.**

Login to add a credential.

- One more image for you: below is an example of my ***SDKConfig.h*** file after filling in the required information.



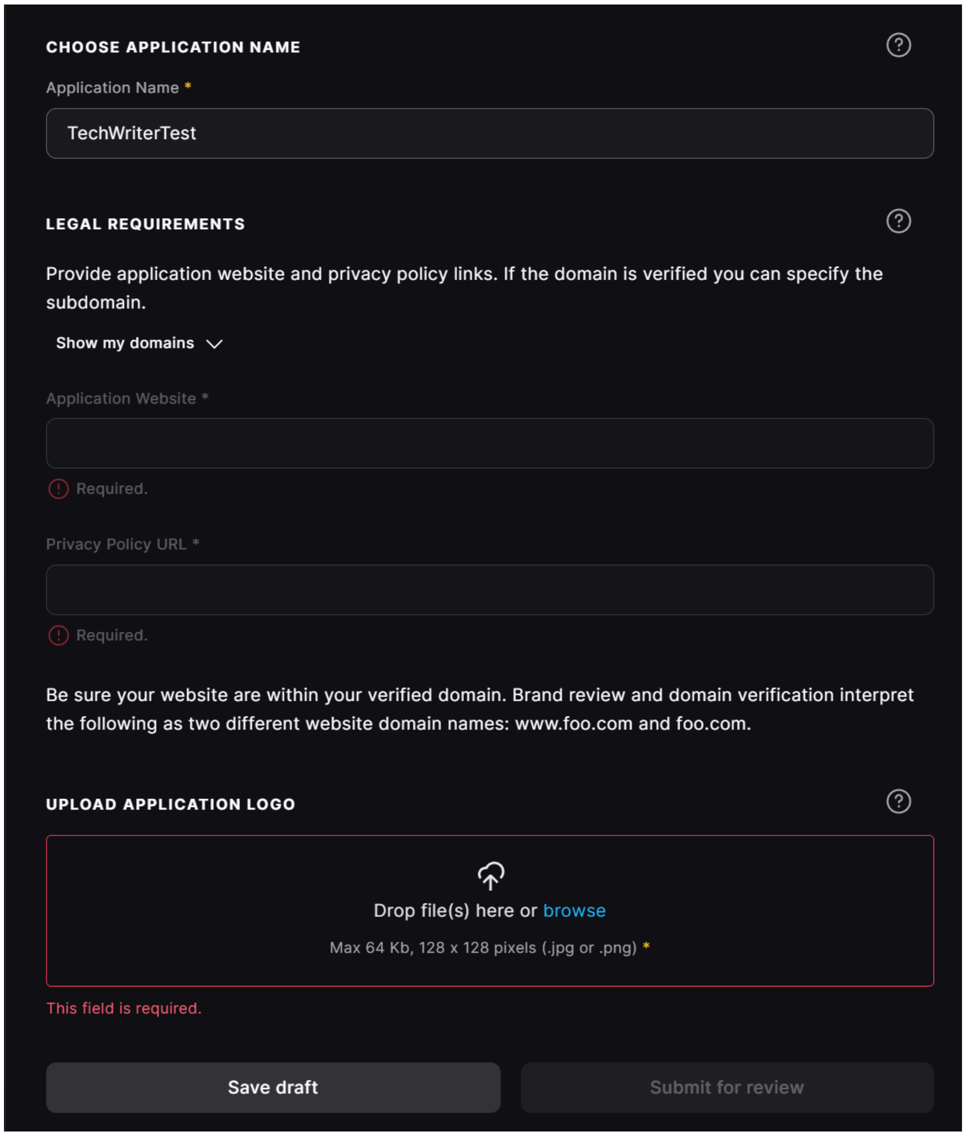
```cpp
// Copyright Epic Games, Inc. All Rights Reserved.

#pragma once

struct SDKConfig
{
    /** The product id for the running application, found on the dev portal */
    static constexpr char ProductId[] = "57daf26c195c4cb48bc0ce54ef328efa";

    /** The sandbox id for the running application, found on the dev portal */
    static constexpr char SandboxId[] = "ec2a53a8e2154bfebc757ea5f3a7b9aa";

    /** The deployment id for the running application, found on the dev portal */
    static constexpr char DeploymentId[] = "77b058a397314d45b35f5521e88baa51";

    /** Client id of the service permissions entry, found on the dev portal */
    static constexpr char ClientCredentialsId[] = "xyza7891P2F3lIejAeaURTt0SWbdAnqh";

    /** Client secret for accessing the set of permissions, found on the dev portal */
    static constexpr char ClientCredentialsSecret[] = "jutStyAUwdAJHBhwsX3R/UF2Aw/AZCdMwkcoOKMRVEY";

    /** Game name */
    static constexpr char GameName[] = "Your game name";

    /** Encryption key */
    static constexpr char EncryptionKey[] = "1111111111111111111111111111111111111111111111111111111111111111";

    /** Credential name in the DevAuthTool */
    static constexpr char CredentialName[] = "JamesRyan";

    /** Host name in the DevAuthTool */
    static constexpr char Port[] = "localhost:1111";
};
```

## Step 3: Set Up An Application On The Dev Site

- We are now receiving a new error: "The Client is not configured correctly. Please make sure it is associated with an Application."
  - This suggests an Application has not been properly linked. In our case, that would be because no application has been set up through the Dev site. Let's set ours up now.
- On the Epic Games Dev organization page, select ***"Epic Account Services"*** on the left. Click ***"New Application"*** and you will be greeted with three "not configured" text boxes: ***"Brand Settings"***, ***"Permissions"***, and ***"Linked Clients"***. Click the first of these.

### Step 3.1 - Brand Settings

- Now we need to fill in the three empty fields on this new page: ***"Application Website"***, ***"Privacy Policy URL"***, and ***"Upload Application Logo"***. Below is an image of that screen looks like:



CHOOSE APPLICATION NAME

Application Name *

TechWriterTest

LEGAL REQUIREMENTS

Provide application website and privacy policy links. If the domain is verified you can specify the subdomain.

Show my domains

Application Website *

Required.

Privacy Policy URL *

Required.

Be sure your website are within your verified domain. Brand review and domain verification interpret the following as two different website domain names: www.foo.com and foo.com.

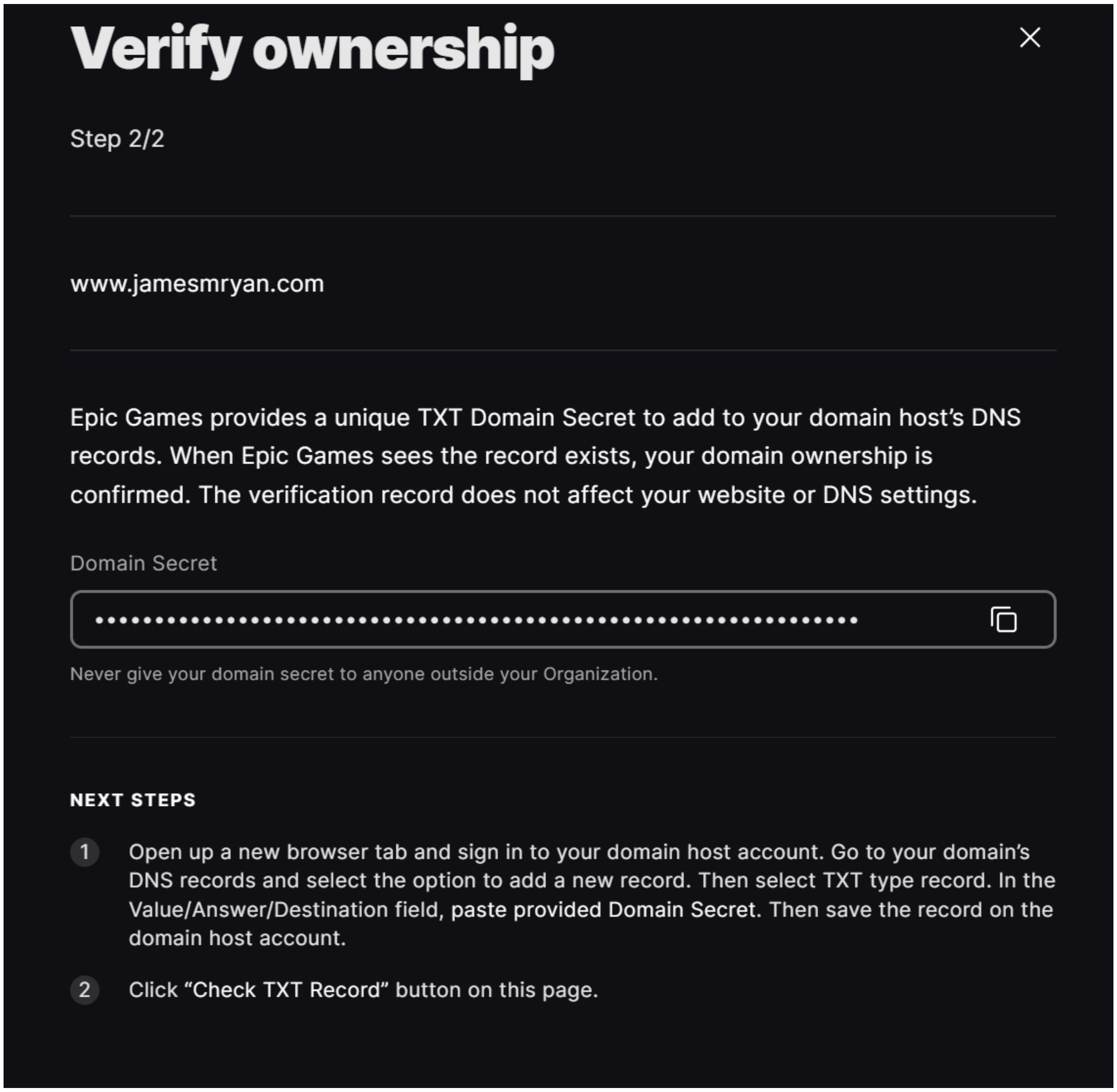
UPLOAD APPLICATION LOGO

Drop file(s) here or browse

Max 64 Kb, 128 x 128 pixels (.jpg or .png) *

This field is required.

    - Hovering over the first of these fields, ***“Application Website”***, will prompt a popup to appear with a hyperlink in it to ***“Organization Management”***. Click on this hyperlink.
- Type a website address in the entry field and click “Next”. A Domain Secret will appear (albeit blurred out) along with some instructions about signing into your domain host’s account. Below is an image of this screen.



Verify ownership

Step 2/2

www.jamesmryan.com

Epic Games provides a unique TXT Domain Secret to add to your domain host’s DNS records. When Epic Games sees the record exists, your domain ownership is confirmed. The verification record does not affect your website or DNS settings.

Domain Secret

••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••

Never give your domain secret to anyone outside your Organization.

NEXT STEPS

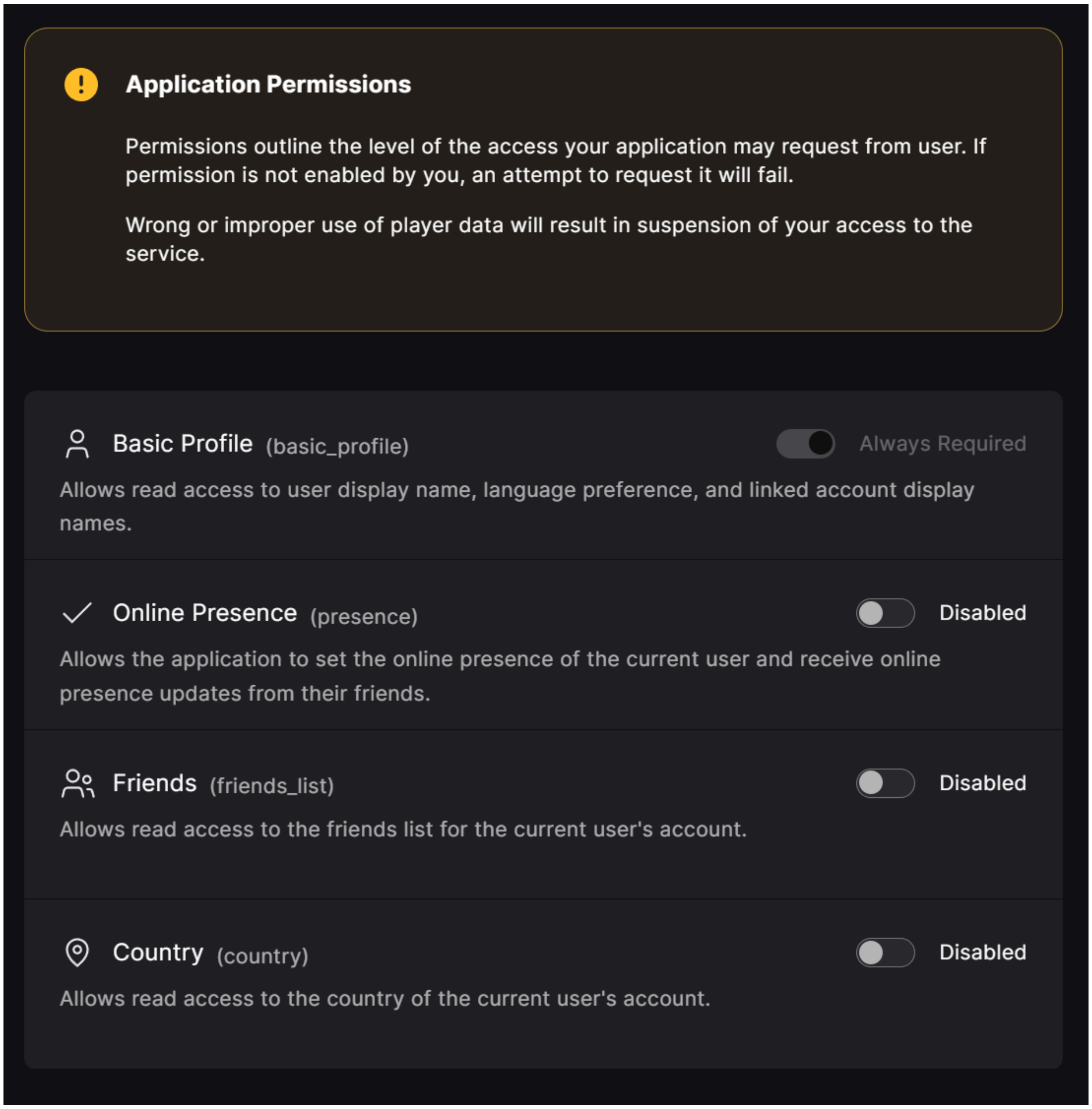
1. Open up a new browser tab and sign in to your domain host account. Go to your domain’s DNS records and select the option to add a new record. Then select TXT type record. In the Value/Answer/Destination field, paste provided Domain Secret. Then save the record on the domain host account.
2. Click “Check TXT Record” button on this page.

- If you’re not sure what your Domain Host is, you can use a website such as [ICANN Lookup](). How you navigate to your domain’s DNS records page can vary based on who

your Domain Host is. In my case, having used SquareSpace as my Domain Host, I found my DNS records through their system.
    - The first of the “Next Steps” outlined above can be a bit finicky due to how specific it is. Ultimately I had to enter my web address on Epic without the preceding “www.”, and I had to set my host on SquareSpace to “@” to get the ***“Check TXT Record”*** button to successfully approve the connection.
- After validating the website, return to the Application setup page from earlier. To populate the first two fields here (***“Application Website”*** and ***“Privacy Policy URL”***) you now have two options:
    - Option 1: Repeat the Website Verification process for a different website so both fields can have different websites associated with them.
    - Option 2: Use the same website for the first two fields.
- Whichever option you choose, all that remains now is the third field: ***“Upload Application Logo”***. This can be any image, as long as it has a size of 64 Kb or less, is of the .jpg or .png format, and has a size of 128 x 128 pixels.
    - You may need to use a website such as ResizePixel.com to resize an image to fit these constraints.
- After filling out all three fields, click ***“Save Draft”*** at the bottom of the screen.
    - There is **no need** to ***“Submit for Review”***, as that is only necessary before listing the game on the storefront. Using the Draft gives us all the functionality we need at this time.
- Return to the Epic Account Services page via the button in the top left when ready.

## Step 3.2 - Permissions

- From the Epic Account Services page, click on the second button, which is labeled ***“Permissions”***.
- The page that opens from here should have four sliders for ***“Basic Profile”***, ***“Online Presence”***, ***“Friends”***, and ***“Country”*** respectively. By default, the first of these will be enabled as it is Always Required, and the other three will be Disabled.
    - Note: Enabling additional sliders unnecessarily here will lead to headaches later (See Step 4).
- Click ***“Save Changes”*** at the bottom of the screen.
- The page should look like the image below:


**Application Permissions**

Permissions outline the level of the access your application may request from user. If permission is not enabled by you, an attempt to request it will fail.

Wrong or improper use of player data will result in suspension of your access to the service.

Basic Profile (basic_profile) — Always Required
Allows read access to user display name, language preference, and linked account display names.

Online Presence (presence) — Disabled
Allows the application to set the online presence of the current user and receive online presence updates from their friends.

Friends (friends_list) — Disabled
Allows read access to the friends list for the current user's account.

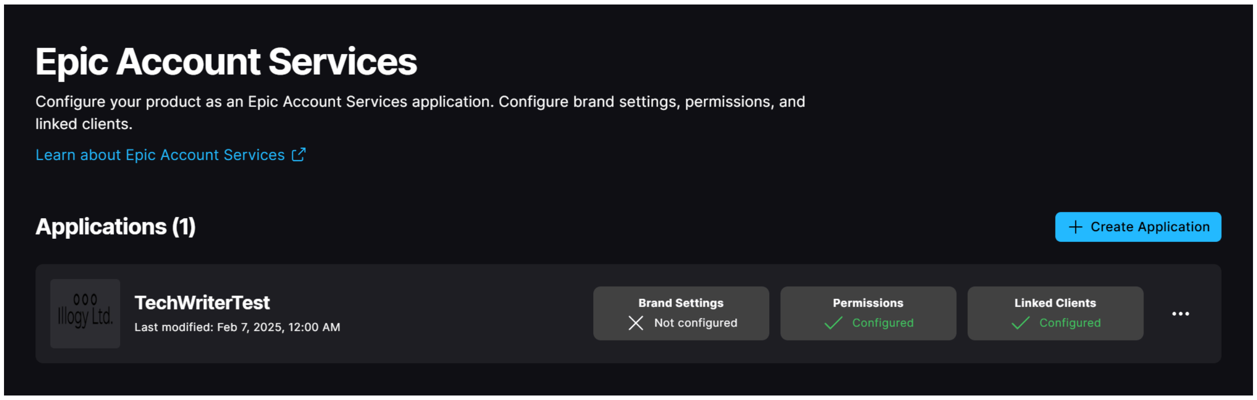
Country (country) — Disabled
Allows read access to the country of the current user's account.

- Return to the Epic Account Services page via the button in the top left when ready.

**<u>Step 3.3 - Linked Clients</u>**

- From the Epic Account Services page, click on the third button, which is labeled ***"Linked Clients"***.
- Select the client you created earlier from the dropdown menu, then click ***"Save Changes"***.
- Return to the Epic Account Services page via the button in the top left when ready.

- After filling out these three fields, the result should match the image below. Notice that the second and third buttons, ***"Permissions"*** and ***"Linked Clients"*** are both labelled as ***"Configured"***, whereas the first button, ***"Brand Settings"***, is not. This is just fine!



Epic Account Services

Configure your product as an Epic Account Services application. Configure brand settings, permissions, and linked clients.

Learn about Epic Account Services

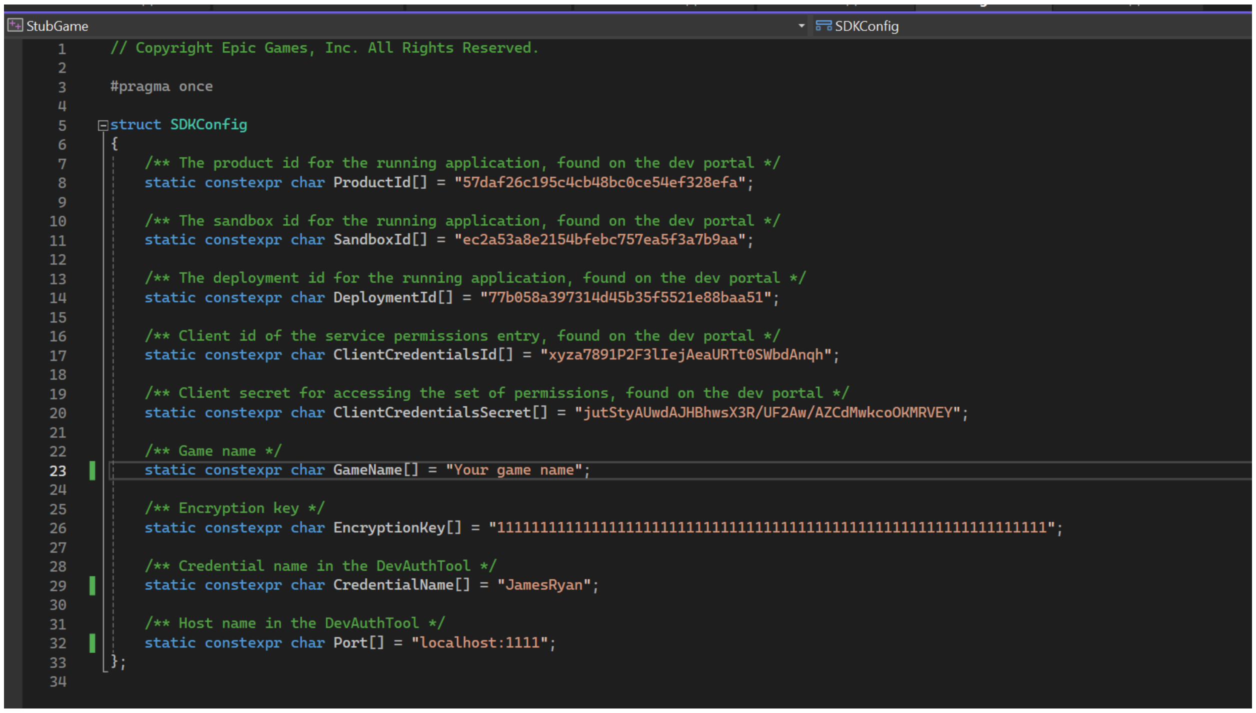
Applications (1)

TechWriterTest

Last modified: Feb 7, 2025, 12:00 AM

## Step 4: Authenticate Your Token And Interact With The EOS Platform Interface

- Now that we have our Application defined, let's try running the program again. You may notice (especially if you ignored the wisdom of Step 3.2) that a new error (18206) occurs: ***"EOS_Auth_CorrectiveActionRequired"***
    - After accepting the request to open the game, the server keeps polling for token authentication/authorization. It is stuck in a loop.
- The solution to this is in the ***AuthHandler.cpp*** file: Check that the Permission settings for the Application match the settings asked for by this file
    - ***LoginOptions.ScopeFlags = EOS_EAuthScopeFlags::EOS_AS_BasicProfile;***
    - Again, since the wisdom of Step 3.2 bears repeating, we need only the Basic Profile and must disable all other authentications/requests for our Application on the Dev site.
- Assuming no other errors pop up, CONGRATULATIONS! You have successfully Implemented the EOS Platform Interface! Below is an image of what my command line looked like after executing successfully, for reference:

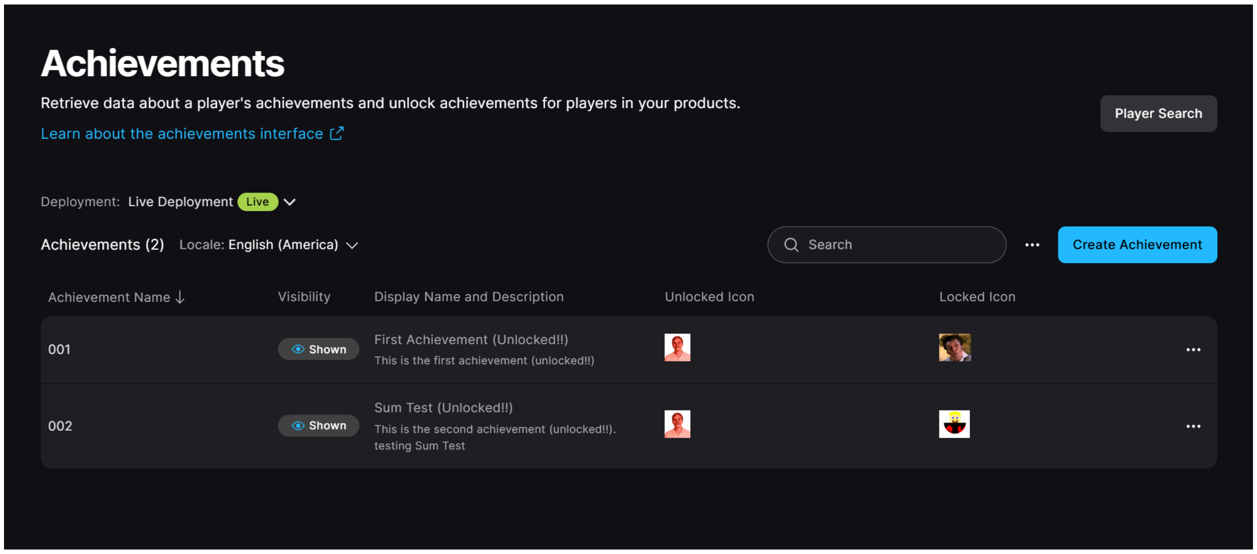


```cpp
// Copyright Epic Games, Inc. All Rights Reserved.

#pragma once

struct SDKConfig
{
    /** The product id for the running application, found on the dev portal */
    static constexpr char ProductId[] = "57daf26c195c4cb48bc0ce54ef328efa";

    /** The sandbox id for the running application, found on the dev portal */
    static constexpr char SandboxId[] = "ec2a53a8e2154bfebc757ea5f3a7b9aa";

    /** The deployment id for the running application, found on the dev portal */
    static constexpr char DeploymentId[] = "77b058a397314d45b35f5521e88baa51";

    /** Client id of the service permissions entry, found on the dev portal */
    static constexpr char ClientCredentialsId[] = "xyza7891P2F3lIejAeaURTt0SWbdAnqh";

    /** Client secret for accessing the set of permissions, found on the dev portal */
    static constexpr char ClientCredentialsSecret[] = "jutStyAUwdAJHBhwsX3R/UF2Aw/AZCdMwkcoOKMRVEY";

    /** Game name */
    static constexpr char GameName[] = "Your game name";

    /** Encryption key */
    static constexpr char EncryptionKey[] = "1111111111111111111111111111111111111111111111111111111111111111";

    /** Credential name in the DevAuthTool */
    static constexpr char CredentialName[] = "JamesRyan";

    /** Host name in the DevAuthTool */
    static constexpr char Port[] = "localhost:1111";
};
```

## II. Implement The EOS Achievements Interface

### <u>Step 1: Create Achievements On The Dev Site</u>

- Start by creating an achievement in the Dev portal, so we have something to eventually unlock
    - In your Organization, navigate via the bar at the left of the screen to select ***"Game Services"***, then ***"Progression"***, then ***"Achievements"***. Click ***"Create Achievement"*** in the top right.
    - A popup window will appear to the right with an empty field for ***"Stat Name"***. You can create a new Stat here to link to your achievement if you wish (i.e. a Sum value that can later be used as a numerical condition for unlocking the achievement, as in "Collect 50 Orbs" or the like).
    - After you have added a Stat (or have decided to abstain from doing so), click "Next" in the bottom right of this and fill out the required fields (***"Achievement ID"***, ***"Unlocked Icon"***, ***"Locked Icon"***, ***"Locked Display Name"***, ***"Locked Description"***, ***"Unlocked Display Name"***, ***"Unlocked Description"***).
    - Below is an example image of the two images I created. The first of these, named "001", has no Stat associated with it, but the second, named "002" is associated with a Stat I created called ***SumTest***, which is set to a value of 5.



Achievements

Retrieve data about a player's achievements and unlock achievements for players in your products.

Learn about the achievements interface

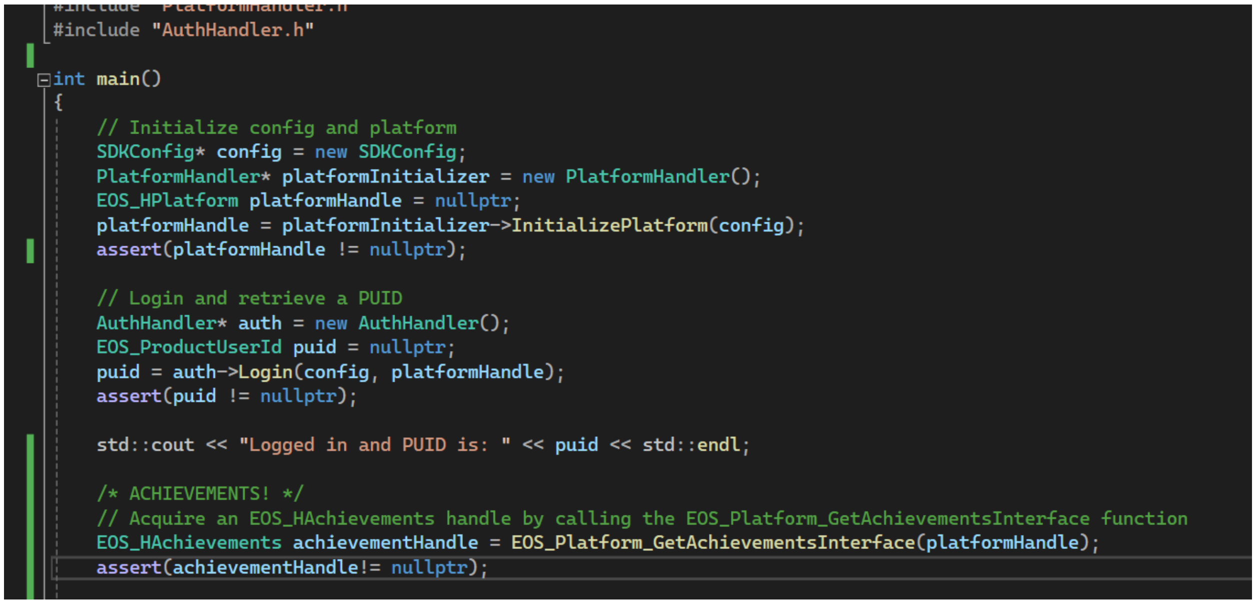
Deployment: Live Deployment Live

Achievements (2) Locale: English (America)

| Achievement Name | Visibility | Display Name and Description | Unlocked Icon | Locked Icon |
|---|---|---|---|---|
| 001 | Shown | First Achievement (Unlocked!!) This is the first achievement (unlocked!!) | | |
| 002 | Shown | Sum Test (Unlocked!!) This is the second achievement (unlocked!!). testing Sum Test | | |

## Step 2: Acquire An AchievementHandle

- Edit the main function in ***StubGame.cpp*** as shown in the image below
    - Acquire an ***EOS_HAchievements*** handle by calling the ***EOS_Platform_GetAchievementsInterface*** function. This will allow us to utilize the Achievement-related functionalities available through the EOS SDK.

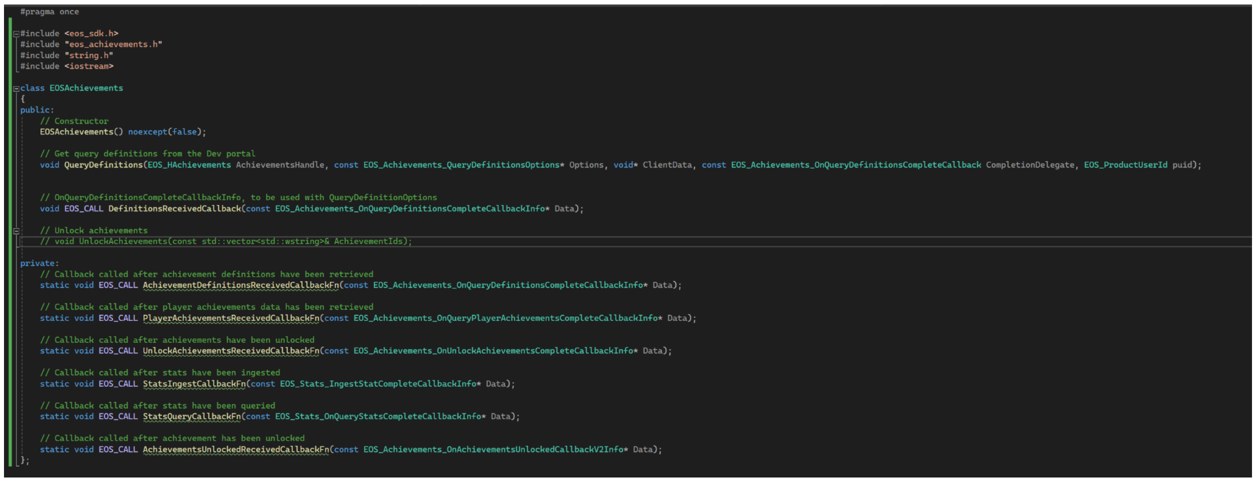


```cpp
#include "AuthHandler.h"

int main()
{
    // Initialize config and platform
    SDKConfig* config = new SDKConfig;
    PlatformHandler* platformInitializer = new PlatformHandler();
    EOS_HPlatform platformHandle = nullptr;
    platformHandle = platformInitializer->InitializePlatform(config);
    assert(platformHandle != nullptr);

    // Login and retrieve a PUID
    AuthHandler* auth = new AuthHandler();
    EOS_ProductUserId puid = nullptr;
    puid = auth->Login(config, platformHandle);
    assert(puid != nullptr);

    std::cout << "Logged in and PUID is: " << puid << std::endl;

    /* ACHIEVEMENTS! */
    // Acquire an EOS_HAchievements handle by calling the EOS_Platform_GetAchievementsInterface function
    EOS_HAchievements achievementHandle = EOS_Platform_GetAchievementsInterface(platformHandle);
    assert(achievementHandle!= nullptr);
```

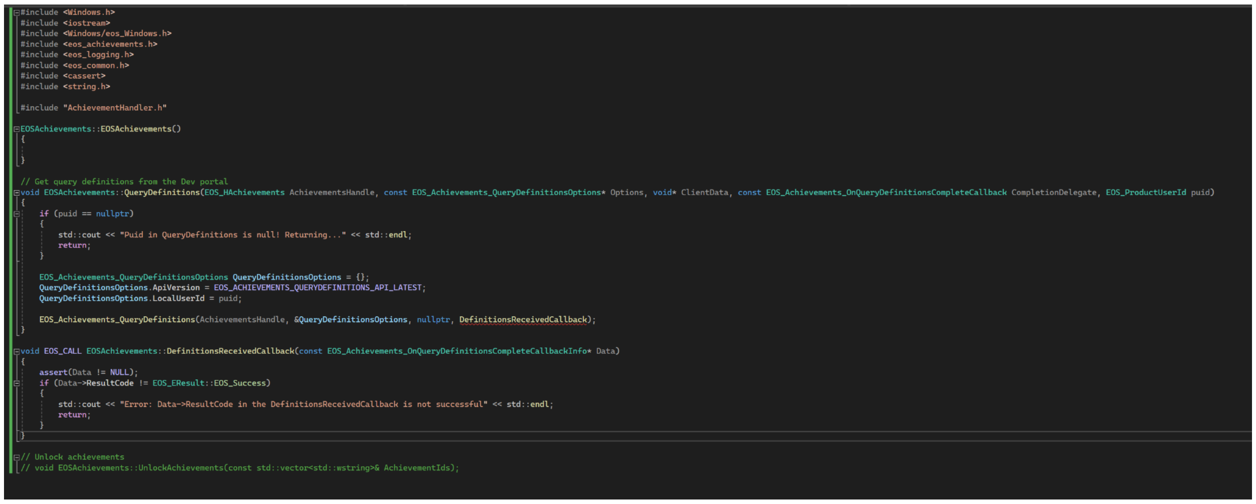
- Run the program. If the assert statement is passed successfully, you're good to go!

## Step 3: Create A New Header File For Implementation

- Since we have to implement the Achievements Interface manually, create a new header file to act as an AchievementHandler (I named mine exactly that, ***AchievementHandler.h***). We will eventually continue this implementation in a corresponding C++ file (which, again, I aptly named ***AchievementHandler.cpp***).
    - The ***Achievements.h*** and ***Achievements.cpp*** Sample Project files installed with EOS are great references for this, though they are a bit more complex than we will ultimately need.
- Below are images of my header and implementation files, respectively:



```cpp
#pragma once

#include <eos_sdk.h>
#include "eos_achievements.h"
#include "string.h"
#include <iostream>

class EOSAchievements
{
public:
    // Constructor
    EOSAchievements() noexcept(false);

    // Get query definitions from the Dev portal
    void QueryDefinitions(EOS_HAchievements AchievementsHandle, const EOS_Achievements_QueryDefinitionsOptions* Options, void* ClientData, const EOS_Achievements_OnQueryDefinitionsCompleteCallback CompletionDelegate, EOS_ProductUserId puid);


    // OnQueryDefinitionsCompleteCallbackInfo, to be used with QueryDefinitionOptions
    void EOS_CALL DefinitionsReceivedCallback(const EOS_Achievements_OnQueryDefinitionsCompleteCallbackInfo* Data);

    // Unlock achievements
    // void UnlockAchievements(const std::vector<std::wstring>& AchievementIds);

private:
    // Callback called after achievement definitions have been retrieved
    static void EOS_CALL AchievementDefinitionsReceivedCallbackFn(const EOS_Achievements_OnQueryDefinitionsCompleteCallbackInfo* Data);

    // Callback called after player achievements data has been retrieved
    static void EOS_CALL PlayerAchievementsReceivedCallbackFn(const EOS_Achievements_OnQueryPlayerAchievementsCompleteCallbackInfo* Data);

    // Callback called after achievements have been unlocked
    static void EOS_CALL UnlockAchievementsReceivedCallbackFn(const EOS_Achievements_OnUnlockAchievementsCompleteCallbackInfo* Data);

    // Callback called after stats have been ingested
    static void EOS_CALL StatsIngestCallbackFn(const EOS_Stats_IngestStatCompleteCallbackInfo* Data);

    // Callback called after stats have been queried
    static void EOS_CALL StatsQueryCallbackFn(const EOS_Stats_OnQueryStatsCompleteCallbackInfo* Data);

    // Callback called after achievement has been unlocked
    static void EOS_CALL AchievementsUnlockedReceivedCallbackFn(const EOS_Achievements_OnAchievementsUnlockedCallbackV2Info* Data);
};
```



```cpp
#include <Windows.h>
#include <iostream>
#include <Windows/eos_Windows.h>
#include <eos_achievements.h>
#include <eos_logging.h>
#include <eos_common.h>
#include <cassert>
#include <string.h>

#include "AchievementHandler.h"

EOSAchievements::EOSAchievements()
{

}

// Get query definitions from the Dev portal
void EOSAchievements::QueryDefinitions(EOS_HAchievements AchievementsHandle, const EOS_Achievements_QueryDefinitionsOptions* Options, void* ClientData, const EOS_Achievements_OnQueryDefinitionsCompleteCallback CompletionDelegate, EOS_ProductUserId puid)
{
    if (puid == nullptr)
    {
        std::cout << "Puid in QueryDefinitions is null! Returning..." << std::endl;
        return;
    }

    EOS_Achievements_QueryDefinitionsOptions QueryDefinitionsOptions = {};
    QueryDefinitionsOptions.ApiVersion = EOS_ACHIEVEMENTS_QUERYDEFINITIONS_API_LATEST;
    QueryDefinitionsOptions.LocalUserId = puid;

    EOS_Achievements_QueryDefinitions(AchievementsHandle, &QueryDefinitionsOptions, nullptr, DefinitionsReceivedCallback);
}

void EOS_CALL EOSAchievements::DefinitionsReceivedCallback(const EOS_Achievements_OnQueryDefinitionsCompleteCallbackInfo* Data)
{
    assert(Data != NULL);
    if (Data->ResultCode != EOS_EResult::EOS_Success)
    {
        std::cout << "Error: Data->ResultCode in the DefinitionsReceivedCallback is not successful" << std::endl;
        return;
    }
}

// Unlock achievements
// void EOSAchievements::UnlockAchievements(const std::vector<std::wstring>& AchievementIds);
```

- You may notice that the above files are incomplete, and they absolutely are! Frankly, with what free time I had from starting this assignment, this was as far as I was able to get with it in a day. I got tripped up on the implementation of ***EOS_Achievements_QueryDefinitions***, specifically with its fourth parameter, ***const EOS_Achievements_OnQueryDefinitionsCompleteCallback CompletionDelegate***.
- I understand that the basic workflow for implementing the Achievement Interface, that being as follows:

    - Call the function ***EOS_Achievements_QueryDefinitions*** to retrieve an array of all Epic Online Services achievement definitions that you have created on the Developer Portal for your application.
        - This function requires a similar function call as a parameter, ***EOS_Achievements_QueryDefinitionsOptions***.
    - Once the query is complete, the Epic Online Services achievement definitions are cached locally rather than being returned through the function. They can then be accessed using ***EOS_Achievements_CopyAchievementDefinitionV2ByIndex*** or ***EOS_Achievements_CopyAchievementDefinitionV2ByAchievementId*** to obtain a copy of an ***EOS_Achievements_DefinitionV2***.
        - ***EOS_Achievements_DefinitionV2*** can be used to access elements of the achievements created in the Dev portal such as ***"AchievementId"***, ***"Unlocked Description"***, etc.
    - After making whichever calls to access and modify achievements (including directly unlocking them or changing the Stats associated with them, as I would've done in the final 3 bullet points outlined in the Task writeup), ultimately the memory for storing these copied definitions must be released using ***EOS_Achievements_DefinitionV2_Release***.
- Given that the Achievement Interface implementation is the prerequisite to the remaining Task bullet points (Send a request to manually unlock an achievement and log a message when this is successfully sent; receive a notification and log a message when an achievement is unlocked; and implement the EOS Stats Interface and unlock an achievement via ingesting a stat.), I decided to stop here in the hopes that the work I have done thus far will be worth something and will speak for itself.

- Thank you for this opportunity; I hope this has showcased my thought processes and potential in this role despite not entirely completing the task.