

# Quickstart: PlayFab Client library for C# in Unity

01/25/2020 • 4 minutes to read •  +1

## In this article

[Requirements](#)

[Download and install PlayFab SDK](#)

[Set your title settings](#)

[Making your first API call](#)

[Finish and execute](#)

[Next steps](#)

Get started with the PlayFab Client library for C# in Unity. Follow steps to install the package and try out example code for a basic task.

This quickstart helps you make your first PlayFab API call in the Unity3d engine. Before continuing, make sure you have completed [Getting started for developers](#), which ensures you have a PlayFab account and are familiar with the PlayFab Game Manager.

## Requirements

- A [PlayFab developer account](#).
- An installed copy of the Unity Editor. To install Unity for personal use via Unity Hub, or Unity+ for professional use, see [Download Unity](#).

### Note

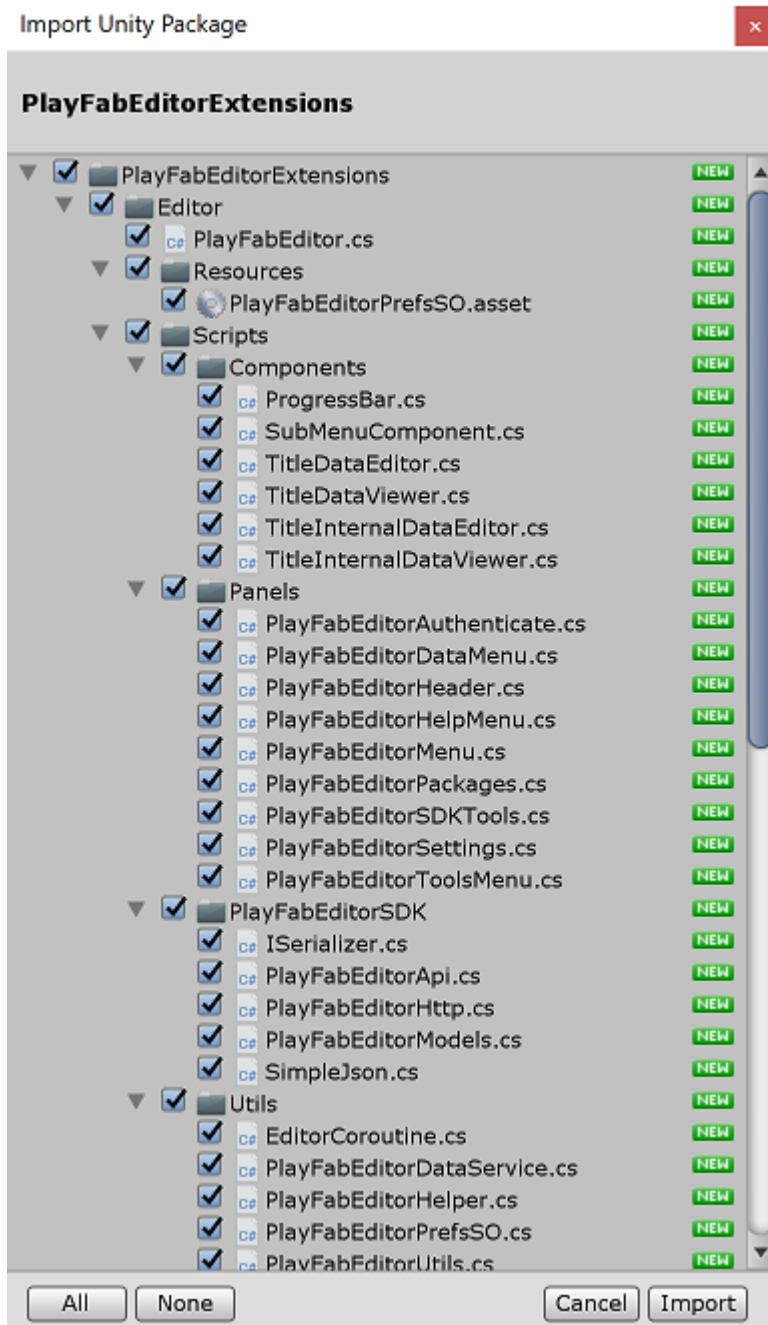
The PlayFab Unity3D SDK supports Unity Editor version 5.3 (released December 2015) and higher.

- A Unity Project - this can be any of the following:
  - A brand new project. For more information, see [Starting Unity for the first time](#).
  - A guided tutorial project. For more information, see [Getting Started with Unity](#).
  - An existing project.
- The PlayFab Unity3D SDK.

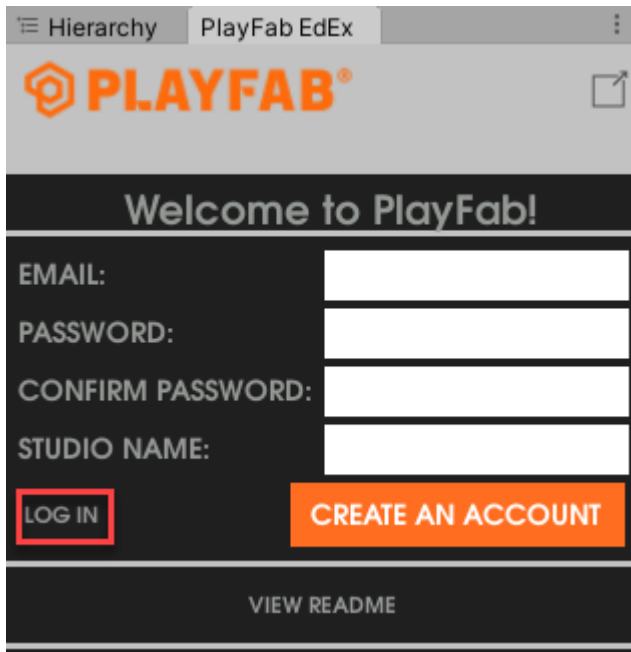
# Download and install PlayFab SDK

Use the PlayFab Editor Extensions package to install the SDK. The PlayFab Editor Extensions are a stand-alone Unity plug-in that streamlines installing the SDK and configuring the PlayFab settings for your Title. For information about installing the SDK without using the PlayFab Editor Extensions, see [Installing the PlayFab SDK for Unity](#).

1. Download the [PlayFab Unity Editor Extensions Asset Package](#).
2. Open your Unity Project.
3. Navigate to where you downloaded the file and double-click on the `PlayFabEditorExtensions.UnityPackage` file to open the **Import Unity Package** dialog in the Unity Editor.



4. To import the PlayFab Unity Editor Extensions into your project, select **Import**.
5. When the import has completed, the PlayFab Unity Editor Extensions panel should open automatically. If you've already created a PlayFab developer account, select the **Log In** link to log in with your PlayFab username and password.

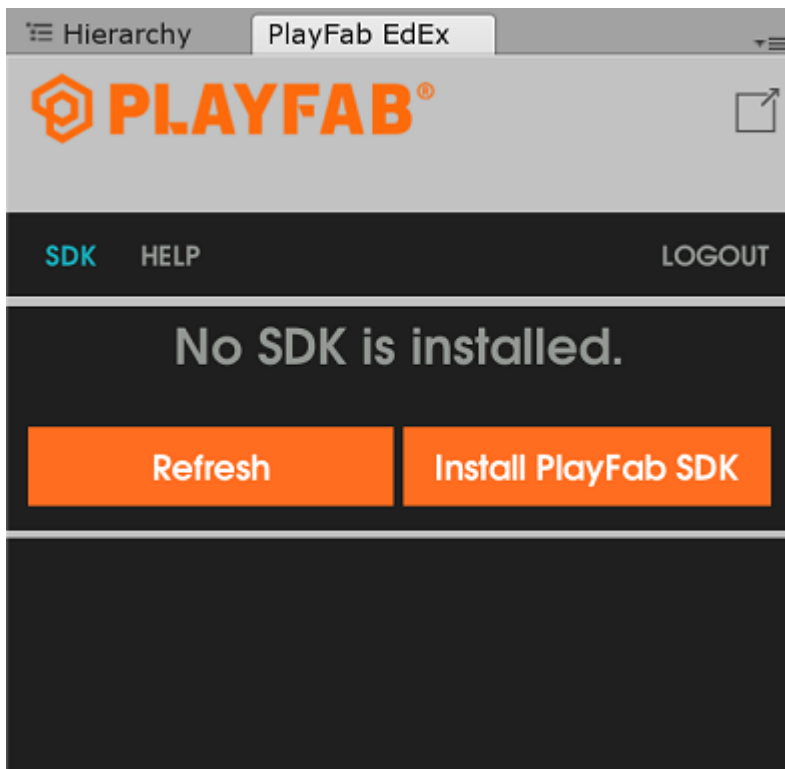


The screenshot shows the PlayFab EdEx interface. At the top, there is a 'Hierarchy' tab and a 'PlayFab EdEx' tab. Below the tabs is the PlayFab logo. The main content area is titled 'Welcome to PlayFab!'. It contains four input fields: 'EMAIL:', 'PASSWORD:', 'CONFIRM PASSWORD:', and 'STUDIO NAME:'. Below these fields are two buttons: 'LOG IN' (highlighted with a red box) and 'CREATE AN ACCOUNT'. At the bottom, there is a 'VIEW README' link.

### ⓘ Note

If the panel did not open, or if you close the panel and want to reopen it, you can do so by selecting **Window > PlayFab > Editor Extensions**

6. After logging in, the extension displays the SDK installation dialog.

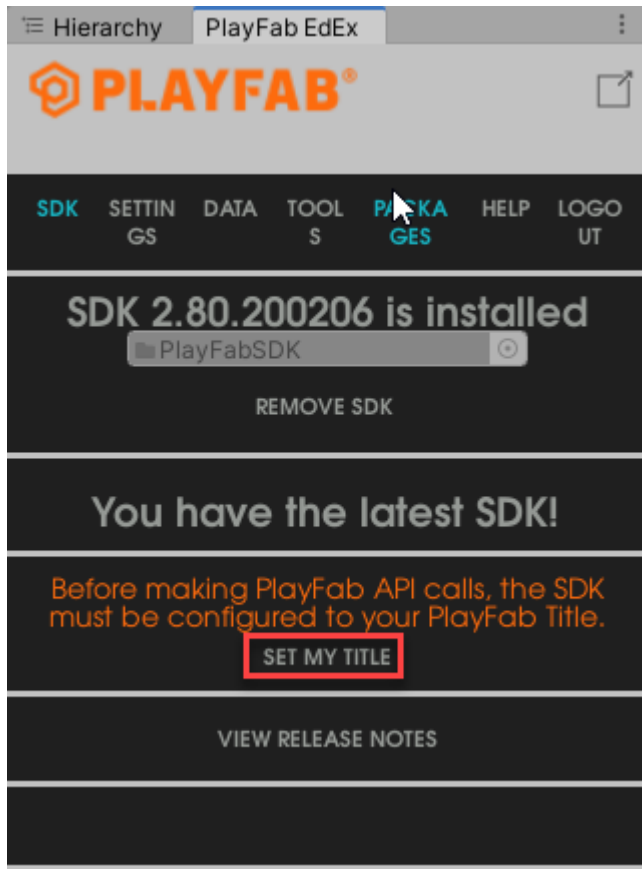


7. Select **Install PlayFab SDK** to automatically import the SDK into your project or upgrade the version that is currently installed.

## Set your title settings

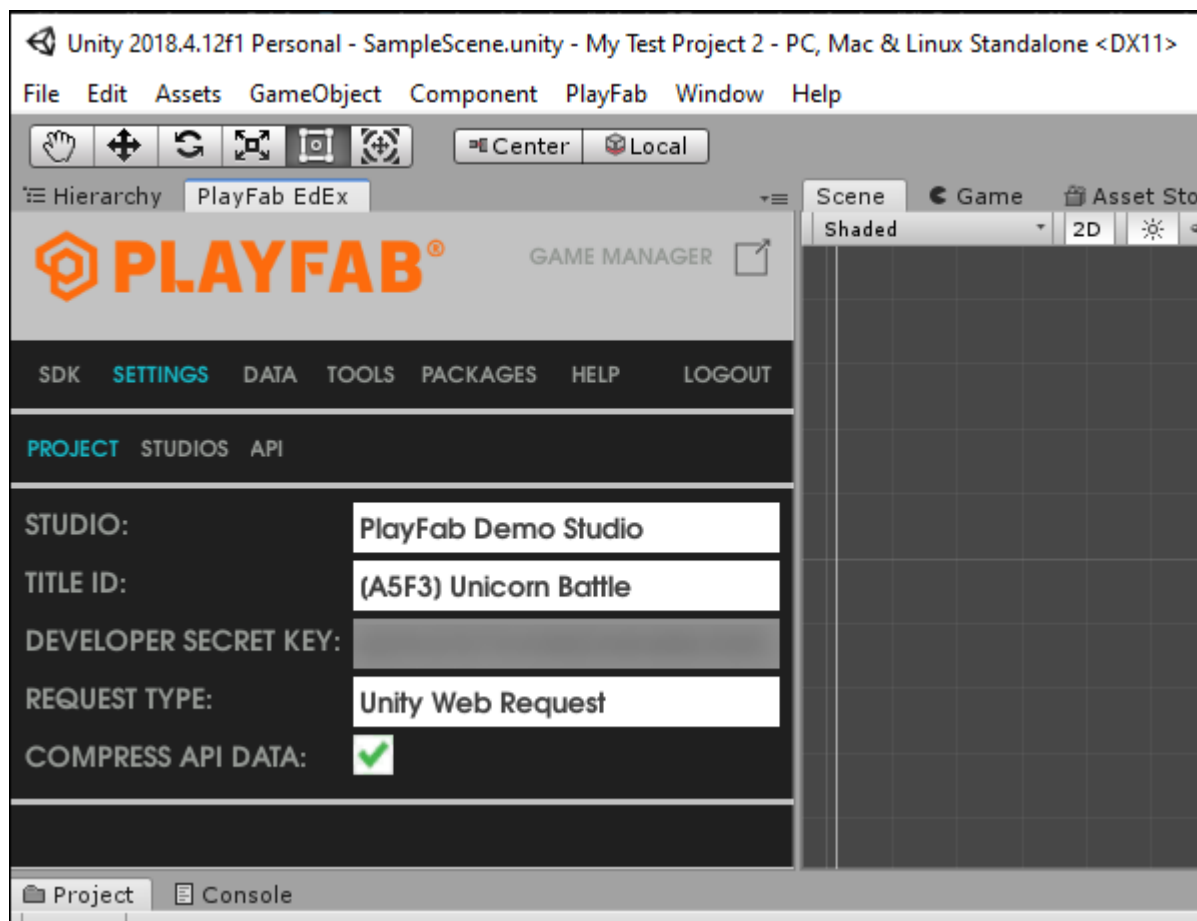
Before you can make an API call, you must specify the Title to receive the call in the PlayFab **Title Settings**. To set the Title:

1. Select **SET MY TITLE** in the **Editor Extensions**.



2. Select the **Studio** entry to open the studio drop-down menu. Select the studio that contains the Title to which you would like to connect.
3. Select the **Title ID** entry to open a drop-down menu of Titles associated with the selected studio.

The **Developer Secret Key** is automatically set to the default secret key for the Title. For more information about secret keys, see [Secret key management](#).



## Making your first API call

This part of the guide provides the minimum steps to make your first PlayFab API call. This example does not provide any GUI or on-screen feedback. Confirmation is displayed in the Console log.

1. If your Unity Project doesn't already have a *Scripts* folder (**HDRP** and **LWRP/URP** templates have one by default), right-click on the **Assets** folder in the Project panel and select **Create > Folder**.
2. In the **Assets** window, name the folder *Scripts*.
3. Right-click the Scripts folder and select **Create > C# Script**.
4. Name the script PlayFabLogin.
5. Double-click the file to open it in a code-editor. Depending on your settings/installed-programs, this is likely Visual Studio or MonoDevelop.
6. In your code editor, replace the contents of PlayFabLogin.cs with the code shown below and save the file.

C#

 Copy

```
using PlayFab;
using PlayFab.ClientModels;
using UnityEngine;

public class PlayFabLogin : MonoBehaviour
{
    public void Start()
    {
        if (string.IsNullOrEmpty(PlayFabSettings.staticSettings.TitleId)){
            /*
             Please change the titleId below to your own titleId from
             PlayFab Game Manager.
             If you have already set the value in the Editor Extensions,
             this can be skipped.
            */
            PlayFabSettings.staticSettings.TitleId = "42";
        }
        var request = new LoginWithCustomIDRequest { CustomId =
"GettingStartedGuide", CreateAccount = true};
        PlayFabClientAPI.LoginWithCustomID(request, OnLoginSuccess,
OnLoginFailure);
    }

    private void OnLoginSuccess(LoginResult result)
    {
        Debug.Log("Congratulations, you made your first successful API
call!");
    }

    private void OnLoginFailure(PlayFabError error)
    {
        Debug.LogWarning("Something went wrong with your first API call.
:(");
        Debug.LogError("Here's some debug information:");
        Debug.LogError(error.GenerateErrorReport());
    }
}
```

### Important

The code shown above is not for use with mobile Titles. This is only an example, and shows how to log in with a `CustomID`. To implement login for a mobile Title, use either **LoginWithAndroidDeviceID**, **LoginWithIOSDeviceID**, or some form of social login such as **LoginWithFacebook**.

7. In the **Hierarchy** panel, right-click your scene, then select **Create Empty** (or **Game Object** > **Create Empty** in older versions of Unity).
8. Select the new Game Object and in the **Inspector** panel, select **Add Component**.
9. From the component drop-down menu, select **Scripts** > **PlayFabLogin**.

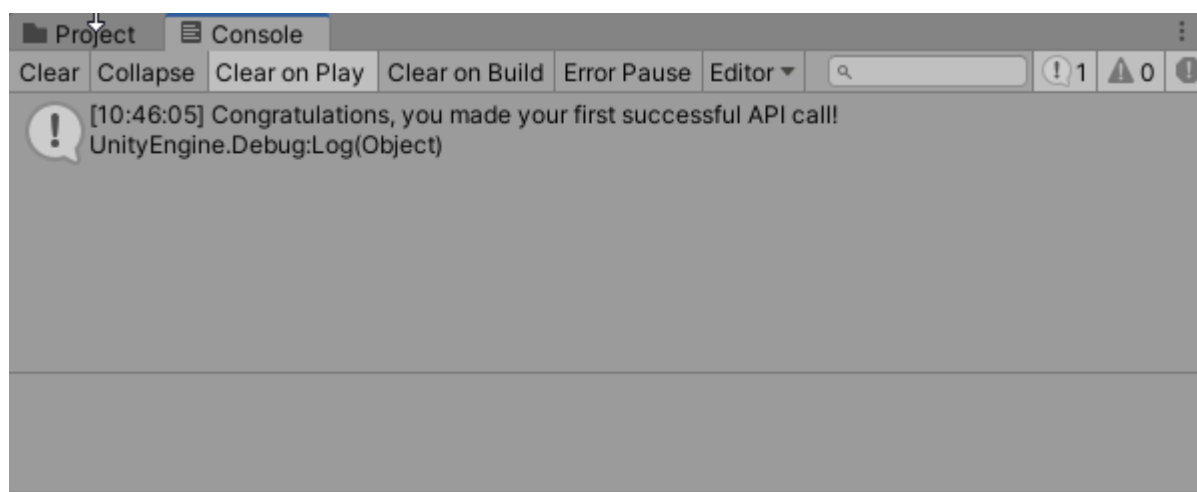
For more information on creating and using scripts in the Unity Editor, see [Creating and Using Scripts](#) in the Unity documentation.

## Finish and execute

You are now ready to test out this sample.

- Be sure to save all files and return to the Unity Editor
- Press the **Play** button at the top of the editor

You should see the following in your Unity Console Panel.



### 💡 Tip

Alternatively, you can log into PlayFab and navigate to the title in Game Manager, and select the **PlayStream Monitor** tab. Each time you **Alt+ TAB** focus away from the actively running Unity Title, it passes an event which you can see and confirm in the PlayStream Monitor.

For a list of all available client API calls, see [PlayFab API References](#).

## Next steps



This quickstart shows a simplified procedure for authenticating a user. For additional information on user authentication, see [Login basics and best practices](#).

Learn how to bind an account to multiple devices and login mechanisms: [Account linking quickstart](#).

---

**Is this page helpful?**

 Yes  No

---