Table of Contents

Quick Start

Please note that the code provided in this page is *purely* for learning purposes and is far from perfect. Remember to null-check all responses!

Breaking Changes Notice

This package currently only supports Google Sign In for Android. Apple Sign In for iOS is expected "soon", and Google Sign In for iOS is being considered. Updates which add support for the aforementioned providers may introduce breaking changes. If you've just updated the package, it is recommended to check the *changelogs*⊡ for information on breaking changes.

Google Sign-In Setup (Android)

Setup your app for Google Sign-In by following these steps (taken from [google-signin-unity](⊡)):

> ### *Configuring the application on the API Console*
>
> To authenticate you need to create credentials on the API console for your application. The steps to do this are available on [Google Sign-In for Android](⊡) or as part of Firebase configuration. In order to access ID tokens or server auth codes, you also need to configure a web client ID.
>
> ### *Get a Google Sign-In configuration file*
>
> This file contains the client-side information needed to use Google Sign-in. The details on how to do this are documented on the [Developer website](⊡).
>
> Once you have the configuration file, open it in a text editor. In the middle of the file you should see the **oauth_client** section:
>
> ```json
>     "oauth_client": [
>       {
>         "client_id": "411000067631-
>   hmh4e210xxxxxxxxxx373t3icpju8ooi.apps.googleusercontent.com",
>         "client_type": 3
>       },
>       {
>         "client_id": "411000067631-
>   udra361txxxxxxxxxx561o9u9hc0java.apps.googleusercontent.com",
>         "client_type": 1,
>         "android_info": {
>           "package_name": "com.your.package.name.",
>           "certificate_hash": "7ada045cccccccccc677a38c91474628d6c55d03"
>         }
> ```

```
        }
    ]
```

There are 3 values you need for configuring your Unity project:

1. The **Web client ID**. This is needed for generating a server auth code for your backend server, or for generating an ID token. This is the `client_id` value for the oauth client with client_type == 3.

2. The **package_name**. The client entry with client_type == 1 is the Android client. The package_name must be entered in the Unity player settings.

3. The keystore used to sign your application. This is configured in the publishing settings of the Android Player properties in the Unity editor. This must be the same keystore used to generate the SHA1 fingerprint when creating the application on the console.

   NOTE: The configutation file does not reference the keystore, you need to keep track of this yourself.

Scene Setup

Add an instance of `GoogleSignInManager` to your first scene (as it is a persistent singleton) and set `ServerClientId` to your Web client ID.

Sign In

To sign in at runtime, just call `SignIn()` and register to its callbacks (`OnSignedIn` and `OnSignInFailed`) or call `SignInAsync()` to get the results asynchronously:

```csharp
using Uralstech.UMoth.GoogleSignIn;

public async void SignIn()
{
    (GoogleIdTokenCredential credential, SignInFailReason failReason) = await
GoogleSignInManager.Instance.SignInAsync();
    if (credential == null)
    {
        Debug.LogError($"Failed to get credentials due to error: {failReason}");
        return;
    }

    Debug.Log($"Got credentials: {credential}");
}
```

Both `SignIn()` and `SignInAsync()` contain optional parameters to configure the operation. Please check the reference documentation for more info.

Sign Out

Like signing in, just call [SignOut()](#) and register to [OnSignedOut](#) and [OnSignOutFailed](#) or call [SignOutAsync()](#) to get the results asynchronously:

```csharp
using Uralstech.UMoth.GoogleSignIn;

public async void SignOut()
{
    bool result = await GoogleSignInManager.Instance.SignOutAsync();
    if (!result)
        Debug.LogError("Could not sign out!");
    else
        Debug.Log("Signed out successfully.");
}
```

Firebase Integration

You can use the [IdToken](#) from signing in to create a Firebase Auth credential, like so:

```csharp
(GoogleIdTokenCredential? result, SignInFailReason failReason) = await GoogleSignInManager.Instance.SignInAsync();
if (result is null)
{
    Debug.LogError($"Could not sign in due to error: {failReason}");
    return;
}

Credential fbCredential = GoogleAuthProvider.GetCredential(result.IdToken, null);

try
{
    AuthResult authResult = await FirebaseAuth.DefaultInstance.SignInAndRetrieveDataWithCredentialAsync(fbCredential).ConfigureAwait(true);
    Debug.Log("User logged in successfully.");
}
catch (FirebaseException exception)
{
    Debug.LogException(exception);
}
```