Snap Kit

Unity Plugin

This Unity plugin allows developers to easily bring Snap Kit functionality into their Unity projects. It includes features from two of our most popular and impactful kits: Login Kit and Creative Kit

App Registration

You need to register your app at the Snap Kit developer portal.

After registering your app you will receive two **OAuth Client IDs**.

You can use the **Development Client ID** anytime even before an app is reviewed and approved. But when using it, only accounts listed under the *Demo Users* in the app registration/profile page on Snap Kit developer portal will be able to use your application.

With the **Production Client ID**, your app can post the content from any Snapchat account. But your app must be approved for this Client ID to work.

Setting up your app

This package includes a demo scene under `Assets/Snap/Demo/Demo.unity` When the setup steps below are correctly configured, running the demo scene will allow you to test some Snap Kit functionality, like Login and Sharing.

In order for the Demo Scene to work, open your App Settings in the SnapKit Developer Portal and check the following:

1. A version for your app is created in the **Versions** menu
2. Your Snapchat username is listed under **Demo Users**
3. Your package identifier is listed under **Platform Identifiers**, for the correct platform
4. LoginKit is enabled in your version, with the following options: **Display Name**, **Bitmoji Avatar**, **Snapchat Verify**.
5. Still in the Login Kit pane, there's a valid **Redirect URI for OAuth**, for example: unitytest://snap-kit/oauth2
6. **Creative Kit** is enabled in your version.

After checking all of the above, select the "Setup" section in the left navigation bar and make sure that the **Version** you just configured is the active Version on **Staging** for your app

---

Unity Set up

1. Download the latest release of `Snapkit.unitypackage`
2. With your Unity project open, double click the package file and import all files
3. After import is done, open the `Assets/Snap/Editor` folder in your project explorer
4. Check your Assets/Snap/Editor folder. If there's an asset called SnapKitSettings.asset file, that's where you'll input the information from your app. If the file is not present, create a new one in that folder by selecting **Assets->Create->Snap Kit Settings**
5. Click on the SnapKitSettings asset and make sure that the Inspector pane is visible
6. In the inspector pane, populate the information from your SnapKit Portal registered app.
    1. **Client ID**: your "OAuth2 Client ID"
    2. **Redirect URI**: one of your "Redirect URIs for OAuth"

3. **URL Scheme**: the first part of the chosen Redirect URI. For example, if the Uri is `myapp://snap-kit/oauth/`, the URL Scheme is `myapp` (Note that the scheme needs to be defined as a lower-case string without special characters or numbers)
4. **Host**: the second part of the chosen Redirect URI. For example, if the Uri is `myapp://snap-kit/oauth/`, the host is `snap-kit`
5. **Path Prefix**: the third part of the chosen Redirect URI. For example, if the Uri is `myapp://snap-kit/oauth/`, the path prefix is `/oauth`

7. Go to Player Settings and make sure that the package name chosen for your app is one of the pre-defined "Platform Identifiers" defined on the Snap Kit portal

Additional steps on Android

1. Go to Unity's Player Settings
2. Set **Internet Access** to "Require"
3. Set the **Minimum API Level** to "21"

Additional steps on iOS

1. Make sure to have CocoaPods installed
2. Once the XCode project is built, select the Unity-iPhone Project -> (Targets) Unity iPhone -> Build Phases -> Expand "Embed Frameworks"
3. Drag the following Frameworks from the "Pods" project into that section: **SCSDKCoreKit**, **SCSDKLoginKit**, **SCSDKCreativeKit**
4. (Optional) if you get an error of "SCSDKLoginKit.h file not found", type `pod deintegrate && pod install` into a terminal at the iOS build folder and then on Xcode go to Product -> Clean Build Folder.

Example: Logging in

The example below shows a minimal `MonoBehaviour` that can perform Login with Snapchat

```
using Snap;
using UnityEngine;

public class LoginManager : MonoBehaviour
{
    void OnEnable()
    {
        LoginKit.OnLoginLinkDidSucceedEvent += OnLoginSuccess;
        LoginKit.OnLoginLinkDidFailEvent += OnLoginFail;
    }

    void OnDisable()
    {
        LoginKit.OnLoginLinkDidSucceedEvent -= OnLoginSuccess;
        LoginKit.OnLoginLinkDidFailEvent -= OnLoginFail;
    }

    public void OnButtonTapped_Login()
    {
        LoginKit.Login();
    }

    void OnLoginSuccess()
    {
        Debug.Log("Login Succeeded");
    }
```

```
    void OnLoginFail()
    {
        Debug.Log("Login Failed");
    }
}
```

SDK Methods

`LoginKit.Login()`

Invokes the Snapchat app for the user to authorize your application. When completed will invoke either `OnLoginLinkDidSucceedEvent` or `OnLoginLinkDidFailEvent` depeneding on the result of the call

`LoginKit.IsLoggedIn()`

Synchronously returns a boolean indicating whether or not the user is already logged in with Snapchat

`LoginKit.UnlinkAllSessions()`

Clears the access token in the client, effectively unlinking the user with the app. Will fire a `LoginDidUnlink` event

`LoginKit.GetAccessToken()`

Synchonously returns the current access token as a String. Will be `null` if the user is not logged in

`LoginKit.Verify(string number, string region)`

(**Only on iOS**) Login Kit's newest feature Verify With Snapchat allows users to quickly register in third-party applications using the phone number attached to their Snapchat account. Will fire a `OnVerifySucceededEvent` or `OnVerifyFailedEvent` depending on the result of the call.

`LoginKit.HasAccessToScope(string scope)`

Synchronously Checks if the user has given permission to the given scope within the app.

`LoginKit.FetchUserDataWithQuery(string graphql, Dictionary<string, object> variables)`

Receives a GraphQL query (with optional variables) and returns the values obtained from the Snap Kit API. Will fire a `OnFetchUserDataSucceededEvent` with a json payload if successful, or a `OnFetchUserDataFailedEvent` with an error message if not.

`CreativeKit.Share(ShareContent share)`

Shares to Snapchat based on the parameters specified in `ShareContent`. Will fire a `OnSendSucceededEvent` if successful or a `OnSendFailedEvent` if failed.

Events

`LoginKit.OnLoginLinkDidSucceedEvent`

Fired when LoginKit.Login() succeeds

`LoginKit.OnLoginLinkDidFailEvent`

Fired when LoginKit.Login() fails

`LoginKit.OnLoginDidUnlinkEvent`

Fired when LoginKit.UnlinkAllSessions() succeeds

`LoginKit.OnFetchUserDataSucceededEvent`

Fired when LoginKit.FetchUserDataWithQuery() succeeds

`LoginKit.OnFetchUserDataFailedEvent`

Fired when LoginKit.FetchUserDataWithQuery() fails

`LoginKit.OnVerifySucceededEvent`

Fired when LoginKit.Verify() succeeds

`LoginKit.OnVerifyFailedEvent`

Fired when LoginKit.Verify() fails

`CreativeKit.OnSendSucceededEvent`

Fired when CreativeKit.Share() succeeds

`CreativeKit.OnSendFailedEvent`

Fired when CreativeKit.Share() fails

Frequently Asked Questions

I can't test with different users

In order to test with another user, you need to first switch to another user in the Snapchat app and then try the SnapKit integration. Trying to login from your game will only allow to log in with the currently signed-in user on Snapchat and won't prompt the user for a new log in.

I get a "Something Went Wrong" error

This is the default error message for anything that can go wrong with SnapKit. If you get one of these messages try checking that all the requirements for getting the Demo Scene working above are set up correctly.

I can share with Creative Kit even if the user is not logged in

This behavior is by design. Your player doesn't need to be connected with LoginKit in order to be able to share content with CreativeKit.