Documentation for Cordova Snapkit plugin

This plugin allows you to integrate Snapchat features into your HTML5 app. It uses the native implementations of Loginkit, CreativeKit, Adkit (see https://kit.snapchat.com/docs) and provides a way for apps exported with Cordova to interact with them.

An example Playcanvas project that uses all 3 kits can be found at: https://playcanvas.com/project/771129/overview/snapkitplugin-demo

Github for the cordova project:

Table of contents

- 1. Installation
- 1.1 Snapkit Developer Portal
- 2 LoginKit
  - 2.1 Android congfiuration
  - iOS 2.2 configuration
  - 2.3 Using LoginKit
    - 2.3.1 Login
    - 2.3.2 Logout
    - 2.3.4 Checking logged in status
    - 2.3.5 Getting userdata
  - 2.4 Problems?
- 3 Creative Kit
  - 3.1 Android congfiuration
  - 3.2 iOS configuration
  - 3.3 Using CreativeKit
- 4 Adkit
  - 4.1 Android configuration
  - 4.2 iOS configuration
  - 4.3 Using Adkit
    - 4.3.1 Loading interstitials/rewarded ads
    - 4.3.2 Playing ads
  - 4.3.3 Events

1. Installation

In your cordova project run:

```
cordova plugin add https://github.com/christianthomsenff/cordova-snapkit-plugin
```

1.1 Snapkit Developer Portal

To use Snapkit you need to create an app in the Snapkit developer portal.

Follow the the documentation on the developer portal https://kit.snapchat.com/docs/developer-portal to setup your app.

2 LoginKit

Make sure you have toggled LoginKit on in the Snapkit developer portal. You'll also need to whitelist a Snapchat user you can use for testing, add a bundle ID and a redirect URL.

2.1 Android congfiuration

LoginKit requires some changes to the AndroidManifest. Paste the following in your app's config.xml between the `<platform name="android">` start and end tag. Input the values for client id, redirect url and scheme, host and path from the Snapkit developer portal.

```
<config-file parent="/manifest/application" target="AndroidManifest.xml">
    <meta-data android:name="com.snapchat.kit.sdk.clientId" android:value="your
app’s OAuth2 client id" />
    <meta-data android:name="com.snapchat.kit.sdk.redirectUrl" android:value="the
url that will handle login completion" />
    <meta-data android:name="com.snapchat.kit.sdk.scopes"
android:resource="@array/snap_connect_scopes" />

    <activity android:name="com.snapchat.kit.sdk.SnapKitActivity"
android:launchMode="singleTask">
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <category android:name="android.intent.category.DEFAULT" />
            <category android:name="android.intent.category.BROWSABLE" />

            <!--
                Enter the parts of your redirect url below
                e.g., if your redirect url is myapp://snap-kit/oauth2
                    android:scheme="myapp"
                    android:host="snap-kit"
                    android:path="/oauth2"   (make sure to include /)
            !-->

            <data
                android:scheme="the scheme of your redirect url"
                android:host="the host of your redirect url"
                android:path="the path of your redirect url"
                />
        </intent-filter>
    </activity>
</config-file>
```

Now create a file called arrays.xml and paste in:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="snap_connect_scopes">
        <item>https://auth.snapchat.com/oauth2/api/user.bitmoji.avatar</item>
        <item>https://auth.snapchat.com/oauth2/api/user.display_name</item>
        <item>https://auth.snapchat.com/oauth2/api/user.external_id</item>
    </string-array>
</resources>
```

Save it at the path res/values/arrays.xml at the root of your project. This defines which scopes LoginKit have access to. Here both the display name, external id and bitmoji.

And add a line after the `</config-file>` tag that copies this file to the correct location when building.

```
<resource-file src="res/values/arrays.xml"
target="app/src/main/res/values/arrays.xml" />
```

More info on scopes and Android configuration can be found at: https://kit.snapchat.com/docs/login-kit-android

Additionally you may need to enable AndroidX to be able to build

```
<preference name="AndroidXEnabled" value="true" />
```

iOS 2.2 configuration

To use LoginKit we need to modify info.plist. Add the following to your config.xml between the `<platform name="iOS">` tags and replace placeholders with your OAuth2 clientID, redirect URL and app bundle ID which can all be found in Snapkit's developer portal.

```
<config-file parent="CFBundleURLTypes" target="*-Info.plist">
    <array>
        <dict>
            <key>CFBundleTypeRole</key>
            <string>Editor</string>
            <key>CFBundleURLName</key>
            <string>your app bundle ID</string>
            <key>CFBundleURLSchemes</key>
            <array>
                <string>Your redirect url scheme</string> //If your redirect URL
is myapp://snap-kit/oath2, this field should be my app<
            </array>
        </dict>
    </array>
</config-file>
```

```
<config-file parent="SCSDKClientId" target="*-Info.plist">
    <string>Your OAuth2 clientId</string>
</config-file>
<config-file parent="SCSDKRedirectUrl" target="*-Info.plist">
    <string>Your redirect URL</string>
</config-file>
<config-file parent="SCSDKScopes" target="*-Info.plist">
    <array>
        <string>https://auth.snapchat.com/oauth2/api/user.display_name</string>
        <string>https://auth.snapchat.com/oauth2/api/user.external_id</string>
        <string>https://auth.snapchat.com/oauth2/api/user.bitmoji.avatar</string>
    </array>
</config-file>
<config-file parent="LSApplicationQueriesSchemes" target="*-Info.plist">
    <array>
        <string>snapchat</string>
    </array>
</config-file>
<config-file parent="NSAppTransportSecurity" target="*-Info.plist">
    <dict>
        <key>NSAllowsArbitraryLoads</key>
        <true />
    </dict>
</config-file>
```

2.3 Using LoginKit

Loginkit is ready for use after Cordova's deviceready event fires. Note you can subscribe to the event at any time. If you subscribe after the event has fired the callback function will be called immediately. The plugin can be accessed through the window object: `window.LoginKit`

```
LoginKitPlugin.prototype.initialize = function() {
    document.addEventListener('deviceready', this.onDeviceReady.bind(this));
};

LoginKitPlugin.prototype.onDeviceReady = function() {
    if(window.LoginKit) {
        //LoginKit is available
    } else {
        //Could not find loginkit
    }
};
```

2.3.1 Login

To initiate a login simply call the corresponding methods on window.Loginkit.

```
window.LoginKit.login();
```

For updates on login success or failure, LoginKit provides two events you can subscribe to:

```
window.LoginKit.onLoginSucceeded = this.onLoginSucceeded.bind(this);
window.LoginKit.onLoginFailed = this.onLoginFailed.bind(this);


LoginKitPlugin.prototype.onLoginSucceeded = function() {
    //Player is now logged in
};

LoginKitPlugin.prototype.onLoginFailed = function(err) {
    console.log("Login failed", err);
};
```

2.3.2 Logout

Similarly to logout a user simply call:

```
window.LoginKit.logout();
```

And subscribe to the onLogout event:

```
window.LoginKit.onLogout = this.onLogout.bind(this);

LoginKitPlugin.prototype.onLogout = function() {
    //Player is now logged out
};
```

2.3.4 Checking logged in status

To check if a user is currently logged in call isLoggedIn. It's a promise that resolves to either "true" or "false".

```
window.LoginKit.isLoggedIn().then(response => {
    if(response == "true")
    {
        //Player is logged in
    }

    if(response == "false")
    {
        //Player is not logged in
    }
});
```

2.3.5 Getting userdata

When a user is logged in we can access his Snapchat display name, 2D bitmoji and external ID. This is done by passing in a query string to the method fetchUserData. A query string for bitmoji, displayname and externalID would look like:

```
var queryString = "{me{bitmoji{avatar},displayName,externalId}}"
```

Passing it to the fetchUserData returns a response object that contains the requested fields.

```
window.LoginKit.fetchUserData(queryString).then(response => {
    //response.displayname
    //response.externalId
    //response.bitmoji
});
```

Make sure to take into account that not all Snapchat users have bitmojis. If they do, the bitmoji field will return a base64 string. You'll most likely need to convert it to a Playcanvas texture to be able to use it. Example:

```
var tex = new pc.Texture(this.app.graphicsDevice, {});
var img = document.createElement('img');
img.src = response.bitmoji;
img.crossOrigin = 'anonymous';
img.onload = (evt) => {
    tex.setSource(img);
    this.bitmoji.element.texture = tex;
};
```

2.4 Problems?

While implementing Loginkit you might get to a point where returning from Snapchat shows an error "Something went wrong!". This is a catch all error that indicates a mismatch between the IDs, bundle IDs, user whitelisting or redirect URIs in your app and in the Snapkit portal. Make sure to double check that your app is setup correctly in Snapkit and all values match with your app.

3 Creative Kit

Currently this plugin only supports sending overlay stickers. Note that stickers must be PNGs 1 MB or smaller.

3.1 Android congfiuration

Paste the following in your app's config.xml between the `<platform name="android">` and add your app's Oauth2 client ID from Snapkit's developer portal.

```
<config-file parent="/manifest/application" target="AndroidManifest.xml">
    <meta-data android:name="com.snapchat.kit.sdk.clientId" android:value="your
app’s OAuth2 client id" />
<config-file>
```

Add access to FileProvider by adding the following in the `<config-file ...>` tag:

```
<provider android:authorities="${applicationId}.fileprovider"
android:exported="false" android:grantUriPermissions="true"
android:name="androidx.core.content.FileProvider">
    <meta-data android:name="android.support.FILE_PROVIDER_PATHS"
android:resource="@xml/file_paths" />
</provider>
```

Images have to be saved to a temporary location before they can be shared. Create an xml file and save it at the path res/xml/file_paths.xml with the following

```
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <cache-path name="tmp" path="/" />
</paths>
```

And this line in your `<platform name="android">` tag to ensure the file is copied over when building.

```
<resource-file src="res/xml/file_paths.xml"
target="app/src/main/res/xml/file_paths.xml" />
```

3.2 iOS configuration

Add the following in your config.xml between the iOS platform tags and input your OAuth2 clientID from Snapkit's developer portal.

```
<config-file parent="SCSDKClientId" target="*-Info.plist">
    <string>Your OAuth2 clientID</string>
</config-file>

<config-file parent="LSApplicationQueriesSchemes" target="*-Info.plist">
    <array>
        <string>snapchat</string>
    </array>
</config-file>
```

3.3 Using CreativeKit

CreativeKit can be accessed on `window.CreativeKit`, and provides a share method that takes a sticker object. The sticker object allows you to specify rotation, size, placement and the optional caption and attachmentUrl fields. Clicking the sticker will take the user to the attachmentUrl if specified. The dataUrl field expects an image encoded as a base64 string. Example:

```
var sticker = {
    width: 500,
    height: 500,
    centerX: 0.5,
    centerY: 0.5,
    rotation: 300,
    dataUrl:
"data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAUAAAAFCAYAAACNbyblAAAAHElEQVQI12P
4//8/w38GIAXDIBKE0DHxgljNBAAO9TXL0Y4OHwAAAABJRU5ErkJggg==",
    caption: "Cool caption" //Optional,
    attachmentUrl: "https://snap.com" //Optional on Android
};

window.CreativeKit.share(sticker).then(() => {
    console.log("Share OK");
}).catch(err => {
    console.log(err);
});
```

Note as of writing the sticker won't show up on iOS without an attachmentUrl!

4 Adkit

As of writing Ad Kit is not available to all developers, reach out to your partner engineer on how to get whitelisted or apply for access on https://snapkit.com/ad-kit. Make sure to read the first section on https://kit.snapchat.com/docs/ad-kit-android as it provides important information on how to properly test AdKit. Currently this requires using a US-based VPN.

4.1 Android configuration

None.

4.2 iOS configuration

Add the following to your config.xml between the iOS platform tags

```
<config-file parent="NSAppTransportSecurity" target="*-Info.plist">
    <dict>
        <key>NSAllowsArbitraryLoads</key>
        <true />
    </dict>
```

```
</config-file>
<config-file parent="SKAdNetworkItems" target="*-Info.plist">
    <array>
        <dict>
            <key>SKAdNetworkIdentifier</key>
            <string>424m5254lk.skadnetwork</string>
        </dict>
    </array>
</config-file>
```

Note the native iOS version allows configuration and targeting of ads. This is not yet exposed in this plugin, but the plugin can be forked and additional configuration can be added to AdKit.swift located in cordova-snapkit-plugin/src/ios/

4.3 Using Adkit

After the deviceready event you can access AdKit on the window object. The first thing you'll want to do is initialize by calling window.AdKit.init(appId). The appId in this case is the "Snap Kit App ID" that can be found by navigating to your app on the Snapkit developer portal.

```
AdKitPlugin.prototype.initialize = function() {
    document.addEventListener('deviceready', this.onDeviceReady.bind(this));
};

AdKitPlugin.prototype.onDeviceReady = function() {
    if(window.AdKit)
    {
        //Found AdKit, initialize...
        window.AdKit.onSnapAdInitSucceeded =
this.onSnapAdInitSucceeded.bind(this);
        window.AdKit.onSnapAdInitFailed = this.onSnapAdInitFailed.bind(this);

        var snapKitAppId = "1abc2345-1234-1234-1234-1abc2345abc";
        window.AdKit.init(snapKitAppId);
    }
}

AdKitPlugin.prototype.onSnapAdInitSucceeded = function() {
    console.log("AdKit init success!");
};

AdKitPlugin.prototype.onSnapAdInitFailed = function(err) {
    console.log("AdKit init failed!", err);
};
```

4.3.1 Loading interstitials/rewarded ads

To request an interstitial:

```
window.AdKit.loadInterstitial(slotId);
```

This loads an interstitial into memory which can be played anytime. To load a rewarded ad use the corresponding

```
window.AdKit.loadRewarded(slotId);
```

You can load several ads into memory and play them one by one. For example, you can load an interstitial ad with various slotIDs, then load a few rewarded ads; then, you'll be able to play interstitial ads or rewarded ads as you please, as long as the requested slotID has an ad loaded in memory.

You can listen to the success or failure of ad loading:

```
....
window.AdKit.onSnapAdLoadSucceeded = this.onSnapAdLoadSucceeded.bind(this);
window.AdKit.onSnapAdLoadFailed = this.onSnapAdLoadFailed.bind(this);
...

AdKitPlugin.prototype.onSnapAdLoadSucceeded = function(slotId) {
    console.log("Loaded ad ("  + slotId + ")");
};

AdKitPlugin.prototype.onSnapAdLoadFailed = function(slotId) {
    console.log("Failed to load ad (" + slotId + ")");
};
```

4.3.2 Playing ads

If an ad is loaded with the passed in slotId, you play the ad by calling:

```
window.AdKit.playAd(slotId)
```

If you request more than one ad of a certain type on iOS, the one you loaded most recently will play.

4.3.3 Events

In addition to events previously mentioned (init success/fail and loaded ad sucess/fail), AdKit comes with these events:

```
...

window.AdKit.onSnapAdRewardedEarned = this.onSnapAdRewardedEarned.bind(this);
window.AdKit.onSnapAdVisible = this.onSnapAdVisible.bind(this);
window.AdKit.onSnapAdClicked = this.onSnapAdClicked.bind(this);
window.AdKit.onSnapAdDismissed = this.onSnapAdDismissed.bind(this);
window.AdKit.onSnapAdImpressionHappened =
this.onSnapAdImpressionHappened.bind(this);

...

AdKitPlugin.prototype.onSnapAdRewardedEarned = function(slotId) {
    console.log("onSnapAdRewardedEarned", slotId);
};

AdKitPlugin.prototype.onSnapAdVisible = function(slotId) {
    console.log("onSnapAdVisible", slotId);
};

AdKitPlugin.prototype.onSnapAdClicked = function(slotId) {
    console.log("onSnapAdClicked", slotId);
};

AdKitPlugin.prototype.onSnapAdDismissed = function(slotId) {
    console.log("onSnapAdDismissed", slotId);
};

AdKitPlugin.prototype.onSnapAdImpressionHappened = function(slotId) {
    console.log("onSnapAdImpressionHappened", slotId);
};
```