OAuth for Native Apps – Draft Best Current Practice

William Denniss
wdenniss@google.com

#OAuthForNativeApps

https://tools.ietf.org/html/draft-ietf-oauth-native-apps


WebView +
OAUTH

The host app can extract the cookies:

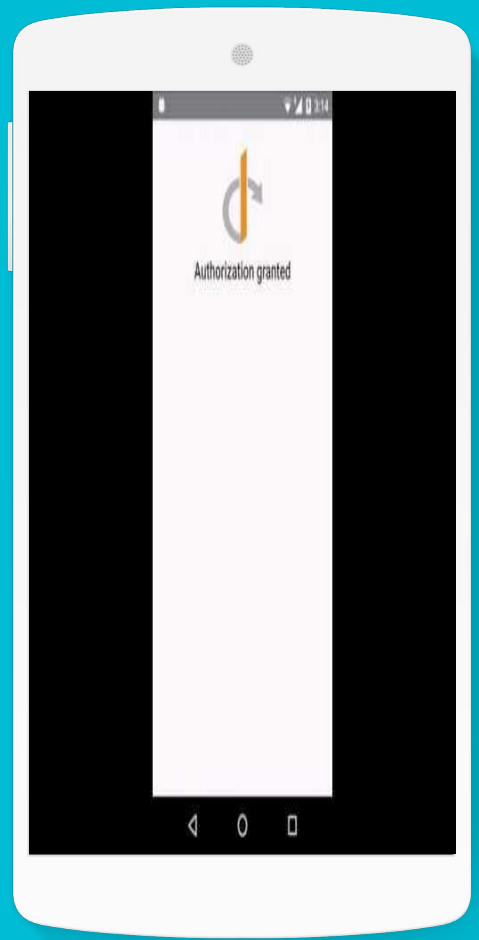
```
String cookies = CookieManager.getInstance().getCookie(url);
```

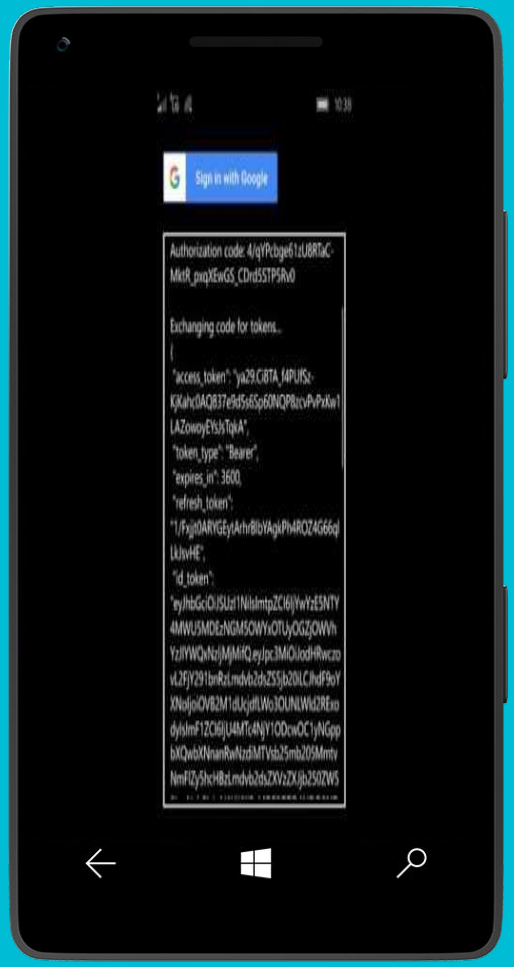
Or inject javascript:

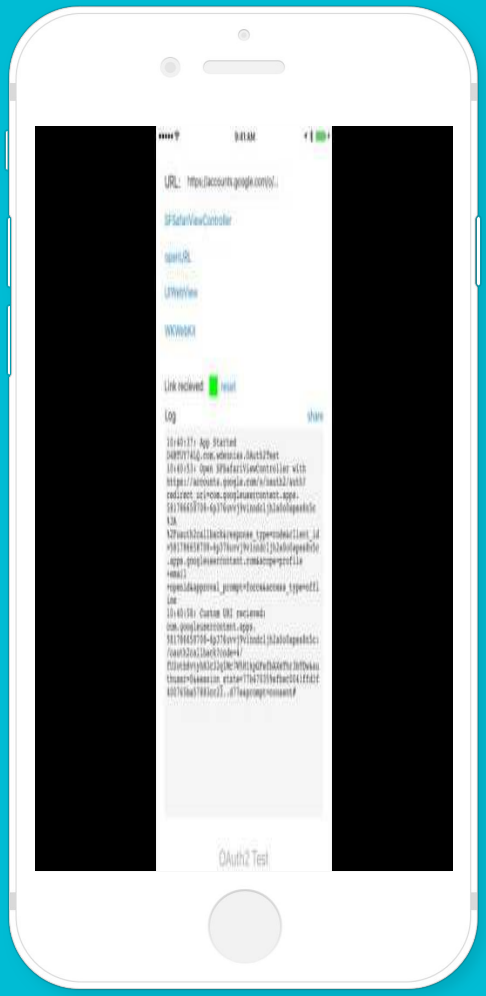
```
webView.evaluateJavascript(
    "(function() { return document.getElementById('password').value; })
();",
    new ValueCallback<String>() {
 @Override public void onReceiveValue(String s) {
   Log.d("WebViewField", s);
 }
});
```

*Single Sign-On means you only sign-on **once**!*

Using the Browser as the OAuth user agent for Apps



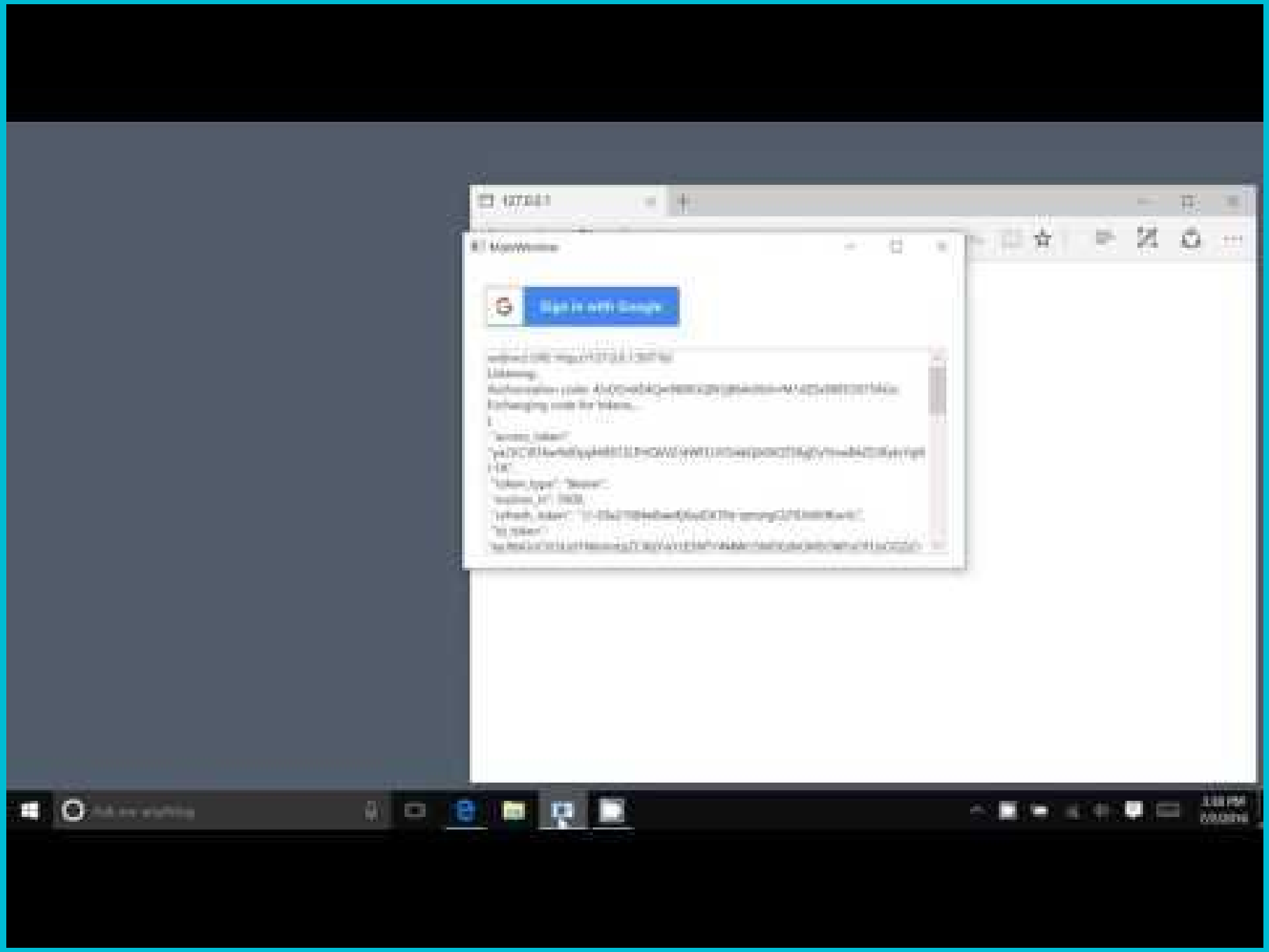




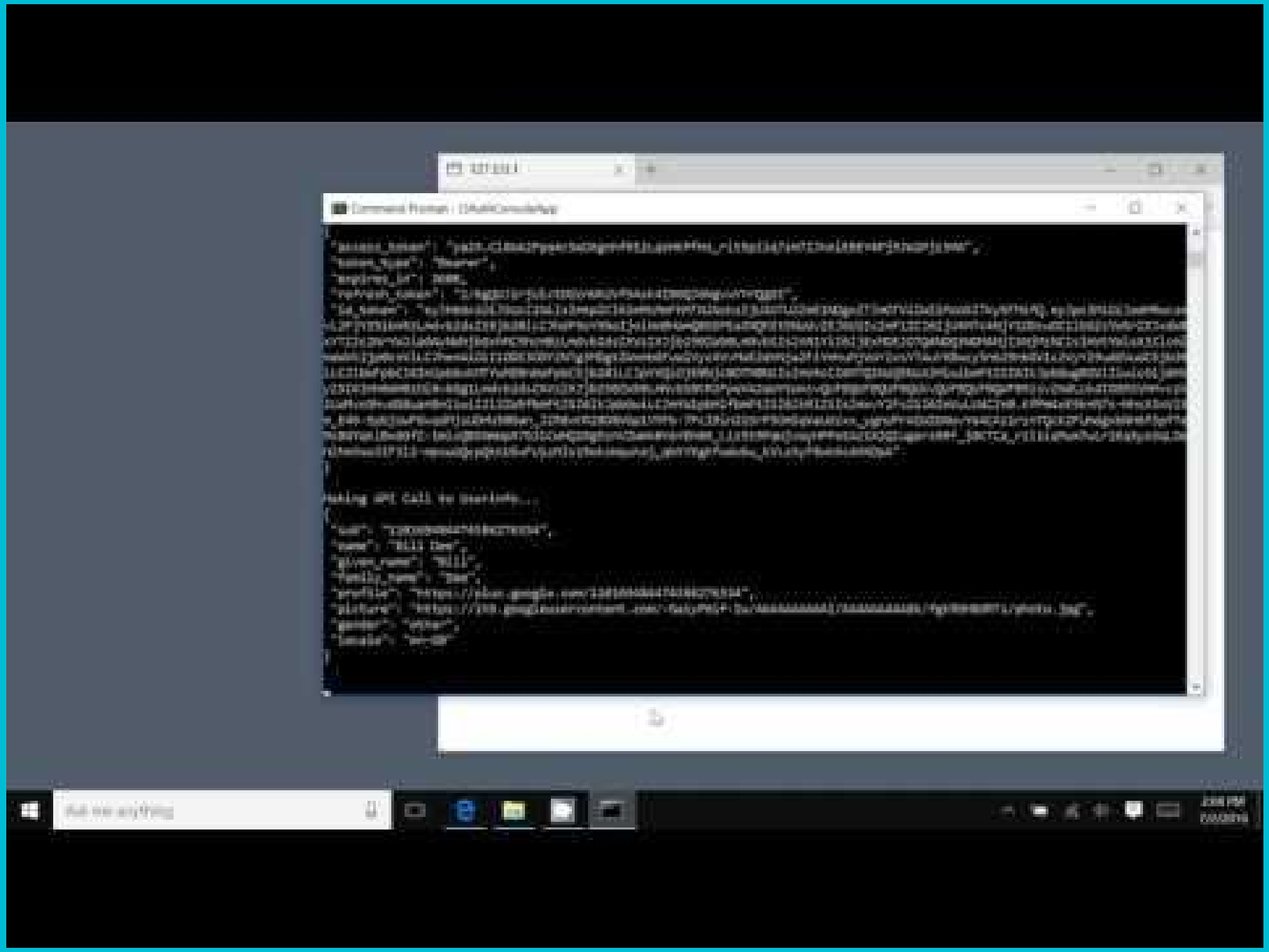
https://youtu.be/3DXhFGfKh-g

https://youtu.be/1l7GAu-do5k

https://youtu.be/Qvu8LPGxcjc



https://youtu.be/F6ZN5x2om54

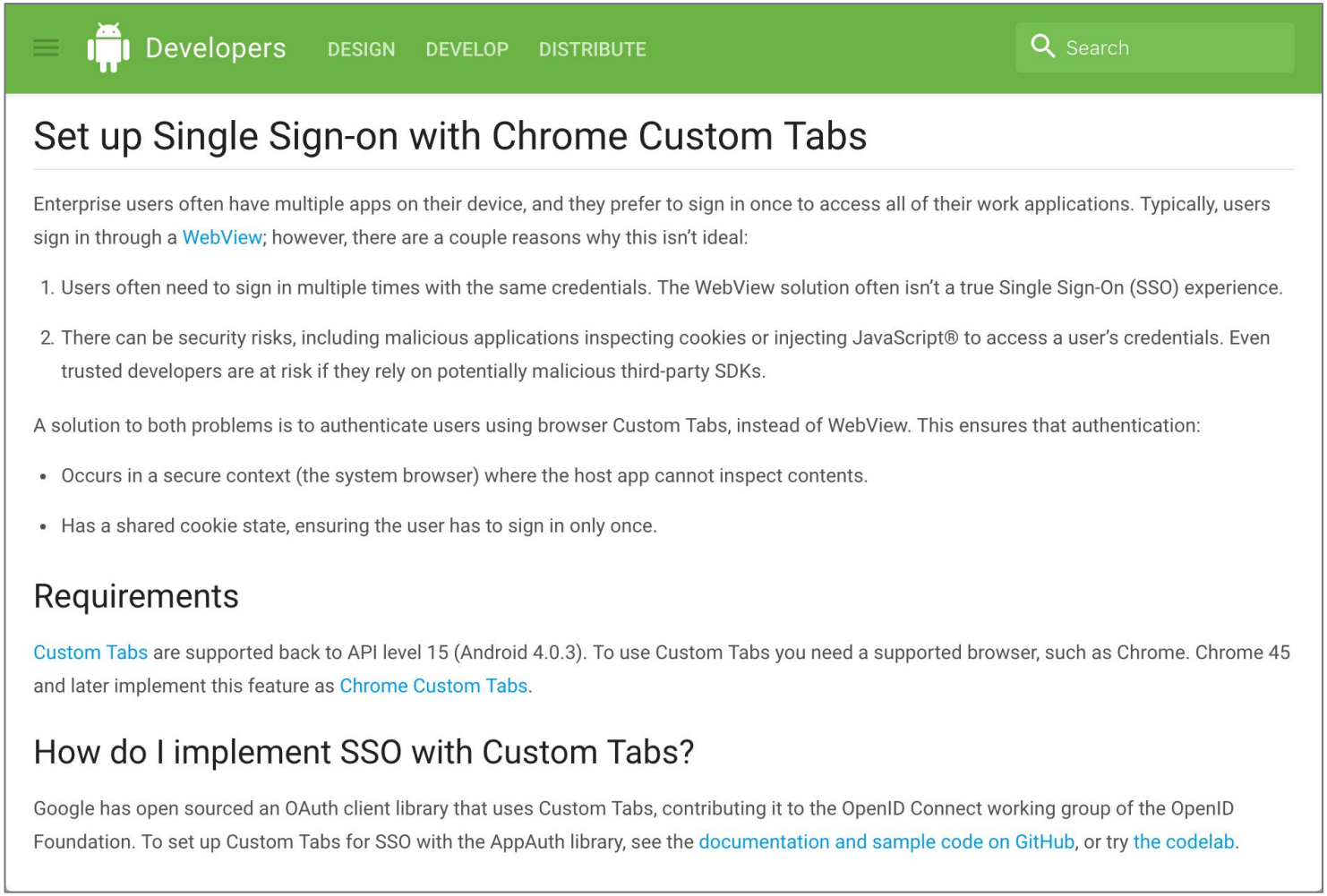


https://youtu.be/dY_opCwVAEA

Android OAuth, Google OAuth

Recommended for SSO on Android

https://developer.android.com/work/guide.html#sso



Developers DESIGN DEVELOP DISTRIBUTE Search

Set up Single Sign-on with Chrome Custom Tabs

Enterprise users often have multiple apps on their device, and they prefer to sign in once to access all of their work applications. Typically, users sign in through a WebView; however, there are a couple reasons why this isn't ideal:

1. Users often need to sign in multiple times with the same credentials. The WebView solution often isn't a true Single Sign-On (SSO) experience.
2. There can be security risks, including malicious applications inspecting cookies or injecting JavaScript® to access a user's credentials. Even trusted developers are at risk if they rely on potentially malicious third-party SDKs.

A solution to both problems is to authenticate users using browser Custom Tabs, instead of WebView. This ensures that authentication:

- Occurs in a secure context (the system browser) where the host app cannot inspect contents.
- Has a shared cookie state, ensuring the user has to sign in only once.

Requirements

Custom Tabs are supported back to API level 15 (Android 4.0.3). To use Custom Tabs you need a supported browser, such as Chrome. Chrome 45 and later implement this feature as Chrome Custom Tabs.

How do I implement SSO with Custom Tabs?

Google has open sourced an OAuth client library that uses Custom Tabs, contributing it to the OpenID Connect working group of the OpenID Foundation. To set up Custom Tabs for SSO with the AppAuth library, see the documentation and sample code on GitHub, or try the codelab.

Recommended for Google OAuth

https://developers.google.com/identity/protocols/OAuth2InstalledApp

Choosing a Redirect Method and Registering your Client

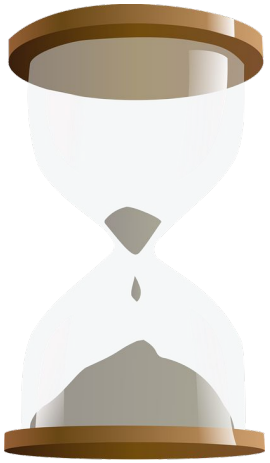
The redirect URI specified in the request determines how the authorization code is returned to your application. There are several options available to installed apps for receiving the authorization code, the availability and user experience of which varies by platform.

Custom URI Scheme

Redirect URI	`com.example.app:`, where *com.example.app* is the reverse DNS notation of a domain under your control. The custom scheme must contain a period to be valid. You may append an optional path component (e.g. `com.example.app:/oauth2redirect`). Note the single "/" after the custom URI scheme, which is different from regular HTTP URLs
Recommended Usage	iOS Apps, Android Apps, Universal Windows Platform (UWP) apps.
Client Type	Register with type **Android** for an Android app, otherwise choose **iOS**. When registering your client, the package name or bundle id is the custom URI scheme used in the redirect (e.g. *com.example.app*). For UWP apps, the scheme cannot exceed 39 characters in length.

Your app must register with the system for the custom URI scheme in order to receive the authorization response. How your app registers its custom URI scheme varies by platform. Our libraries and samples demonstrate some platform-specific implementations of custom URI scheme redirects.

OAuth in a WebView is on Deprecation Watch



Running Code

OAuth for Apps: Code!

Android library:
http://openid.github.io/AppAuth-Android

iOS library (macOS coming soon!):
http://openid.github.io/AppAuth-iOS

Google OAuth Samples for Windows:
https://github.com/googlesamples/oauth-apps-for-windows

AppAuth for Android Codelab!

https://codelabs.developers.google.com/codelabs/appauth-android-codelab/

Next Steps

TODO: Add Implementation Detail Appendix for Windows

Implementation suggestions of the browser pattern for standards-based OAuth on Windows.

Note: The “implementation detail” appendix is non-normative, and only documents some current platform-specific libraries and patterns.

- Open issues resolved.
- Fairly stable for ~1yr.
- Multiple interoperable implementations.

Time for last call

Thank You!

William Denniss
@WilliamDenniss

#OAuthForNativeApps