# 

**Guide to**

**Temenos WebView(WKWebView) plugin**

Table of Contents

[1](#_Toc531873769)

[Introduction 4](#_Toc531873770)

[Overview 4](#_Toc531873771)

[Cordova and WKWebView 4](#_Toc531873772)

[Cordova Upcoming Release 5](#_Toc531873773)

[How to use the plugin 6](#_Toc531873774)

[Installation 6](#_Toc531873775)

[Verifying the usage of WKWebView 8](#_Toc531873776)

[Lifecycle of the application 9](#_Toc531873777)

[WKWebView didFailNavigation event handler 9](#_Toc531873778)

[WKWebView didFinishNavigation event handler 9](#_Toc531873779)

[WKWebView didFailProvisionalNavigation event handler 10](#_Toc531873780)

[WKWebView didFailNavigation event handler 10](#_Toc531873781)

[WKWebView decidePolicyForNavigationAction event handler 10](#_Toc531873782)

[WKWebView didReceiveScriptMessage event handler 10](#_Toc531873783)

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Author** | **Version** | **Date** | **Details** |
| Vamadevan Arun | 1.0 | 26.11.2018 | Creation |
| Vamadevan Arun | 1.1 | 06.12.2018 | Added few more details of the plugin |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Comments:**

|  |
| --- |
|  |
|  |
|  |

# Introduction

The purpose of this document is to provide a description of what has been done for improving the Hybrid application performance on iOS. It’s been achieved by changing the current browser control with comparatively newer one. We have done some performance testing with TCMB 2.0 and the result were positive.

This is intended for developers and people who needs to use this improvement.

# Overview

Current iOS hybrid application uses UIWebView to load web resources. Apple introduces this control back in iOS 2.0. iOS applications were using it till Apple releases WKWebView(with iOS 8.0). This control is just a replacement for the UIWebView control. Apple says there are number of improvements with the replacement controls and some of them are given below.

* Uses more advanced JavaScript engine (Nitro) which speed up the script execution
* Handles java script asynchronously
* Screen rendering is improved compared to UIWebView
* CPU usage is minimal
* Supports Indexed DB

Apart from the above-mentioned advantages, there are some limitations as well especially when we use UIWebView previously. Below are some of them.

* Cookies don't persist
* Can't delete cookies
* Can't execute JavaScript code in the background
* XmlHttpRequests don't work
* Migration of localStorage from UIWebView

# Cordova and WKWebView

UXP hybrid application uses Cordova bridge to fill the gap between native and a web application. Cordova engine internally uses UIWebView. It still uses UIWebView as there are some known issues with WKWebView. Please follow below link to get more details.

[Known issues - WKWebview](https://issues.apache.org/jira/browse/CB-14180?jql=project%20%3D%20CB%20AND%20labels%20%3D%20wkwebview-known-issues)

Considering the performance improvement given by WKWebView, Cordova creates another plugin which internally uses WKWebView. Cordova tried to rectify most of the issues WKWebView has via this plugin.

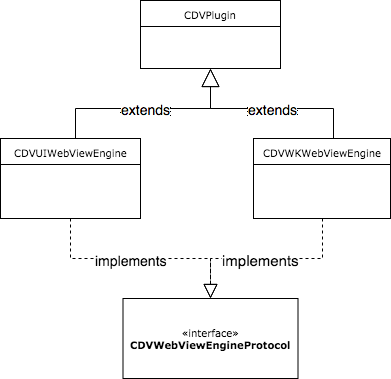
TemenosWebView is created on top of[cordova-plugin-wkwebview-file-xhr](https://github.com/oracle/cordova-plugin-wkwebview-file-xhr)which internally uses Cordova WKWebView plugin. We choose this plugin because of below reasons:

* The default behavior of WKWebView is to raise a cross origin exception when loading files from the main bundle using the file protocol - "file://". This plugin works around this shortcoming by loading files via native code if the web view's current location has "file" protocol and the target URL passed to the open method of the XMLHttpRequest is relative. As a security measure, the plugin verifies that the standardized path of the target URL is within the "www" folder of the application's main bundle.
* Since the application's starting page is loaded from the device's file system, all XHR requests to remote endpoints are considered cross origin. For such requests, WKWebView specifies "null" as the value of the Origin header, which will be rejected by endpoints that are configured to disallow requests from the null origin. This plugin works around that issue by handling all remote requests at the native layer where the origin header will be excluded.

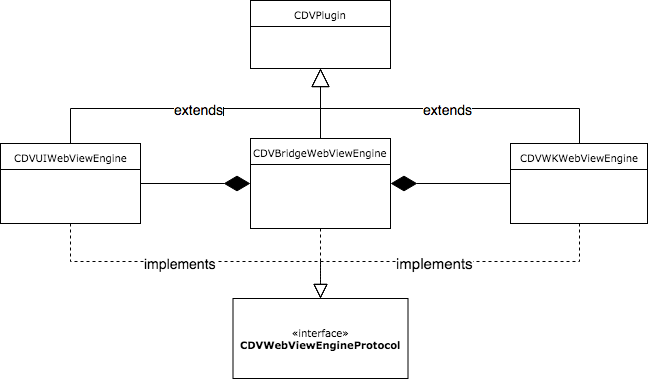
## Cordova Upcoming Release

Apple deprecated UIWebView class from iOS12 onwards. It means that sometime in the future, Apple will remove UIWebView from their SDK, and developers should migrate to using the WKWebView. This is the other reason we are also moving from UIWeBview.

Cordova iOS, starting with version 4, has anticipated this by moving the webview that is used by the platform into a plugin. The default webview that is used is still UIWebView, but we have the option to use WKWebView instead, with the “[cordova-plugin-wkwebview-engine](https://github.com/apache/cordova-plugin-wkwebview-engine)” plugin. Both the UIWebView and WKWebView webviews are plugins themselves, with the former included in the cordova-ios platform.



Starting with iOS version 5 (the next major release for Cordova iOS platform. The current version used by UXP is 4.5.4), Cordova will ship both the webview plugins. Also, Cordova iOS 5 will ship with a bridge webview plugin that can switch usage of the webview plugin used at runtime. Previously, we could only choose which webview we would use at build time.



When UIWebView support has been removed in a future iOS SDK, Cordova will aim to release a future Cordova iOS version, which will remove UIWebView support, and WKWebView will then be the default webview engine.

| **Cordova iOS 4** | **Cordova iOS 5** | **Cordova iOS Future Release** |
| --- | --- | --- |
|  | UIWebView deprecated | UIWebView removed |
|  | WKWebView Engine Plugin Integrated into Platform | WKWebView Engine Plugin only |
|  | Bridge WebView Plugin Added | Bridge WebView Plugin Removed |

# How to use the plugin

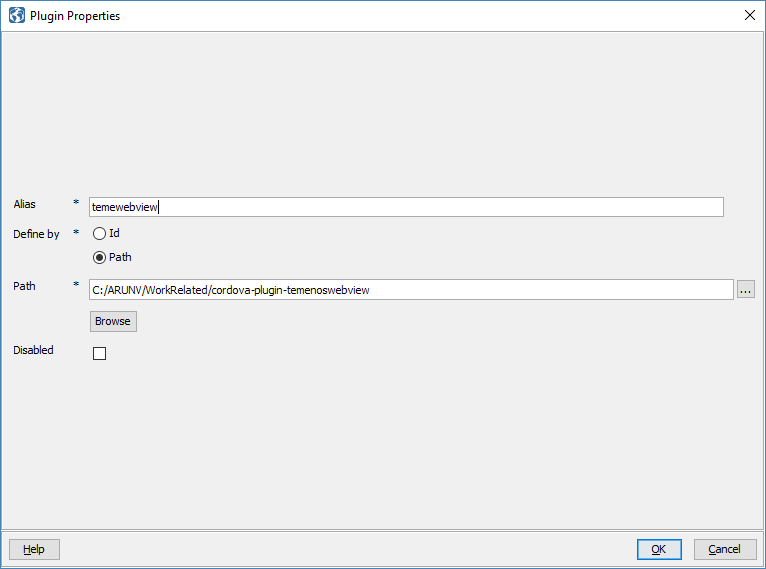
This plugin can be used like any other Cordova plugin we use with UXP. There are no special configurations needed plugin.xml file.

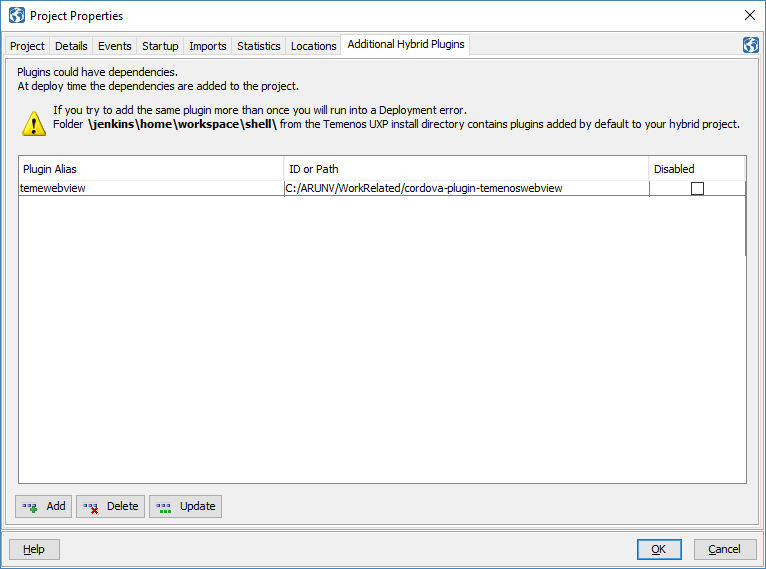
## Installation

Please note that the minimum iOS Cordova version should be 4.0 and the minimum IOS SDK version is 9.0. UXP latest versions should work fine without any issues. Cordova CLI version used for testing was 7.0.

Add Cordova plugin via UXP developer IDE. Installation will happen during the build process. Below are the steps.

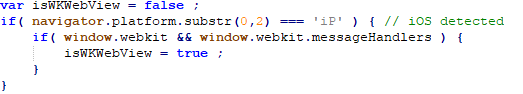
* Copy cordova-plugin-temenoswebview in your local computer (e.g. D drive)
* In UXP developer IDE, go to Project 🡪 Properties…🡪Additional Hybrid plugins
* Click Add button. In Plugin Properties Window, give a desired alias and then choose Path radio button. Navigate to the cordova-plugin-temenoswebviewfolder on D drive you copied earlier. Please find the below screen shot for reference.





## Verifying the usage of WKWebView

We can verify whether the app is running on top of WKWebViewEngine by the help of simple widget in UXP. Below is the sample script we could use in a UXP project.

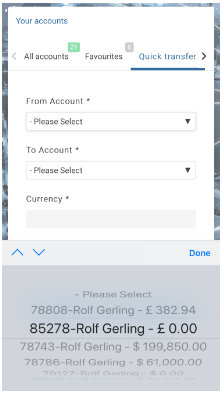
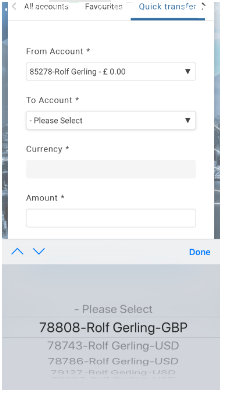
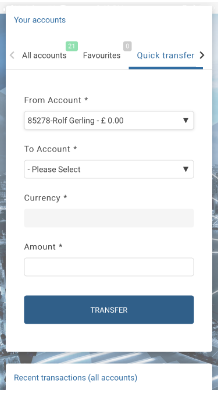


Window.webkit namespace is not available with UIWebView. So, the above variable “isWKWebView” will be false in UIWebView.

Please find the attached Jenkins log while using this plugin.



Below are some screen-print which shows Temenos WebView in action over TCMB application.

# Lifecycle of the application

In our hybrid app architecture, we need to communicate between the native code (ObjectiveC) and the web app (html/javascript). WKWebView in the native app can invoke a javascript method by calling a native method “evaluateJavaScript:<script> completionHandler:^(id result, NSError \*error)*”*

We have a custom method to perform this operation:

(NSString \*)stringByEvaluatingJavaScriptFromString:(NSString \*)script {}

## WKWebView didFailNavigation event handler

- (void)webView:(WKWebView\*)theWebView didFailNavigation:(WKNavigation\*)navigation withError:(NSError\*)error

This is called when the navigation did not complete successfully, for example when we are in the online web app and the device goes offline. In this handler we transition to the offline portion of the app, by loading the main offline page.

## WKWebView didFinishNavigation event handler

- (void)webView:(WKWebView\*)webView didFinishNavigation:(WKNavigation\*)navigation

This event occurs when a page is done loading.

There are 3 distinct situations:

1. The WebView has done loading the index-hybrid.html, on application startup. In this case we invoke a javascript function (loadHybridApp) in index-hybrid.html:

NSString \*setHybridURLoCall = [NSString stringWithFormat:@"loadHybridApp('%@', '%@', '%@');", self.applicationUrl, self.localeAlias, self.pushNotifications ];

[self.webView stringByEvaluatingJavaScriptFromString:setHybridURLoCall];

1. The WebView has done loading an online page, in which case the javascript function window.getRefreshInfo() is invoked. The result is stored in the variables self.pageKey and self.pageVal. These are the “page key” and “page value” variables from UXP. They are used for maintaining an internal stack of the visited pages.
2. The WebView has done loading an offline page. Now we are in the offline portion of the application. But, if we want to go back online, we will need the “page key”, “page value” and the cookies that were used for displaying the last online page. If we don’t set those variables right, we might end up in a session expired page or in an error page. The javascript function setRefreshInfo() is invoked for setting the “page key” and “page value” variables.

## WKWebView didFailProvisionalNavigation event handler

- (void)webView:(WKWebView\*)theWebView didFailProvisionalNavigation:(WKNavigation\*)navigation withError:(NSError\*)error

This event is called when an error occurs while the web view loading content.

## WKWebView didFailNavigation event handler

- (void)webView:(WKWebView\*)theWebView didFailNavigation:(WKNavigation\*)navigation withError:(NSError\*)error

This event is called when an error occurs during navigation. An active connection will be terminated in this case and corresponding error message are shown to the user.

## WKWebView decidePolicyForNavigationAction event handler

This event handle will Routes a navigation action internally or to an external viewer. This is a place where we can check for network connection before we actually navigate to any phase.

## WKWebView didReceiveScriptMessage event handler

This event is invoked when a script message is received from a webpage. The webpage can define a function like:

window.webkit.messageHandlers.<messageName>.postMessage("Hello, world!");

For each handler you want to add, call [add(\_:name:)](https://developer.apple.com/documentation/webkit/wkusercontentcontroller/1537172-add) on your *WKUserContentController* object.