Migrating existing Cordova and hybrid applications

Overview

To migrate an existing Cordova or hybrid application that was created with IBM MobileFirst Foundation version 6.2.0 or later, you must create a Cordova project that uses the plug-ins from the current version. Then you replace the client-side APIs that are discontinued or not in v8.0. The migration assistance tool can help you in this task.

Jump to

Comparison of Cordova apps developed with v8.0 versus v7.1 and before

Compare Cordova apps developed with IBM MobileFirst Foundation v8.0 and Cordova and hybrid apps developed with IBM MobileFirst Platform Foundation v7.1.

Feature	Cordova app with IBM MobileFirst Foundation v8.0	Cordova app with IBM MobileFirst Platform Foundation v7.1	MobileFirst hybrid app with IBM MobileFirst Platform Foundation V7.1
IDE Eclipse Studio			
Eclipse plug-in and integration	Yes	Unsupported	Yes (Proprietary)
Application Components	Yes (Cordova) Note: Create your own Cordova plug-ins to manage application components in your organization.	Yes (Cordova) Note: Create your own Cordova plug-ins to manage application components in your organization.	Yes (Proprietary)
Project Templates	Yes (Cordova) Note: Use the Apache Cordova `cordova create --template` command.	Yes (Cordova) Note: Use `mfp cordova create --template` or the Apache Cordova command `cordova create --copy-from`	Yes (Proprietary)
Dojo and jQuery IDE instrumentation	Yes Note: Dojo and jQuery Mobile are JavaScript frameworks that you can use in Cordova apps.	Yes Note: Dojo and jQuery Mobile are JavaScript frameworks that you can use in Cordova apps.	Yes
Mobile UI Patterns	Unsupported	Unsupported	Deprecated
Application sub types			
Shell Component	Unsupported Note: If the previous Hybrid app used shells and inner applications, it is recommended to adopt Cordova design patterns and implement the shell components as Cordova plug-ins, that can be shared across applications.	Unsupported	Yes
Inner Hybrid Application	Unsupported Note: If the previous Hybrid app used shells and inner applications, it is recommended to adopt Cordova design patterns and implement the shell components as Cordova plug-ins, that can be shared across applications.	Unsupported	Yes
Application Features			
Mobile OS	iOS 8 or higher, Android 4.1 or higher, Windows Phone 8.1, Windows Phone 10.	iOS 7 or higher, Android 4 or higher.	iOS, Android, and Windows Phone 8
Web applications	Yes, as a JavaScript application developed without Apache Cordova.	Unsupported	Yes, as a desktopbrowser or mobilewebapp environment.
Direct Update	Yes.	Yes	Yes
MobileFirst Security Framework	Yes	Yes	Yes
Application Authenticity	Yes	Yes	Yes

Feature	Cordova app with IBM MobileFirst Foundation v8.0	Cordova app with IBM MobileFirst Platform Foundation v7.1	MobileFirst hybrid app with IBM MobileFirst Platform Foundation V7.1
Certificate pinning	Yes	No	Yes
JSONStore	Yes.	Use the cordova-plugin-mfp-jsonstore plug-in.	Yes
FIPS 140-2	Yes. Use the cordova-plugin-mfp-fips plug-in. Restriction: FIPS is supported for Android and iOS. FIPS is not supported for Windows.	No	Yes
Encryption of web resources that are associated with the application within the application binary file	Yes	No	Yes
Verification of the integrity of web resources by using a checksum each time the app starts running	Yes	Unsupported	Yes
Specification of the app's target category (B2E or B2C) for addressable device license tracking	Yes	No	Yes
Simple data sharing	No	Yes	Yes
Single sign-on	Yes Note: Device single sign-on (SSO) is now supported by way of the new predefined enableSSO security-check application-descriptor configuration property	Yes	Yes
MobileFirst application skins	No Note: To detect and handle different device screen sizes, use standard web development practices such as responsive web design	No Note: To detect and handle different device screen sizes, use standard web development practices such as responsive web design.	Yes
Environment optimizations	Yes (Cordova).	Use the merges directory to define web resources specific to a platform.	Yes (Cordova). Use the merges directory to define web resources specific to a platform. For more information, see Using merges to Customize Each Platform in the Apache Cordova documentation.
Push Notifications	Yes. Use the cordova-plugin-mfp-push plug-in. Restriction: You can map predefined MobileFirst security checks only to the push.mobileclient scope. Custom security checks are not supported because JavaScript challenge handlers are not called.	Yes Note: For Android, you must add the cordova-plugin-mfp-push plug in. You don't need this plug in for iOS because the push client-side support for iOS is included in the core mfp plugin.	Yes
Cordova plug-ins management	Yes	Yes	No
MESSAGES (i18n)	Yes	Yes	Yes
Token licensing	Yes	Yes	Yes
Application optimizations			
Minification	Yes (Cordova) Note: Use common open source tools.	Yes (Cordova) Note: Use common open source tools.	Yes (Proprietary)
Concatenation of JS and CSS	Yes (Cordova) Note: Use common open source tools.	Yes (Cordova) Note: Use common open source tools.	Yes (Proprietary)
Obfuscation	Yes (Cordova) Note: Use common open source tools.	Yes (Cordova) Note: Use common open source tools.	Yes (Proprietary)

Feature	Cordova app with IBM MobileFirst Foundation v8.0	Cordova app with IBM MobileFirst Platform Foundation v7.1	MobileFirst hybrid app with IBM MobileFirst Platform Foundation V7.1
Android Pro Guard	Yes Note: IBM MobileFirst Platform Foundation V8.0.0 does not include the predefined proguard-project.txt configuration file for Android ProGuard obfuscation with a MobileFirst Android application.	Yes Note: See Android documentation to enable Pro Guard.	Yes

Migrating existing hybrid or cross-platform apps to Cordova apps supported by MobileFirst Foundation 8.0

You can migrate existing hybrid or cross-platform (Cordova) apps that were developed with IBM MobileFirst™ Platform Foundation version 6.2 or later to Cordova apps that are supported by IBM MobileFirst Foundation v8.0.

Jump to

Starting the Cordova app migration with the migration assistance tool

The migration assistance tool helps you prepare your cross-platform apps that were created with earlier versions of IBM MobileFirst Foundation for migraiton by identifying APIs that are no longer valid and copying the projects into Cordova apps that are supported by v8.0.

The following information is important to know before you use the migration assistance tool:

- You must have an existing IBM MobileFirst Platform Foundation hybrid application or a Cordova application that you created with the `mfp cordova create` command.
- You must have internet access.
- You must have node.js version 4.0.0 or later installed.
- You must have the Cordova Command-Line Interface (CLI) installed, and any prerequisites installed that are required for using the Cordova CLI for your target platforms. For more information, see The Command-Line Interface (http://cordova.apache.org/docs/en/5.1.1/guide/cli/index.html) at the Apache Cordova website.
- Review and understand the limitations of the migration process. For more information, see Migrating apps from earlier releases (../).

Cross-platform apps that were created with earlier versions of IBM MobileFirst Platform Foundation commands or the Cordova with IBM MobileFirst Platform Foundation commands are not supported in version 8.0 without some changes. The migration assistance tool simplifies the process with the following functions:

- Scans the JavaScript files in the existing hybrid app or Cordova with IBM MobileFirst Platform Foundation app and identifies APIs that are deprecated, no longer supported, or modified in version 8.0.
- Copies the structure, script, and configuration files of the initial hybrid app or Cordova with IBM MobileFirst Platform Foundation app to a Cordova structure that is supported in version 8.0.

The migration assistance tool does not modify or move any developer code or comments of your app. You must continue the migration process with either Completing migration of a MobileFirst hybrid app or Completing migration of a MobileFirst Cordova app after you run this tool.

1. Download the migration assistance tool by using one of the following methods:
   - Download the .tgz file from the Jazzhub repository (https://hub.jazz.net/project/ibmmfpf/mfp-migrator-tool).
   - Download the Developer Kit, which contains the migration assistance tool as a file named mfpmigrate-cli.tgz, from the MobileFirst Operations Console.
   - Download the tool by using the instructions that are provided.
2. Install the migration assistance tool.
   - Change to the directory where you downloaded the .tgz file.
   - Use NPM to install the tool by entering the following command:

   ```
   npm install -g tgz_filename
   ```

3. Scan and copy the IBM MobileFirst Platform Foundation app by entering the following command:

   ```
   mfpmigrate client --in rource_directory --out destination_directory --projectName new-project-directory
   ```

   - **source_directory**
     The current location of the project that you are migrating. In Hybrid applications, this should point to the **application** folder of the application.
   - **destination_directory**
     The optional name of the directory where the new version 8.0 compatible Cordova structure is output. This directory is a parent of the **new-project-directory** folder. If it is not specified, then the folder is created in the directory where the command is run.
   - **new-project-directory** The optional name of the folder where the new content of your project is located. This folder is located within the *destination_directory* folder and contains all of the information for your Cordova app. If this option is not specified, the default name is `app_name-app_id-version`.

     When it is used with the client command, the migration assistance tool completes the following actions:
     - Identifies APIs in the existing IBM MobileFirst Platform Foundation app that are removed, deprecated, or changed in version 8.0.
     - Creates a Cordova structure based on the structure of the initial app.
     - Copies or adds the following items, when applicable:
       - Android operating system
       - iPhone and iPad operating system
       - Windows operating system
       - Cordova-mfp-plugin
       - Cordova-plugin-mfp-jsonstore plug-in, if the JSONStore feature was installed on the old project.
       - Cordova-plugin-mfp-fips plug-in, if the FIPS feature was installed on the old project.
       - Cordova-plugin-mfp-push plug-in, if the push notification feature was installed on the old project.

- Hybrid certificates, if certificate pinning was enabled on the old project.
- Application, script, and XML files

> **Important:** The migration assistance tool does not copy developer code or commented text into the new structure.

4. Resolve the API issues in the new Cordova app.
   - Review the **api-report.html** file that is created in the **destination_directory** directory. Each row of the table in this file identifies a deprecated, changed, or removed API that is used in the app that is not compatible with version 8.0. This file also specifies the replacement for removed APIs, when one is available.

File path	Line number	API	Line content	Category of API change	Description and action item
c:\local\Cordova\www\js\index.js	15	`WL.Client.getAppProperty`	• `document.getElementById('app_version')` • `textContent = WL.Client.getAppProperty("APP_VERSION");`	Not supported	Removed from 8.0. Use Cordova plug-in to get app version. No replacement API.

   - Address the API issues that are identified in the **api-report.html** file.
5. Manually copy the developer code from the initial app structure into the correct location in the new Cordova structure. Copy the content in the following directories, according to the type of the source IBM MobileFirst Platform Foundation app:
   - **IBM MobileFirst Platform Foundation hybrid app**
     Copy the contents of the **common** directory of the source app to the **www** directory in your new Cordova app.
   - **Cordova with IBM MobileFirst Platform Foundation app** Copy the contents of the **www** directory of the source app to the **www** directory in your new Cordova app.
6. Run the migration assistance tool with the scan command on your new app to confirm that your API changes are complete.
   - Enter the following command to run the scan:

     ```
     mfpmigrate scan --in source_directory --out destination_directory --type hybrid
     ```

     - **source_directory**
       The current location of the files to scan. In an IBM MobileFirst Platform Foundation hybrid app, this location is the **common** directory of your app. In an IBM MobileFirst Foundation version 8.0 Cordova cross-platform app, this location is the **www** directory.
     - **destination_directory**
       The directory where your scan results are output.
     - **scan_type**
       The type of project to scan.
   - Address any remaining API issues that are identified in the **api-report.html** file.
7. Repeat step 6 to run the scan tool on the new Cordova app until all of the issues are resolved.

Completing migration of a MobileFirst hybrid app

After you use the migration assistance tool, you must modify some portions of your code manually to complete the migration process.

- You must have already run the mfpmigrate migration assistance tool on your existing hybrid app. For more information, see Starting the Cordova app migration with the migration assistance tool.
- You must have the Cordova Command-Line Interface (CLI) installed, and any prerequisites installed that are required for using the Cordova CLI for your target platforms if you need to install any additional Cordova plug-ins. (See step 6.) For more information, see The Command-Line Interface (http://cordova.apache.org/docs/en/5.1.1/guide/cli/index.html) at the Apache Cordova web site.
- You must have internet access if you need to download a new version of JQuery (step 1c) or if you need to install any additional Cordova plug-ins (step 6).
- You must have node.js version 4.0.0 or later installed if you need to install additional Cordova plug-ins (step 6).

Complete the steps in this task to finish migrating your MobileFirst hybrid application from IBM MobileFirst Platform Foundation 7.1 to a Cordova application that includes support for IBM MobileFirst Platform Foundation 8.0.

After you complete the migration, your app can use Cordova platforms and plug-ins that you obtain independently of IBM MobileFirst Platform Foundation, and you can continue to develop the app with your preferred Cordova development tools.

1. Update the **www/index.html** file.
   - Add the following CSS code to the head of your index.html file, before your CSS code that is already there.

     ```
     <link rel="stylesheet" href="worklight/worklight.css">
     <link rel="stylesheet" href="css/main.css">
     ```

     > **Note:** The **worklight.css** file sets the body attribute to relative. If this affects the style of your app, then declare a different value for the position in your own CSS code. For example:

     ```
     body {
       position: absolute;
     }
     ```
   - Add Cordova JavaScript to the head of the file after the CSS definitions.

     ```
     <script type="text/javascript" src="cordova.js"></script>
     ```

- Remove the following line of code if it is present.

  ```
  <script>window.$ = window.jQuery = WLJQ;</script>
  ```

  You can download your own version of JQuery, and load it as shown in the following code line.

  ```
  <script src="lib/jquery.min.js"></script>
  ```

  You do not have to move the optional jQuery addition to the **lib** folder. You can move this addition anywhere you want to, but you must correctly reference it in the **index.html** file.

2. Update the **www/js/InitOptions.js** file to call `WL.Client.init` automatically.
   - Remove the following code from **InitOptions.js**

     The function `WL.Client.init` is called automatically with the global variable **wlInitOptions**.

     ```
     if (window.addEventListener) {
         window.addEventListener('load', function() { WL.Client.init(wlInitOptions); }, false);
     } else if (window.attachEvent) {
         window.attachEvent('onload',  function() { WL.Client.init(wlInitOptions); });
     }
     ```

3. Optional: Update the **www/InitOptions.js** to call `WL.Client.init` manually.
   - Edit the **config.xml** file and set the `<mfp:clientCustomInit>` element's enabled attribute to true.
   - If you are using the MobileFirst hybrid default template, replace this code:

     ```
     if (window.addEventListener) {
         window.addEventListener('load', function() { WL.Client.init(wlInitOptions); }, false);
     } else if (window.attachEvent) {
         window.attachEvent('onload',  function() { WL.Client.init(wlInitOptions); });
     }
     ```

     with the following code:

     ```
     if (document.addEventListener) {
         document.addEventListener('mfpready', function() { WL.Client.init(wlInitOptions); }, false);
     } else if (window.attachEvent) {
         document.attachEvent('mfpready',  function() { WL.Client.init(wlInitOptions); });
     }
     ```

4. Optional: If you have logic specific to a hybrid environment, for example in Your **app/iphone/js/main.js**, copy the function `wlEnvInit()` and append it at the end of **www/main.js**.

   ```
   // This wlEnvInit method is invoked automatically by MobileFirst runtime after successful initialization.
   function wlEnvInit() {
       wlCommonInit();
       if (cordova.platformId === "ios") {
           // Environment initialization code goes here for ios
       } else if (cordova.platformId === "android") {
           // Environment initialization code goes here for android
       }
   }
   ```

5. Optional: If your original application uses the FIPS feature, change the JQuery event listener to a JavaScript event listener that listens to the WL/FIPS/READY event. For more information about FIPS, see FIPS 140-2 support (../../../administering-apps/federal/#fips-140-2-support).
6. Optional: If your original application uses any third-party Cordova plug-ins that are not replaced or supplied by the migration assistance tool, manually add the plug-ins to the Cordova app with the `cordova plugin add` command. For information about which plug-ins are replaced by the tool, see Starting the Cordova app migration with the migration assistance tool.

Completing migration of a MobileFirst Cordova app

After you use the migration assistance tool, you must modify some portions of your code manually to complete the migration process.

- You must have already run the **mfpmigrate** migration assistance tool on your existing Cordova app. For more information, see Starting the Cordova app migration with the migration assistance tool.
- You must have the Cordova Command-Line Interface (CLI) installed, and any prerequisites installed that are required for using the Cordova CLI for your target platforms. For more information, see The Command-Line Interface (http://cordova.apache.org/docs/en/5.1.1/guide/cli/index.html) at the Apache Cordova web site.
- You must have internet access.
- You must have node.js version 4.0.0 or later installed.

The Cordova app that you created with **mfp cordova create** uses the Cordova platform and plug-in versions that were supplied with IBM MobileFirst Platform Foundation previous version. After you complete the migration, your migrated app can use Cordova platforms and plug-ins that you obtain independently of IBM MobileFirst Platform Foundation. This is the only type of support for Cordova applications that is available with IBM MobileFirs Foundation v8.0.

To migrate, you run the migration assistance tool and then make other modifications to your app.

1. With the Cordova development tool of your choice, add any Cordova plug-ins other than Cordova plug-ins that enable MobileFirst features that were in your original application. For example, with the Cordova CLI, to add the plug-ins **cordova-plugin-file** and **cordova-plugin-file-transfer**, enter:

   ```
   cordova plugin add cordova-plugin-file cordova-plugin-file-transfer
   ```

   > **Note:** The **mfpmigrate** migration assistance tool added the Cordova plug-ins for MobileFirst features, so you do not have to add them. For more information about these plug-ins, see Cordova plug-ins for MobileFirst (../../../application-development/sdk/cordova).

2. Optional: If your original application uses the FIPS feature, change the JQuery event listener to a JavaScript event listener that listens to the WL/FIPS/READY event. For more information about FIPS, see FIPS 140-2 support (../../../administering-apps/federal/#fips-140-2-support).
3. Optional: If your original application uses any third-party Cordova plug-ins that are not replaced or supplied by the migration assistance tool, manually add the plug-ins to the Cordova app with the **cordova plugin add** command. For information about which plug-ins are replaced by the tool, see Starting the Cordova app migration with the migration assistance tool.
4. Optional: (Only for apps that include the iOS platform, and that use OpenSSL.) Add the **cordova-plugin-mfp-encrypt-utils** plug-in to your app. The **cordova-plugin-mfp-encrypt-utils** plug-in provides iOS OpenSSL frameworks for encryption for Cordova applications with the iOS platform.

You now have a Cordova app that you can continue to develop with your preferred Cordova tools, but that also includes MobileFirst functionality.

Migrating encryption for iOS Cordova

If your iOS Hybrid or Cordova application used OpenSSL encryption, you may want to migrate your app to the new V8.0.0 native encryption. If you want to continue using OpenSSL you need to add an additional Cordova plug-in.

For more information on the iOS Cordova encryption options for migration see the Migration options (../../../application-development/sdk/cordova/additional-information/#migration-options) section within the Enabling OpenSSL in Cordova Applications (../../../application-development/sdk/cordova/additional-information/#enabling-openssl-in-cordova-applications) topic.

Migrating Direct Update

Direct Update is triggered after the first access to a protected resource. The process to deploy new web resources has changed in v8.0.

Unlike in previous versions, in v8.0, if an application does not access a secure MobileFirst resource, the client application does not receive updates, even if updates are available on the server. A resource might be unprotected, for example because OAuth has been disabled by the annotation `@OAuth(security=false)` or by configuration. You can work around this risk in one of the following ways:

- Explicitly obtain an access token. See the `obtainAccessToken` API in the `WLAuthorizationManager` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/html/refjavascript-client/html/WLAuthorizationManager.html?view=kc) class.
- Call another protected resource. See the `WLResourceRequest` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/html/refjavascript-client/html/WLResourceRequest.html?view=kc) class.

To use Direct Update: Starting with v8.0, you no longer upload a **.wlapp** file to MobileFirst Server. Instead, you upload a smaller web resource archive (.zip file). The archive file no longer contains the web preview files or skins that were widely used in previous versions. These have been discontinued. The archive contains only the web resources that are sent to the clients, as well as checksums for Direct Update validations.

> For more information, see the Direct Update documentation (../../../application-development/direct-update).

Upgrading the WebView

IBM MobileFirs Foundation v8.0 Cordova SDK (JavaScript) introduced numerous changes that require adaptations of your code.

The manual migration process involves a few stages:

- Creating a new Cordova MobileFirst project
- Replacing the necessary web resource elements with the code from your previous version
- Making the necessary changes to your JavaScript code to conform to SDK changes

Many MobileFirst API elements were removed in v8.0. Removed elements are clearly marked as non-existent in an IDE that supports autocorrect for JavaScript.

The table below lists those API elements that require removal, with suggestions on how to replace the functionality. Many of the removed elements are UI elements that can be replaced with Cordova plug-ins or HTML 5 elements. Some methods have changed.

Discontinued JavaScript UI elements

API element	Migration path
• `WL.BusyIndicator` • `WL.OptionsMenu` • `WL.TabBar` • `WL.TabBarItem`	Use Cordova plug-ins or HTML 5 elements.
`WL.App.close()`	Handle this event outside of MobileFirst.
`WL.App.copyToClipboard()`	Use Cordova plug-ins providing this functionality.
`WL.App.openUrl(url, target, options)`	Use Cordova plug-ins providing this functionality. Note: For your information, the Cordova InAppBrowser plug-in provides this feature.
• `WL.App.overrideBackButton(callback)` • `WL.App.resetBackButton()`	Use Cordova plug-ins providing this functionality. Note: For your information, the Cordova backbutton plug-in provides this feature.
`WL.App.getDeviceLanguage()`	Use Cordova plug-ins providing this functionality. Note: For your information, the Cordova **cordova-plugin-globalization** plug-in provides this feature.
`WL.App.getDeviceLocale()`	Use Cordova plug-ins providing this functionality. Note: For your information, the Cordova **cordova-plugin-globalization** plug-in provides this feature.
`WL.App.BackgroundHandler`	To run a custom handler function, use the standard Cordova pause event listener. Use a Cordova plug-in that provides privacy and prevents iOS and Android systems and users from taking snapshots or screen captures. For more information, see the description of the PrivacyScreenPlugin at https://github.com/devgeeks/PrivacyScreenPlugin (https://github.com/devgeeks/PrivacyScreenPlugin).

API element	Migration path
• `WL.Client.close()` • `WL.Client.restore()` • `WL.Client.minimize()`	The functions were provided to support the Adobe AIR platform, which is not supported by IBM MobileFirst Foundation v8.0
`WL.Toast.show(string)`	Use Cordova plug-ins for Toast.

Other Discontinued JavaScript elements

API	Migration path
`WL.Client.checkForDirectUpdate(options)`	No replacement. Note: You can call `WLAuthorizationManager.obtainAccessToken` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WLAuthorizationManager.html?view=kc#obtainAccessToken) to trigger a direct u access to a security token triggers a direct update if one is available on the server. But you c demand.
• `WL.Client.setSharedToken({key: myName, value: myValue})` • `WL.Client.getSharedToken({key: myName})` • `WL.Client.clearSharedToken({key: myName})`	No replacement.
• `WL.Client.isConnected()` • `connectOnStartup` init option	Use `WLAuthorizationManager.obtainAccessToken` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WLAuthorizationManager.html?view=kc#obtainAccessToken) to check connectivi application management rules.
• `WL.Client.setUserPref(key,value, options)` • `WL.Client.setUserPrefs(userPrefsHash, options)` • `WL.Client.deleteUserPrefs(key, options)`	No replacement. You can use an adapter and the MFP.Server.getAuthenticatedUser (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d server/html/MFP.Server.html?view=kc#MFP.Server.getAuthenticatedUser) API to manage u
• `WL.Client.getUserInfo(realm, key)` • `WL.Client.updateUserInfo(options)`	No replacement.
`WL.Client.logActivity(activityType)`	Use `WL.Logger` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WL.Logger.html?view=kc)
`WL.Client.login(realm, options)`	Use `WLAuthorizationManager.login` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WLAuthorizationManager.html?view=kc#login).
`WL.Client.logout(realm, options)`	Use `WLAuthorizationManager.logout` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WLAuthorizationManager.html?view=kc#logout).
`WL.Client.obtainAccessToken(scope, onSuccess, onFailure)`	Use `WLAuthorizationManager.obtainAccessToken` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WLAuthorizationManager.html?view=kc#obtainAccessToken).
• `WL.Client.transmitEvent(event, immediate)` • `WL.Client.purgeEventTransmissionBuffer()` • `WL.Client.setEventTransmissionPolicy(policy)`	Create a custom adapter for receiving notifications of these events.
• `WL.Device.getContext()` • `WL.Device.startAcquisition(policy, triggers, onFailure)` • `WL.Device.stopAcquisition()` • `WL.Device.Wifi` • `WL.Device.Geo.Profiles` • `WL.Geo`	Use native API or third-party Cordova plug-ins for GeoLocation.
`WL.Client.makeRequest (url, options)`	Create a custom adapter that provides the same functionality
`WL.Device.getID(options)`	Use Cordova plug-ins providing this functionality. Note: For your information, **device.uuid** from the **cordova-plugin-device** plug-in provides th
`WL.Device.getFriendlyName()`	Use `WL.Client.getDeviceDisplayName`
`WL.Device.setFriendlyName()`	Use `WL.Client.setDeviceDisplayName`
`WL.Device.getNetworkInfo(callback)`	Use Cordova plug-ins providing this functionality. Note: For your information, the **cordova-plugin-network-information** plug-in provides this f
`WLUtils.wlCheckReachability()`	Create a custom adapter to check server availability.
`WL.EncryptedCache`	Use JSONStore to store encrypted data locally. JSONStore is in the **cordova-plugin-mfp-js**

API	Migration path
`WL.SecurityUtils.remoteRandomString(bytes)`	Create a custom adapter that provides the same functionality.
`WL.Client.getAppProperty(property)`	You can retrieve the app version property by using the cordova plugin add **cordova-plugin-** that is returned is the native app version (Android and iOS only).
`WL.Client.Push.*`	Use JavaScript client-side push API (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d view=kc#r_client_push_api) from the **cordova-plugin-mfp-push** plug-in. For more informati notifications from event source-based notifications (../../migrating-push-notifications).
`WL.Client.Push.subscribeSMS(alias, adapterName, eventSource, phoneNumber, options)`	Use `MFPPush.registerDevice(org.json.JSONObject options, MFPPushResponseLi` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d hybrid/html/MFPPush.html?view=kc#registerDevice) to register the device for push and SMS
`WLAuthorizationManager.obtainAuthorizationHeader(scope)`	Use `WLAuthorizationManager.obtainAccessToken` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WLAuthorizationManager.html?view=kc#obtainAccessToken) to obtain a token fo
`WLClient.getLastAccessToken(scope)`	Use `WLAuthorizationManager.obtainAccessToken` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WLAuthorizationManager.html?view=kc#obtainAccessToken)
• `WLClient.getLoginName()` • `WL.Client.getUserName(realm)`	No replacement
`WL.Client.getRequiredAccessTokenScope(status, header)`	Use `WLAuthorizationManager.isAuthorizationRequired` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WLAuthorizationManager.html?view=kc#isAuthorizationRequired) and `WLAuthorizationManager.getResourceScope` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WLAuthorizationManager.html?view=kc#getResourceScope).
`WL.Client.isUserAuthenticated(realm)`	No replacement
`WLUserAuth.deleteCertificate(provisioningEntity)`	No replacement
`WL.Trusteer.getRiskAssessment(onSuccess, onFailure)`	No replacement
`WL.Client.createChallengeHandler(realmName)`	To create a challenge handler for handling custom gateway challenges, use `WL.Client.createGatewayChallengeHandler(gatewayName)` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WL.Client.html?view=kc#createGatewayChallengeHandler). To create a challeng security-check challenges, use `WL.Client.createSecurityCheckChallengeHandler(se` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WL.Client.html?view=kc#createSecurityCheckChallengeHandler).
`WL.Client.createWLChallengeHandler(realmName)`	Use `WL.Client.createSecurityCheckChallengeHandler(securityCheckName)` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WL.Client.html?view=kc#createSecurityCheckChallengeHandler).
`challengeHandler.isCustomResponse()` where `challengeHandler` is a challenge-handler object that is returned by `WL.Client.createChallengeHandler()`	Use `gatewayChallengeHandler.canHandleResponse()` where `gatewayChallengeHand` that is returned by `WL.Client.createGatewayChallengeHandler()` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WL.Client.html?view=kc#createGatewayChallengeHandler).
`wlChallengeHandler.processSucccess()` where `wlChallengeHandler` is a challenge-handler object that is returned by `WL.Client.createWLChallengeHandler()`	Use `securityCheckChallengeHandler.handleSuccess()` where `securityCheckChall` handler object that is returned by `WL.Client.createSecurityCheckChallengeHandler(` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WL.Client.html?view=kc#createSecurityCheckChallengeHandler).
`WL.Client.AbstractChallengeHandler.submitAdapterAuthentication()`	Implement similar logic in your challenge handler. For custom gateway challenge handlers, u is returned by `WL.Client.createGatewayChallengeHandler()` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WL.Client.html?view=kc#createGatewayChallengeHandler). For MobileFirst secu challenge-handler object that is returned by `WL.Client.createSecurityCheckChallenge` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WL.Client.html?view=kc#createSecurityCheckChallengeHandler).
`WL.Client.AbstractChallengeHandler.submitFailure(err)`	Use `WL.Client.AbstractChallengeHandler.cancel()` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.d client/html/WL.Client.AbstractChallengeHandler.html?view=kc#cancel).
`WL.Client.createProvisioningChallengeHandler()`	No replacement. Device provisioning is now handled automatically by the security framework

Deprecated JavaScript APIs

API	Migration path
• `WLClient.invokeProcedure(WLProcedureInvocationData invocationData,WLResponseListener responseListener)` • `WL.Client.invokeProcedure(invocationData, options)` • `WLClient.invokeProcedure(WLProcedureInvocationData invocationData, WLResponseListener responseListener, WLRequestOptions requestOptions)` • `WLProcedureInvocationResult`	Use `WLResourceRequest` instead. Note: The implementation of invokeProcedure uses WLResourceRequest.
`WLClient.getEnvironment`	Use Cordova plug-ins providing this functionality. Note: For your information, the device.platform plug-in provides this feature.
`WL.Client.getLanguage`	Use Cordova plug-ins providing this functionality. Note: For your information, the **cordova-plugin-globalization** plug-in provides this feature.
`WL.Client.connect(options)`	Use `WLAuthorizationManager.obtainAccessToken` to check connectivity to the server and apply application management rules.

Removed components

The Cordova project created by MobileFirst Platform Foundation Studio 7.1 included many resources that supported propriety functionality. However in v8.0 only pure Cordova is supported and the MobileFirst API no longer supports these features.

Skins

MobileFirst application skins provided a way of optimizing the UI for adapting to different devices and formats and is no longer supported in v8.0.
To replace this type of functionality it is recommended to adopt responsive web design methods provided by Cordova and HTML 5.

Shells

Shells allowed the development of a set of functionalities to be used by and shared among applications. In this way developers who were more experienced with the native environment could provide a set of core functions. These shells were bundled into **inner applications** and used by developers who are involved with business logic or UI development.

If the previous hybrid app used shells and inner applications, it is recommended to adopt Cordova design patterns and implement the shell components as Cordova plug-ins, that can be shared across applications. Developers may find ways to reuse parts of shell code and migrate them to Cordova plug-ins.

For example, if a customer has a set of web resources (JavaScript, css files, graphics, html) that are common across all their apps they can create a Cordova plug-in that copies these resources into the app's www folder.

Let's say these resources are within the src/www/acme/ folder:

- src/www/acme/js/acme.js
- src/www/acme/css/acme.css
- src/www/acme/img/acme-logo.png
- src/www/acme/html/banner.html
- src/www/acme/html/footer.html
- plugin.xml

The **plugin.xml** file contains the `<asset>` tag, containing the source and target for copying the resources:

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin
    xmlns="http://apache.org/cordova/ns/plugins/1.0"
    xmlns:rim="http://www.blackberry.com/ns/widgets"
    xmlns:android="http://schemas.android.com/apk/res/android"
    id="cordova-plugin-acme"
    version="1.0.1">
<name>ACME Company Shell Component</name>
<description>ACME Company Shell Component</description>
<license>MIT</license>
<keywords>cordova,acme,shell,components</keywords>
<issue>https://www.acme.com/support</issue>
<asset src="src/www/acme" target="www/acme"/>
</plugin>
```

After the **plugin.xml** is added to the Cordova **config.xml** file, the resources listed in the asset src are copied to the asset target during compilation.
Then in their **index.html** file or anywhere inside their app they can reuse these resources.

```
<link rel="stylesheet" type="text/css" href="acme/css/acme.css">
<script type="text/javascript" src="acme/js/acme.js"></script>
<div id="banner"></div>
<div id="app"></div>
<div id="footer"></div>
<script type="text/javascript">
   $("#banner").load("acme/html/banner.html");
   $("#footer").load("acme/html/footer.html");
</script>
```

Settings page

The **settings page** was a UI available in the MobileFirst hybrid app that allowed the developer to change the server URL at runtime for testing purposes. The developer can now use existing MobileFirst Client API to change the server URL at runtime. For more information, see WL.App.setServerUrl (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/html/refjavascript-client/html/WL.App.html?lang=en-

us&cp=SSHS8R_8.0.0&view=kc#setServerUrl).

Minification

MobileFirst Studio 7.1 provided an OOTB method of reducing the size of your JavaScript code by removing all unnecessary characters before compilation. This removed functionality can be replaced by adding Cordova hooks to your project.

Many hooks are available for minifying your Javascript and css files and can be placed in the config.xml at the before_prepare event.

Here are some recommended hooks:

- https://www.npmjs.com/package/uglify-js (https://www.npmjs.com/package/uglify-js)
- https://www.npmjs.com/package/clean-css (https://www.npmjs.com/package/clean-css)

These hooks can be defined in either a plug-in file or in the app's config.xml file, using the `<hook>` elements.
In this example, using the before_prepare hook event, a script is run for minifying before cordova prepare copies the files to each platform's www/ folder:

```
<hook type="before_prepare" src="scripts/uglify.js" />
```

Last modified on