Migrating existing Android applications

Overview

To migrate an existing native Android project that was created with IBM MobileFirst™ Platform Foundation version 6.2.0 or later, you must modify the project to use the SDK from the current version. Then you replace the client-side APIs that are discontinued or not in v8.0. The migration assistance tool can scan your code and generate reports of the APIs to replace.

Jump to

Scanning existing MobileFirst native Android apps to prepare for MobileFirst version 8.0

The migration assistance tool helps you prepare your apps that were created with a previous version of IBM MobileFirst™ Platform Foundation for migration by scanning the sources of the native Android app and generating a report of APIs that are deprecated or discontinued in version 8.0.

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

- You must have an existing IBM MobileFirst Platform Foundation native Android application.
- You must have internet access.
- You must have node.js version 4.0.0 or later installed.
- Review and understand the limitations of the migration process. For more information, see Migrating apps from earlier releases (../).

Apps that were created with previous versions of IBM MobileFirst Platform Foundation are not supported in version 8.0 without some changes. The migration assistance tool simplifies the process by scanning the source files in the existing app and identifies APIs that are deprecated, no longer supported, or modified in version 8.0.

The migration assistance tool does not modify or move any developer code or comments of your app.

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.
2. Install the migration assistance tool.
   - Change to the directory where you downloaded the tool.
   - Use NPM to install the tool by entering the following command:

   ```
   npm install -g
   ```

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

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

   source_directory

   The current location of the project.

   destination_directory

   The directory where the report is created.

   When it is used with the scan command, the migration assistance tool identifies APIs in the existing IBM MobileFirst Platform Foundation app that are removed, deprecated, or changed in version 8.0 and saves them in the identified destination directory.

Migrating an Android project with Gradle

Migrate your Android application with MobileFirst SDK using Gradle.

Ensure that your Android Studio and the Android SDK are set up properly. For more information about how to set up your system, see Android Studio Overview (http://developer.android.com/tools/studio/index.html). Your project must conform to the Android Studio/Gradle setup and compile without errors before you upgrade to IBM MobileFirst Foundation.

> **Note:** This task assumes that the Android project is created with Android Studio and that the MobileFirst SDK is added with as described in Adding the IBM MobileFirst Platform Foundation SDK to a new or existing application with Android Studio (7.1) (https://www.ibm.com/support/knowledgecenter/SSHS8R_7.1.0/com.ibm.worklight.dev.doc/dev/t_dev_new_w_gradle.html).

If your Android Studio project was set up to add a previous version of MobileFirst SDK, remove the **compile** group from the **build.gradle** dependencies enclosure. For example, if you are upgrading from 7.1, remove this group:

```
compile group: 'com.ibm.mobile.foundation',
        name:'ibmmobilefirstplatformfoundation',
        version:'7.1.0.0',
        ext: 'aar',
        transitive: true
```

You can now add the V8.0.0 SDK and configuration, by using local or remote SDK files. See Adding the MobileFirst SDK to Android applications (../../../application-development/sdk/android).

> Note: After you import the new SDK, you need to import the Javadoc files manually. See Registering Javadocs to an Android Studio Gradle project (../../../application-development/sdk/android/additional-information).

You can now start developing your native Android application with the MobileFirst SDK. You might need to adapt your code to changes in the V8.0.0 API (see Updating the Android code).

What to do next

Replace the client-side APIs that are discontinued or not in v8.0.

Updating the Android code

IBM MobileFirst Foundation v8.0 introduces a number of changes to the Android SDK that might require changes to apps developed in earlier versions. The tables below list changes in the MobileFirst Android SDK.

Discontinued Android API elements

API element	Migration path
`WLConfig WLClient.getConfig()`	No replacement.
• `WLDevice WLClient.getWLDevice()` • `WLClient.transmitEvent(org.json.JSONObject event)` • `WLClient.setEventTransmissionPolicy(WLEventTransmissionPolicy policy)` • `WLClient.purgeEventTransmissionBuffer()`	Use Android API or third-party packages for GeoLocation.
• `WL.Client.getUserInfo(realm, key)` • `WL.Client.updateUserInfo(options)`	No replacement
• `WL.Client.getUserInfo(realm, key` • `WL.Client.updateUserInfo(options)`	No replacement
`WLClient.checkForNotifications()`	Use `WLAuthorizationManager.obtainAccessToken("", listener)` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.do native/html/com/worklight/wlclient/api/WLAuthorizationManager.html?view=kc#obtainAccessToken(java.lang.String,%20com.worklight.wlclient.api.WLAccessToken and apply application management rules.
• `WLClient.login(java.lang.String realmName, WLRequestListener listener, WLRequestOptions options)` • `WLClient.login(java.lang.String realmName, WLRequestListener listener)`	Use `AuthorizationManager.login()` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.do native/html/com/worklight/wlclient/api/WLAuthorizationManager.html?view=kc#login(java.lang.String,%20org.json.JSONObject,%20com.worklight.wlclient.api.WLL
• `WLClient.logout(java.lang.String realmName, WLRequestListener listener, WLRequestOptions options)` • `WLClient.logout(java.lang.String realmName, WLRequestListener listener)`	Use `AuthorizationManager.logout()` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.do native/html/com/worklight/wlclient/api/WLAuthorizationManager.html?view=kc#logout(java.lang.String,%20com.worklight.wlclient.api.WLLogoutResponseListener))
`WLClient.obtainAccessToken(java.lang.String scope,WLResponseListener responseListener)`	Use `WLAuthorizationManager.obtainAccessToken(String, WLAccessTokenListener` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.do native/html/com/worklight/wlclient/api/WLAuthorizationManager.html?view=kc#obtainAccessToken(java.lang.String,%20com.worklight.wlclient.api.WLAccessToken and apply application management rules.
• `WLClient.getLastAccessToken()` • `WLClient.getLastAccessToken(java.lang.String scope)`	Use `AuthorizationManager` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8. worklight-android-native/html/com/worklight/wlclient/api/WLAuthorizationManager.html?view=
`WLClient.getRequiredAccessTokenScope(int status, java.lang.String header)`	Use `AuthorizationManager` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8. worklight-android-native/html/com/worklight/wlclient/api/WLAuthorizationManager.html?view=
`WLClient.logActivity(java.lang.String activityType)`	Use `com.worklight.common.Logger` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.do native/html/com/worklight/common/Logger.html?view=kc)
`WLAuthorizationPersistencePolicy`	No replacement. To implement authorization persistence, store the authorization token in the requests. For more information, see Java™ custom resource-request implementation sample (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.dev.doc/ view=kc#c_custom_request_to_resource_hybrid).
• `WLSimpleSharedData.setSharedToken(myName, myValue)` • `WLSimpleSharedData.getSharedToken(myName)` • `WLSimpleSharedData.clearSharedToken(myName)`	Use the Android APIs to share tokens across applications.
`WLUserCertificateManager.deleteCertificate(android.content.Context context)`	No replacement
`BaseChallengeHandler.submitFailure(WLResponse wlResponse)`	Use `BaseChallengeHandler.cancel()` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.do native/html/com/worklight/wlclient/api/challengehandler/BaseChallengeHandler.html?view=kc)
`ChallengeHandler`	For custom gateway challenges, use GatewayChallengeHandler (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.do native/html/com/worklight/wlclient/api/challengehandler/GatewayChallengeHandler.html?view challenges, use SecurityCheckChallengeHandler (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.do native/html/com/worklight/wlclient/api/challengehandler/SecurityCheckChallengeHandler.html
`WLChallengeHandler`	Use SecurityCheckChallengeHandler (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.do native/html/com/worklight/wlclient/api/challengehandler/SecurityCheckChallengeHandler.html
`ChallengeHandler.isCustomResponse()`	Use `GatewayChallengeHandler.canHandleResponse()` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.do native/html/com/worklight/wlclient/api/challengehandler/GatewayChallengeHandler.html?view
`ChallengeHandler.submitAdapterAuthentication`	Implement similar logic in your challenge handler. For custom gateway challenge handlers, us (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.do native/html/com/worklight/wlclient/api/challengehandler/GatewayChallengeHandler.html?view

Android APIs depending on the legacy `org.apach.http` APIs are no longer supported

API element	Migration path
`org.apache.http.Header[]` is now deprecated. Therefore, the following methods are removed:	
`org.apache.http.Header[] WLResourceRequest.getAllHeaders()`	Use instead the new `Map<String, List<String>> WLResourceRequest.getAllHeaders()` API.

API element	Migration path
`WLResourceRequest.addHeader(org.apache.http.Header header)`	Use instead the new `WLResourceRequest.addHeader(String name, String value)` API.
`org.apache.http.Header[] WLResourceRequest.getHeaders(java.lang.String headerName)`	Use instead the new `List<String> WLResourceRequest.getHeaders(String headerName)` API.
`org.apache.http.Header WLResourceRequest.getFirstHeader(java.lang.String headerName)`	Use instead the new `WLResourceRequest.getHeaders(String headerName)` API.
`WLResourceRequest.setHeaders(org.apache.http.Header[] headers)`	Instead, use the new `WLResourceRequest.setHeaders(Map<String, List<String>> headerMap)` API.
`WLResourceRequest.setHeader(org.apache.http.Header header)`	Instead, use the new `WLResourceRequest.setHeaders(Map<String, List<String>> headerMap)` API.
`org.apache.http.client.CookieStore WLClient.getCookieStore()`	Replaced with `java.net.CookieStore getCookieStore WLClient.getCookieStore()`
`java.net.CookieStore getCookieStore WLClient.getCookieStore()`	No replacement. MFP Client allows circular redirects.
`WLClient.setAllowHTTPClientCircularRedirect(boolean isSet)` • `WLHttpResponseListener` • `WLResourceRequest`, all methods that take `WLHttpResponseListener`: ◦ `WLResourceRequest.send(java.util.HashMap formParameters,WLHttpResponseListener listener)` ◦ `WLResourceRequest.send(org.json.JSONObject json, WLHttpResponseListener listener)` ◦ `WLResourceRequest.send(byte[] data, WLHttpResponseListener listener)` ◦ `WLResourceRequest.send(java.lang.String requestBody,WLHttpResponseListener listener)` ◦ `WLResourceRequest.send(WLHttpResponseListener listener)` ◦ `WLClient.sendRequest(org.apache.http.client.methods.HttpUriRequest request,WLHttpResponseListener listener)` ◦ `WLClient.sendRequest(org.apache.http.client.methods.HttpUriRequest request, WLResponseListener listener)`	Removed due to deprecated Apache HTTP Client dependencies. Create your own request to have full control over the request and response.