Migrating from Earlier Releases

Overview

IBM MobileFirst Foundation v8.0 introduces new concepts for application development and deployment, and some API changes. Learn about these changes to prepare and plan for the migration of your MobileFirst applications.

> Review the Migration Cookbook (migration-cookbook) to quickly get started with the migration process.

Jump to

Changes in the development and deployment process

> For a quick hands-on experience of the development process with IBM MobileFirst Platform Foundation V8.0.0, you can review the Quick Start tutorials (../quick-start).

In this version of the product, you no longer create a project WAR file that needs to be installed in the application server that is running MobileFirst Server before you can upload your apps. Instead, MobileFirst Server is installed once, and you upload the server-side **configuration** of your apps, of the resource security, or of the push service to the server. You can modify the configuration of your apps with the MobileFirst Operations Console. You can also upload a new **configuration file** for your apps by using a command-line tool or the server REST API.

MobileFirst projects no longer exist. Instead, you develop your mobile app with the development environment of your choice. You develop the server-side of your application separately, in Java™ or in JavaScript. You can develop adapters with Apache Maven or a Maven enabled IDE such as Eclipse, IntelliJ, and others.

In previous versions, applications were deployed to the server by uploading a .wlapp file. The file contained data that described the application and for hybrid applications, the web resources. In v8.0, the .wlapp file is replaced by an application descriptor JSON file for registering an app to the server. For Cordova applications that use Direct Update, instead of uploading a new version of the .wlapp, you now upload a web resource archive to the server.

When you develop your app, you use the IBM MobileFirst Platform Command Line Interface (CLI) for many tasks, such as registering an app to its target server or uploading its server-side configuration.

Discontinued features and replacement path

IBM MobileFirst Foundation V8.0.0 is radically simplified compared to the previous version. As a result of this simplification, some features that were available in V7.1 are discontinued in v8.0.

> For more information about discontinued features and replacement path, seeFeatures that are discontinued in v8.0 and features that are not included in v8.0 (../product-overview/release-notes/deprecated-discontinued).

Migrating a Cordova or hybrid application

You start developing Cordova apps with the Apache Cordova command-line tool or with a Cordova enabled IDE such as Visual Studio Code, Eclipse, IntelliJ, and others.

Add support for the MobileFirst features by adding the MobileFirst plug-ins to your app. For more information about the differences between V7.1 Cordova or hybrid apps and V8.0 Cordova apps, see Comparison of Cordova apps developed with v8.0 versus v7.1 and before (migrating-client-applications/cordova/#comparison-of-cordova-apps-developed-with-v80-versus-v71-and-before).

To migrate a Cordova or hybrid app, you need to

- For planning purposes, run the migration assistance tool on your existing project. Review the generated report and assess the effort required for migration. For more information, see Starting the Cordova app migration with the migration assistance tool (migrating-client-applications/cordova/#starting-the-cordova-app-migration-with-the-migration-assistance-tool).
- Replace the client-side APIs that are discontinued or not in V8.0.0. For a list of API changes, see Upgrading the WebView (migrating-client-applications/cordova/#upgrading-the-webview).
- Modify the call to client resources that use the classic security model. For example, use the `WLResourceRequest` API instead of `WL.Client.invokeProcedure`, which is deprecated.
- If you use Direct Update, review Migrating Direct Update (migrating-client-applications/cordova/#migrating-direct-update).
- For more information about migrating Cordova or hybrid apps, see Migrating existing Cordova and hybrid applications (migrating-client-applications/cordova).

> **Note:** The migration of push notification support requires client-side and server-side changes and is described later on in Migrating push notification support.

Migrating a native application

To migrate native application, you need to follow these steps:

- For planning purpose, run the migration assistance tool on your existing project. Review the generated report and assess the effort required for migration.
- Update your project to use the SDK from IBM MobileFirst Foundation v8.0
- 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.
- Modify the call to client resources that use the classic security model. For example, use the `WLResourceRequest` API, instead of `invokeProcedure`, which is deprecated.
    - For more information about migrating native iOS apps, see Migrating existing native iOS applications (migrating-client-applications/ios).
    - For more information about migrating native Android apps, see Migrating existing native Android applications (migrating-client-applications/android).

    - For more information about migrating native Windows apps, see Migrating existing native Windows applications (migrating-client-applications/windows).

> **Note:** The migration of push notification support requires client-side and server-side changes and is described later on in Migrating push notification support.

Migrating adapters and security

Starting with v8.0, adapters are Maven projects. The MobileFirst security framework is based on OAuth, security scopes, and security checks. Security scopes define the security requirements to access a resource. Security checks define how a security requirement is verified. Security checks are written as Java adapters. For a hands-on experience with adapters and security, see the tutorials for Creating Java and JavaScript Adapters (../adapters/creating-adapters) and Authorization concepts (../authentication-and-security).

MobileFirst Server operates only in session-independent mode and adapters should not store a state locally to a Java virtual machine (JVM).

You can externalize adapter properties to configure adapters for the context where they run, for example a test server or a production server. But the values of these properties are no longer included in a property file of a project WAR file. Instead, you define them from the MobileFirst Operations Console, or by using a command-line tool or the server REST API.

- For more information about migrating adapters, see Migrating existing adapters (migrating-adapters) to work under MobileFirst Server v8.0.
- For more information about server-side API changes, see Server-side API (../product-overview/release-notes/deprecated-discontinued/#server-side-api-changes) changes in v8.0.
- For an introduction to Apache Maven used to develop adapters, see Adapters as Apache Maven projects (../adapters).
- For more information on migrating authentication and security see Migrating Authentication and Security (migrating-security) to MobileFirst v8.0.

Migrating push notifications support

The event-source-based model is no longer supported. Instead, use tag-based notification. To learn more about migrating push notification for your client apps and your server-side components, see Migrating push notifications (migrating-push-notifications) from event source-based notifications and Migration scenarios (migrating-push-notifications/#migration-scenarios).

Starting with v8.0, you configure the push service on the server side. The push certificates are stored on the server. You can set them from the MobileFirst Operations Console or you can automate certificate uploads by using a command-line tool or the push service REST API. You can also send push notifications from the MobileFirst Operations Console.

The push service is protected by the OAuth security model. You must configure server-side components that use the push service REST API must be configured as confidential clients of MobileFirst Server.

Changes in the server databases and in the server structure

MobileFirst Server enables changes to app security, connectivity and push without code change, app rebuild or redeployment. But these changes imply changes in the database schemas, the data stored in the database, and the installation process.

Because of these changes, IBM MobileFirst Platform Foundation does not include automated scripts to

migrate your databases from earlier versions to V8.0.0 or to upgrade an existing server installation. To move new versions of your apps to V8.0.0, install a new server that you can run side by side with your previous server. Then, upgrade your apps and adapters to V8.0.0 and deploy them to the new server.

Storing mobile data in Cloudant

Storing mobile data in Cloudant with the IMFData framework or CloudantToolkit is no longer supported. For an alternative API, see Migrating apps storing mobile data in Cloudant with IMFData or Cloudant SDK (migrating-data).

Applying a fix pack to IBM MobileFirst Server

Find out how to use the Server Configuration Tool to upgrade MobileFirst Server V8.0.0 to a fix pack or an interim fix. Alternatively, if you installed MobileFirst Server with Ant tasks, you can also use Ant tasks to apply the fix pack or interim fix.

To apply an interim fix or fix pack on MobileFirst Server, choose one of the following topics based on your initial installation method:

- Applying a fix pack or an interim fix with the Server Configuration Tool (../installation-configuration/production/appserver/#applying-a-fix-pack-by-using-the-server-configuration-tool)
- Applying a fix pack by using the Ant files (../installation-configuration/production/appserver/#applying-a-fix-pack-by-using-the-ant-files)

Last modified on