Migrating existing Windows applications

Overview

To migrate an existing native Windows 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 Windows apps to prepare for MobileFirst version 8.0

The migration assistance tool helps you prepare your apps that were created with earlier versions of IBM MobileFirst™ Platform Foundation for migration by scanning the sources of the native Windows 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 Windows 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 earlier 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 native Windows 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 windows
   ```

   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 a Windows project

To work with existing native Windows project that was created with IBM MobileFirst™ Platform Foundation V6.2.0 or later, you must modify the project.

MobileFirst v8.0 only supports Windows Universal environments, that is Windows 10 Universal Windows Platform (UWP) and Windows 8 Universal (Desktop and Phone). Windows Phone 8 Silverlight is not supported.

You can upgrade your Visual Studio project to v8.0 manually. MobileFirstv8.0 introduces a number of changes to the Visual Studio SDK that may require changes to apps developed in earlier versions. For information on the API's that have changed, see Updating the Windows code.

1. Update your MobileFirst SDK to v8.0.
    - Remove the MobileFirst SDK packages manually. This includes the **wlclient.properties** file, as well as the following references:
        - Newtonsoft.Json
        - SharpCompress
        - worklight-windows8

      → **Note:** If your app uses the application authenticity or extended authenticity feature, you must add either Microsoft Visual C++ 2013 Runtime Package for Windows or Microsoft Visual C++ 2013 Runtime Package for Windows Phone as a reference to your app. To so do, in Visual Studio, right-click on the references of your native project and complete one of the following choices depending on which environment you added to your native API app: → → * For Windows desktops and tablets: Right click **References → Add reference → Windows 8.1 → Extensions → Microsoft Visual C++ 2013 Runtime Package for Windows → OK**. → * For Windows Phone 8 Universal: Right click **References → Add reference → Windows 8.1 → Extensions → Microsoft Visual C++ 2013 Runtime Package for Windows Phone → OK**. → * For Windows 10 Universal Windows Platform (UWP): Right click **References → Add reference → Windows 8.1 → Extensions → Microsoft Visual C++ 2013 Runtime Package for Windows Universal → OK**.
    - Add the MobileFirst V8.0.0 SDK packages through NuGet. See Adding the MobileFirst SDK by using NuGet (../../../application-development/sdk/windows-8-10).
2. Updating your application code to use MobileFirst V8.0.0 API's.
    - For earlier releases, the Windows API's were part of the **IBM.Worklight.namespace**. These API's are now obsolete and have been replaced by equivalent **WorklightNamespace** API in the. You need to modify the app to replace all references to the **IBM.Worklight.namespace** with the corresponding equivalent in the **WorklightNamespace**.

    For example, the following snippet is an example of using the

    ```
    WLResourceRequest request = new WLResourceRequest
                    (new Uri(uriBuilder.ToString()), "GET", "accessRestricted");
                    request.send(listener);
    ```

    The snippet updated with the new API would be:

    ```
    WorklightResourceRequest request = newClient.ResourceRequest
                    (new Uri(uriBuilder.ToString(), UriKind.Relative), "GET", "accessRestricted");
                    WorklightResponse response = await request.Send();
    ```

    - All methods that performed asynchronous operations previously used a Response listener call back model. These have been replaced by the **await**/**async** model.

You can now start developing your native Windows application with the MobileFirst SDK. You might need to update your code to reflect the changes for MobileFirstV8.0.0 API.

What to do next

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

Updating the Windows code

IBM MobileFirst Foundation v8.0 introduces a number of changes to the Windows SDK that might require changes to apps developed in earlier versions.

Deprecated Windows C# API Classes

Category	Description	Recommended action
`ChallengeHandler`	For custom gateway challenges, use `GatewayChallengeHandler`. For MobileFirst security-check challenges, use `SecurityCheckChallengeHandler`.	
`ChallengeHandler`, `isCustomResponse()`	Use `GatewayChallengeHandler.canHandleResponse().`	
`ChallengeHandler.submitAdapterAuthentication`	Implement similar logic in your challenge handler. For custom gateway challenge handlers, use `GatewayChallengeHandler`. For MobileFirst security-check challenge handlers, use `SecurityCheckChallengeHandler`.	
`ChallengeHandler.submitFailure(WLResponse wlResponse)`	For custom gateway challenge handlers, use `GatewayChallengeHandler.Shouldcancel()`. For MobileFirst security-check challenge handlers, use `SecurityCheckChallengeHandler.ShouldCancel()`.	
`WLAuthorizationManager`	Use `WorklightClient.WorklightAuthorizationManager` instead.	
`WLChallengeHandler`	Use `SecurityCheckChallengeHandler`.	
`WLChallengeHandler.submitFailure(WLResponse wlResponse)`	Use `SecurityCheckChallengeHandler.ShouldCancel()`.	
`WLClient`	Use `WorklightClient` instead.	
`WLErrorCode`	Not supported.	
`WLFailResponse`	Use `WorklightResponse` instead.	
`WLResponse`	Use `WorklightResponse` instead.	
`WLProcedureInvocationData`	Use `WorklightProcedureInvocationData` instead.	
`WLProcedureInvocationFailResponse`	Not supported.	
`WLProcedureInvocationResult`	Not supported.	
`WLRequestOptions`	Not supported.	
`WLResourceRequest`	Use `WorklightResourceRequest` instead.	

Deprecated Windows C# API Interfaces

Category	Description	Recommended action
`WLHttpResponseListener`	Not supported.	
`WLResponseListener`	The response will be available as a `WorklightResponse` object	
`WLAuthorizationPersistencePolicy`	Not supported.	