

Application Authenticity Protection in Hybrid applications

fork and edit tutorial (<https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/7.1/authentication-security/application-authenticity-protection/application-authenticity-protection-hybrid-applications.html>) | report issue (<https://github.ibm.com/MFPSamples/DevCenter/issues/new>)

This tutorial is a continuation of the Application Authenticity Protection (../) tutorial.

The application-descriptor.xml file

Add the `securityTest` attribute to the relevant environment element. For example:

```
<iphone bundleId="com.worklight.MyBankApp" applicationId="MyBankApp" securityTest="customTests" version="1.0">
```

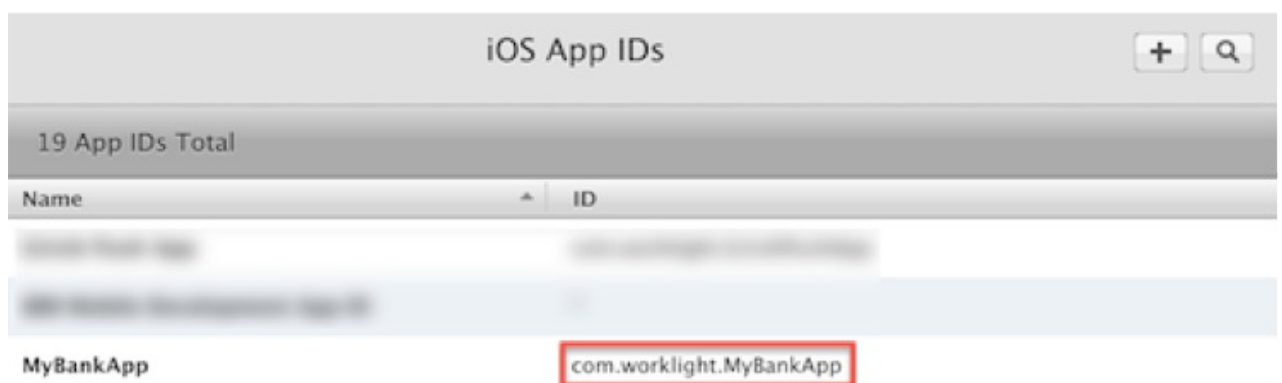
Next, you must make modifications that are specific to each environment.

iOS

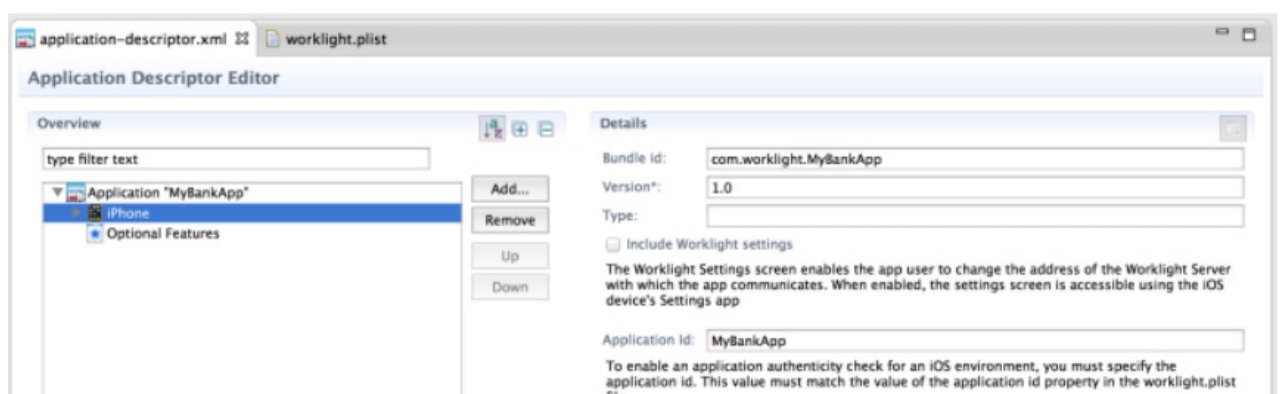
To enable application authenticity protection check for the iPhone/iPad environment, specify the following identifiers in the `application-descriptor.xml` file.

Specifying the `bundleId` and `applicationId` identifiers

1. Specify the `bundleId` value of the application exactly as it is defined in the **Apple Developer portal**.



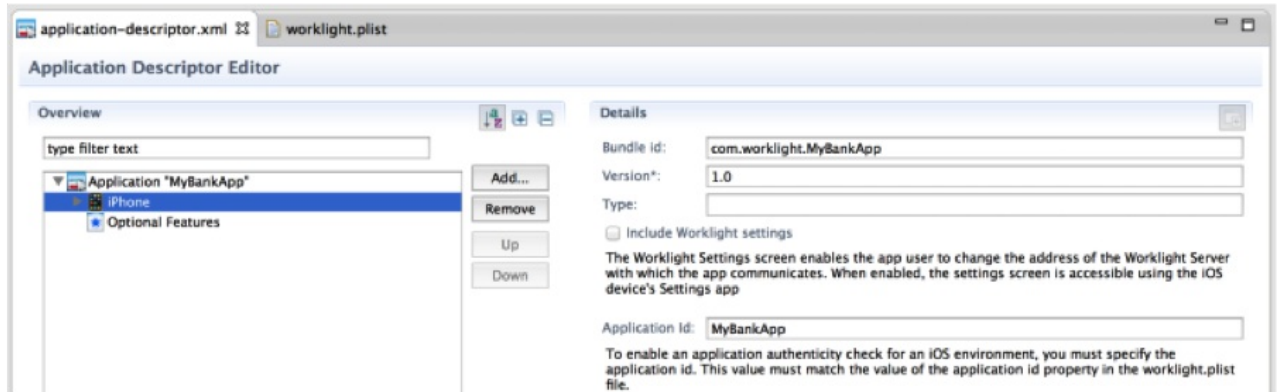
You can add the value either in the Application Description Design view:



Or in the Application Descriptor Source view:

```
<iphone bundleId="com.worklight.MyBankApp" version="1.0">
```

2. Specify the `applicationId` value. The application identifier must match the value of the `application id` property, which is located in the `native\worklight.plist` file. You can add the value either in the Application Description Design view:



Or in the Application Descriptor Source view:

```
<iphone bundleId="com.worklight.MyBankApp" applicationId="MyBankApp" securityTest="customTests" version="1.0">
```

3. In Xcode, verify that the following value exists in the **Other Linker Flags** field: `-ObjC`

Android

To enable application authenticity protection check for the Android hybrid environment, follow these steps:

1. Extract the public signing key of the certificate that is used to sign application bundle (`.apk` file).
 - If the application is built for distribution (production), extract the public key from the certificate that is used to sign the production-ready application.
 - If the application is built in the development environment, you can use the default public key that is supplied by the Android SDK. You can find the development certificate in a keystore that is in a `{user-home}/.android/debug.keystore` file.
 - You can extract the public signing key either manually or by using the wizard that MobileFirst Studio provides.

Extracting the public signing key by using the wizard (Eclipse)

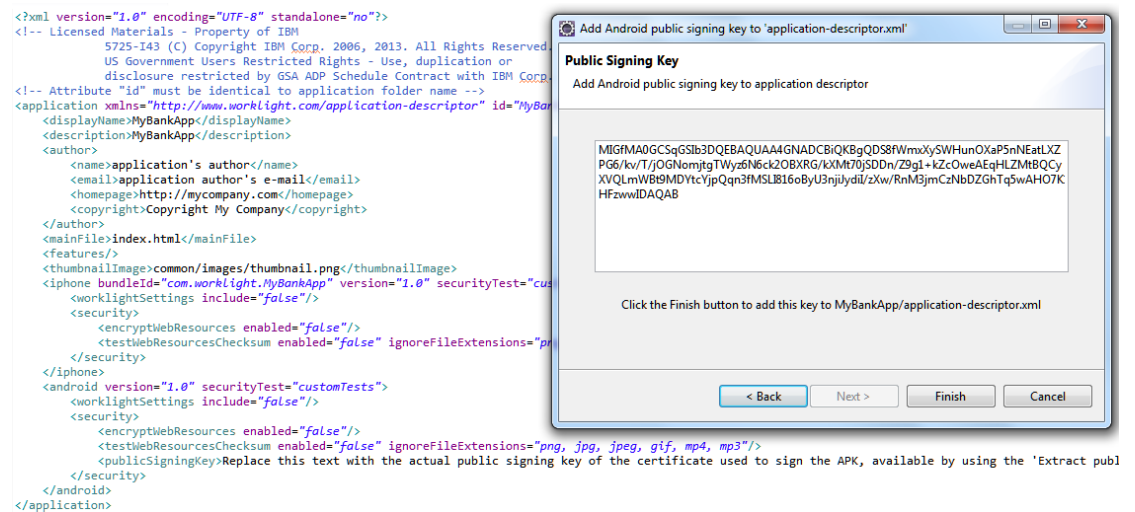
If you are using Android Studio - look here (<http://developer.android.com/tools/publishing/app-signing.html>) for information about how to sign your Android application.

The following instructions are for Eclipse:

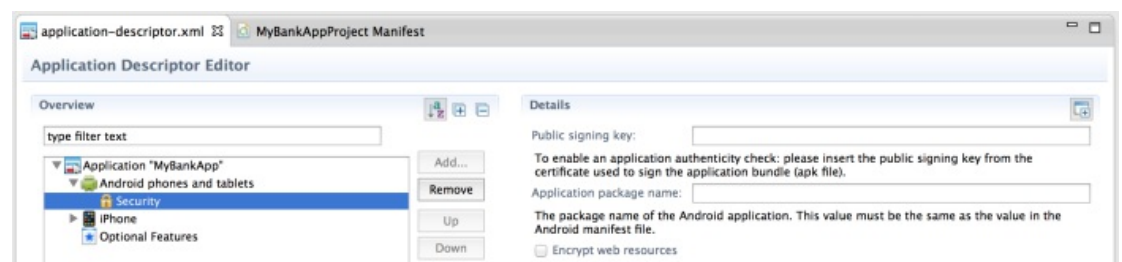
1. Right-click the Android environment folder and select **Extract public signing key**.
2. Specify the location and the password of a keystore file and click **Load Keystore**.
The default password for **debug.keystore** is `android`.
3. Set the **Key alias** and click **Next**.

A dialog displays the public key.

4. Click **Finish**. The public key is automatically pasted to the relevant section of the `application-descriptor.xml` file.



5. Add the Application package name by using the Application Descriptor Editor (design view):



6. Take the value of the application package name from the package attribute of the *manifest* node in the `AndroidManifest.xml` file.

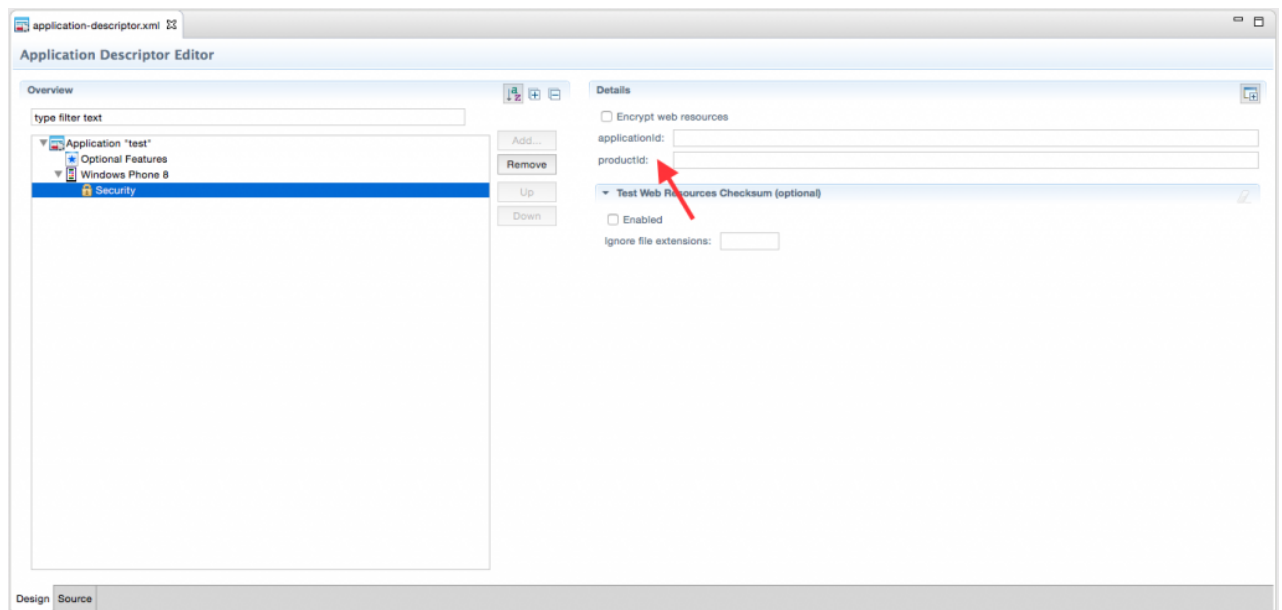
If you decide to change the value, make sure that you change it in both locations. You can also directly edit the `application-descriptor.xml` file and add a package name.

```
<android version="1.0">
  <worklightSettings include="false"/>
  <security>
    <encryptWebResources enabled="false"/>
    <testWebResourcesChecksum enabled="false" ignoreFileExtensions="
png, jpg, jpeg, gif, mp4, mp3"/>
    <publicSigningKey>MIGff ...</publicSigningKey>
    <packageName>com.MyBankApp</packageName>
  </security>
</android>
```

Windows Phone 8 - Silverlight

To enable application authenticity check for the Windows Phone 8 hybrid environment, modify the `application-descriptor.xml` file as follows.

- In the Application Descriptor Design view, supply the `applicationId` and `productId` in the Windows Phone 8 Security section:



You can find the `productId` value in the `native\PropertiesWMAAppManifest.xml` file. The `applicationId` value must match the value of the `wlAppId` property, which you can find in the `native\wlclient.properties` file.

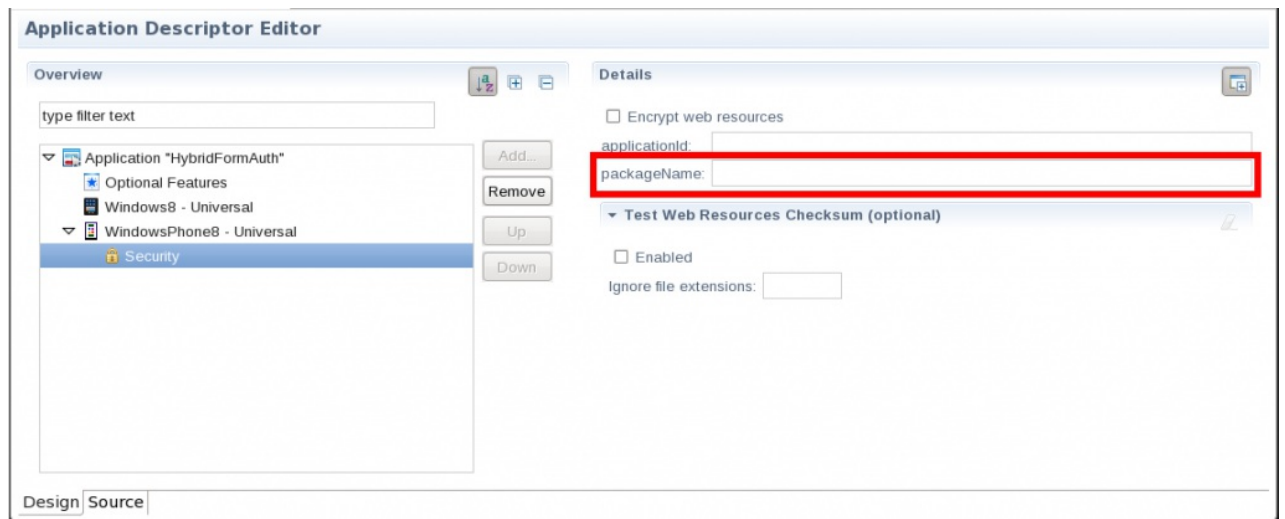
- You can also supply these values in the Application Descriptor Source view. For example:

```
<windowsPhone8 version="1.0">
  <uuid>b5542877-7afe-4edc-a817-5341b5027633</uuid>
  <security>
    <productId>fca81480-7b4a-4ed0-b387-178e8fa0c3d5</productId>
  >
    <applicationId>HelloWorld</applicationId>
  </security>
</windowsPhone8>
```

Windows 8 - Universal

To enable application authenticity check for the Windows 8 Hybrid Universal applications, modify the `application-descriptor.xml` file as follows.

- Add the `packageName` attribute to the Windows / Windows Phone environment element:



- You can also supply the values in the Application Descriptor Source view. For example:

```
<windowsphoneuniversal version="1.0">
  <uuid>3b081db3-9b9e-44c4-b7c0-1f09b067ce98</uuid>
  >
  <security>
    <packageName></packageName>
  </security>
</windowsphoneuniversal>
```

- You can find the `packageName` value in the `native project\environment(Win8 or WinPh8)\package.appxmanifest` file:

