# Application Authenticity Protection

## Overview

By issuing an HTTP request, any entity can access the HTTP services (APIs) that IBM MobileFirst Platform Foundation Server offers.
The Application Authenticity Protection feature ensures that an application that tries to connect to a MobileFirst Server instance is the authentic one and was not tampered with or modified by a third-party attacker.

Application authenticity protection is available for:

- Android: native and hybrid
- iOS: native and hybrid
- Windows Universal: native and hybrid
- Windows Phone Silverlight 8: hybrid only

**Note:**
Application Authenticity Protection is **not available** in the MobileFirst Development Server. To test, deploy the application to a MobileFirst Server instance on a remote application server.

### Agenda

- Authenticity flow and types
- Setting up authenticity

## Authenticity flow and types

The challenge token is processed by compiled native code, so that third-party attackers cannot see the logic of this processing.

Application Authenticity Protection is based on certificate keys that are used to sign the application bundles. Only the developer or the enterprise who have the original private key that was used to create the application are able to modify, repackage, and re-sign the bundle.

Three levels of authenticity are available:

1. No Authenticity
2. Basic Authenticity
3. Extended Authenticity (Provides a more detailed verification and is more secured)

# Setting up authenticity

To set up application authenticity protection, you go through the following two steps:

1. Project-level setup
2. Environment-specific setup

## Project-level setup

Whether you want Basic or Extended authenticity protection, the following setup steps are required. *Extended* authenticity protection requires additional setup steps, which are explained in the Enabling extended protection section.

### Modify the `authenticationConfig.xml` configuration file

1. Add the relevant authentication realm to a security test.
   - If a `mobileSecurityTest` is used, add the `testAppAuthenticity` child element to it:

     ```
     <mobileSecurityTest name="MyMobileAuthenticityTest">
       <testAppAuthenticity/>
       <testDeviceId provisioningType="none" />
       <testUser realm="myMobileLoginForm" />
     </mobileSecurityTest>
     ```

   - If a `customSecurityTest` is used, add the `wl_authenticityRealm` realm to it:

     ```
     <customSecurityTest name="MyCustomAuthenticityTest">
       <test realm="wl_authenticityRealm" step="1"/>
       <test realm="wl_anonymousUserRealm" isInternalUserID="true" step="1"/>
       <test realm="wl_deviceNoProvisioningRealm" step="2" isInternalDeviceID="true"/
     >
     </customSecurityTest>
     ```

2. After you have configured the authenticity realm and security check, go to Environment-specific setup and select your environment to complete the authenticity configuration.

## Enabling extended application authenticity protection

To enable extended authenticity checking, you must deploy a modified `.wlapp` file, instead of the original `.wlapp` file that is generated by the build process.

1. Modify the `.wlapp` file by using the `wladm` program or the `wladm` Ant task:
   - **By using the `wladm` program**
     Use the `wladm` program (provided in the MobileFirst installation directory) to run the `enable extended-authenticity` command:

     ```
     wladm enable extended-authenticity src-wlapp-file device-file > dest-wlapp-file
     ```

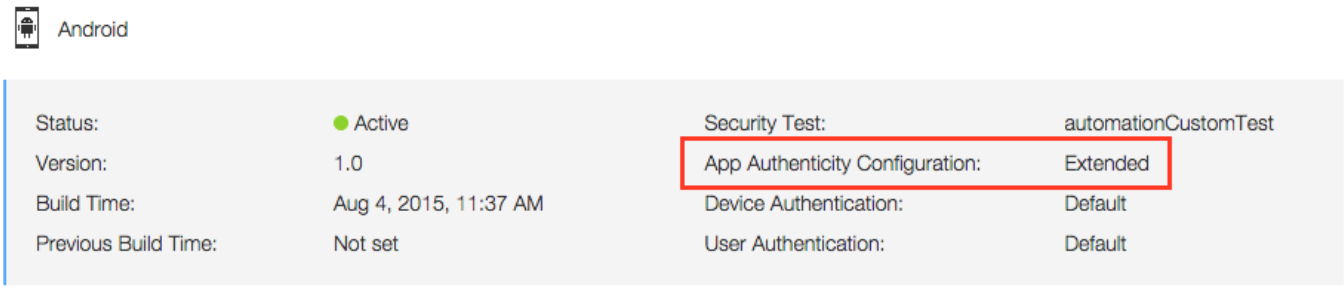
     - `src-wlapp-file` => Original binary app file (`.wlapp`)
     - `device-file` => Binary mobile app file (`.apk`, `.ipa`, or `.xap`)
     - `dest-wlapp-file` => Output binary app file (`.wlapp`)

     After running the `wladm`, deploy the resulting file to the MobileFirst Server instance.
   - **By using the `wladm` Ant task**
     Use the `wladm` Ant task (provided in the MobileFirst installation directory) to run the `enable-extended-authenticity` command:

     ```
     <enable-extended-authenticity srcwlappfile="original-.wlapp-file"
       devicefile="device file(.apk, .ipa, or .xap)"
       destwlappfile="output-.wlapp-file"/>
     ```

2. After running the `wladm` command or Ant task, deploy the resulting file to MobileFirst Server instance.

You can use the MobileFirst Operations Console to see the authenticity level of an application.




> For more information, see the topic about configuring extended app authenticity checking, in the user documentation.

## Environment-specific setup

For basic authenticity, go through the basic configuration steps.
For extended authenticity, it is recommended to make these extra configuration changes.