

Oracle® Mobile Application Framework

Installing Oracle Mobile Application Framework



2.5.1.0.0
E92588-01
June 2018

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Oracle Mobile Application Framework Installing Oracle Mobile Application Framework, 2.5.1.0.0

E92588-01

Copyright © 2015, 2018, Oracle and/or its affiliates. All rights reserved.

Primary Author: Walter Egan

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

	Preface	
	Audience	v
	Documentation Accessibility	v
	Related Documents	v
	Conventions	v
1	Installing Mobile Application Framework with JDeveloper	
	Prerequisites for Installing the MAF Extension	1-1
	Installing the MAF Extension in JDeveloper	1-2
2	Setting Up the Development Tools for the iOS Platform	
	Installing Xcode and iOS SDK	2-1
	Specifying the iOS Settings in MAF	2-1
	Using the GUI	2-1
	Using the Command Line	2-2
	Setting Up an iOS Device or Simulator	2-2
	Testing the iOS Environment Setup	2-3
3	Setting Up Development Tools for the Android Platform	
	Installing the Android SDK	3-1
	Specifying the Android Settings in MAF	3-1
	Using the GUI	3-2
	Using the Command Line	3-2
	Installing an Emulator Accelerator	3-3
	Creating an Android Virtual Device	3-4
	Setting Up Your Android Device to Install an App from Your Development Machine	3-4
	Testing the Android Environment Setup	3-5

4 Setting Up Development Tools for the Universal Windows Platform

Installing Visual Studio	4-1
Creating a PFX File for MAF Applications	4-3
Installing a PFX File on Windows 10	4-4
Specifying the UWP Settings in MAF	4-5
Enabling Developer Mode on Windows 10	4-5
Testing the Windows Environment Setup	4-6

5 Migrating Your Application to MAF 2.5.1

Migrating an Application to MAF 2.5.1	5-1
Using Xcode 9.x with MAF 2.5.1	5-8
How To Maintain Separate Xcode 9.x and Xcode 8.3.x Installations	5-9
Migrating Cordova Plugins from Earlier Releases to MAF 2.5.1	5-9
Evaluating EL Expressions in the Java VM Layer	5-11
Configuring Application Features with AMX Content to Use WKWebView on iOS	5-12
Security Changes in Release 2.4.0 and Later of MAF	5-12
Security Changes in Release 2.2.1 and Later of MAF	5-13
Migrating an Application Developed Using AMPA to MAF 2.5.1	5-14
Migrating MAF Applications that Use Custom URL Schemes to Invoke Other Applications	5-17
Migrating to JDK 8 in MAF 2.5.1	5-17
Retaining Legacy Behavior When Navigating a MAF Application Using Android's Back Button	5-18
How to Retain Pre-MAF 2.2.0 Application Behavior in Response to Usage of Android's Back Button	5-19
Migrating to New cacerts File for SSL in MAF 2.5.1	5-19

Preface

Welcome to *Installing Oracle Mobile Application Framework*.

Audience

This manual is intended for developers who want to install the Oracle Mobile Application Framework for use with Oracle JDeveloper to create mobile applications that run natively on devices.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Documents

For more information, see *Developing Mobile Applications with Oracle Mobile Application Framework*.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements (for example, menus and menu items, buttons, tabs, dialog controls), including options that you select.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates language and syntax elements, directory and file names, URLs, text that appears on the screen, or text that you enter.

1

Installing Mobile Application Framework with JDeveloper

This chapter describes how to install JDeveloper and the Mobile Application Framework (MAF) extension for application development. This chapter includes the following sections:

- [Prerequisites for Installing the MAF Extension](#)
- [Installing the MAF Extension in JDeveloper](#)

Prerequisites for Installing the MAF Extension

Before you can develop MAF applications, you must install Oracle JDeveloper and the MAF extension.

The MAF extension provides JDeveloper with the design-time support (wizards, editors, and dialogs) that you use to develop MAF applications in JDeveloper. Ensure you review and satisfy these prerequisites before proceeding with the installation of the MAF Extension.

- Procure a computer running the required operating system based on the platform to which you need to deploy MAF applications.
 - For the iOS platform, you need a computer with Apple Mac OS X.
 - For the Android platform, you need a computer with Apple Mac OS X or a supported Microsoft Windows or Linux version.
 - For the Universal Windows Platform (UWP), you need a computer with x86 architecture running Windows 10 (version 1511 or later).

For the supported versions of operating systems that you can use to develop MAF applications, see Certification Information on Oracle Technology Network at: <http://www.oracle.com/technetwork/developer-tools/maf/documentation/index.html>.

- Download and install the latest version of JDK 1.8.

For the certified JDK versions required for different operating systems, see Certification Information on Oracle Technology Network at: <http://www.oracle.com/technetwork/developer-tools/jdev/documentation/index.html>.

- Download and install Oracle JDeveloper.
 1. Download the installer from <http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html>.
 2. Install and set up JDeveloper as described in Starting the Oracle JDeveloper Studio Installation Program in *Installing Oracle JDeveloper*.

Installing the MAF Extension in JDeveloper

After you install Oracle JDeveloper, download and install the MAF extension.

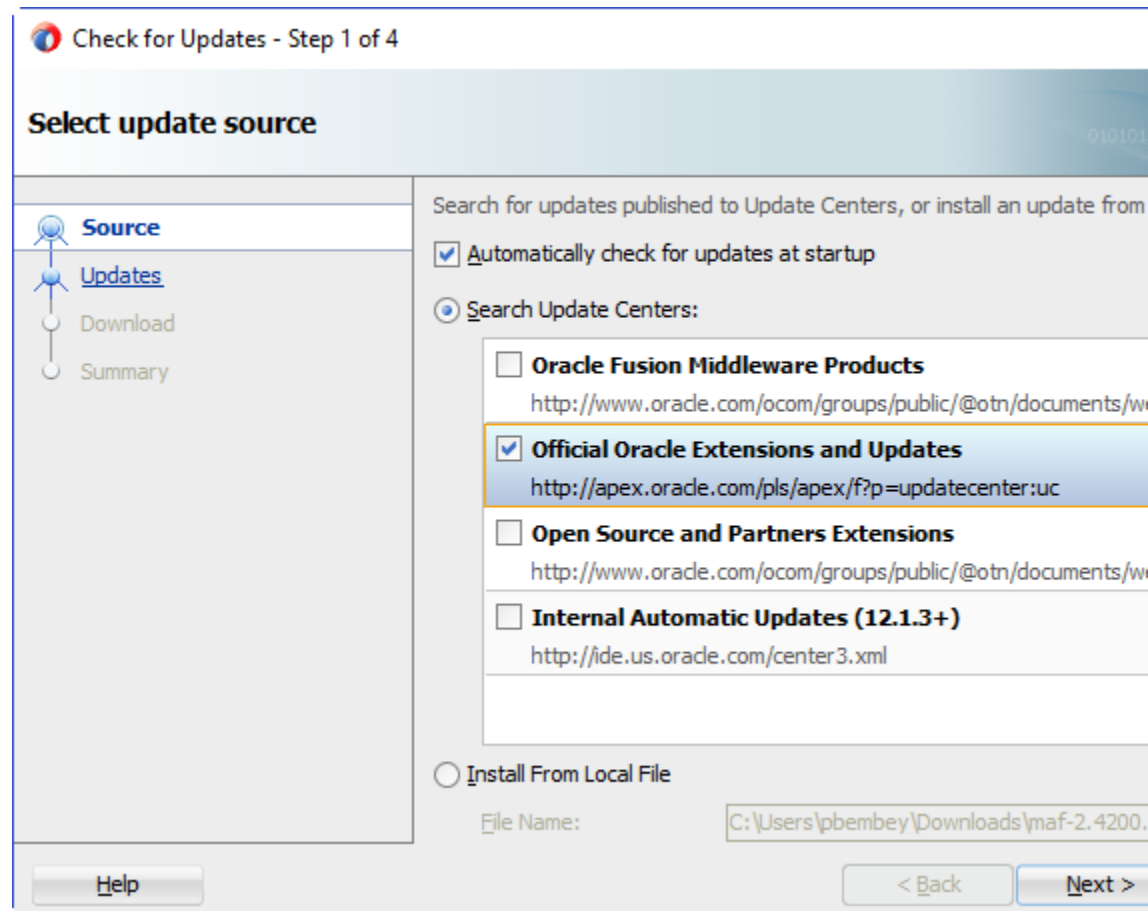
Before you can install the MAF extension, complete these prerequisites:

- Close any existing MAF applications open in JDeveloper. This ensures that MAF migrates your existing applications to successfully use the new MAF version. Verify that the application no longer appears in the Applications window of JDeveloper. For information about migrating a MAF application, see [Migrating Your Application to MAF 2.5.1](#).
- Configure proxy settings on your development computer if you are behind a corporate firewall.
 - On Windows, select **Tools** then **Preferences**, and then **Web Browser and Proxy** from the tree on the left of the Preferences dialog.
 - On Mac OS X, select **JDeveloper** then **Preferences**, and then **Web Browser and Proxy** from the tree on the left of the Preferences dialog.

To download and install the MAF extension:

1. In JDeveloper, select **Help** then **Check for Updates**.
2. On the Select update source page, select **Official Oracle Extensions and Updates** under **Search Update Centers**, and then click **Next**.

Figure 1-1 Checking for Updates in JDeveloper



 **Note:**

If you need to install an older version of MAF, download the file from the [Update Center](#) and select the **Install From Local File** option on the Select update source page. Browse and select the MAF extension file that you downloaded to your development computer from the Update Center.

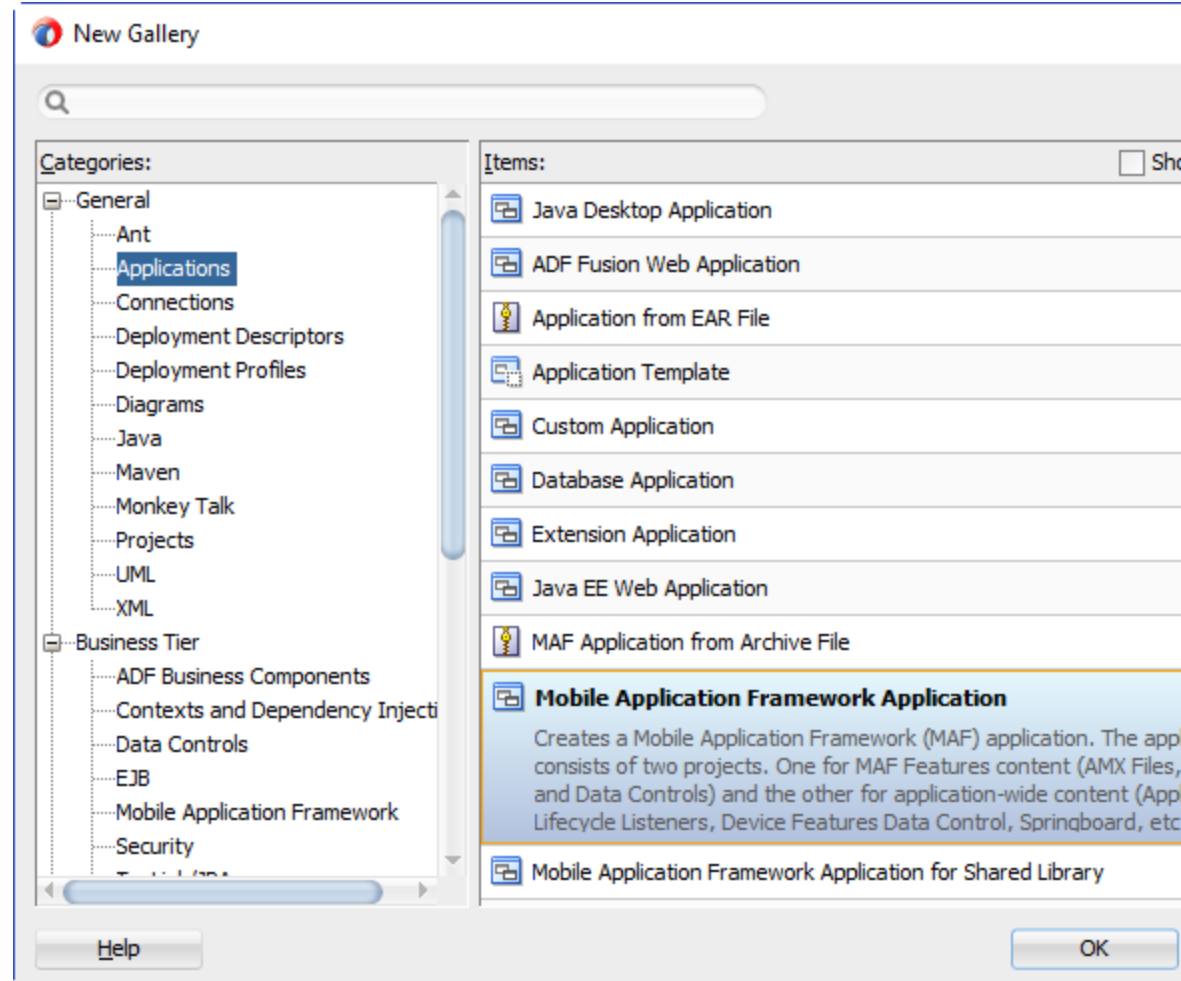
3. In the Select updates to install dialog, select the **Mobile Application Framework** update.
4. On the License Agreements page, review *The Oracle Technology Network License Terms for Oracle Mobile*, and then click **I Agree**.

You must comply with all the license terms and conditions with respect to the Oracle Mobile Application Framework Program. See <http://www.oracle.com/technetwork/indexes/downloads/index.html>.

5. Click **Next**, and then click **Finish**.
6. Restart JDeveloper.
7. Check whether MAF has been successfully added to JDeveloper:

- a. Select **File**, then **New** and then **From Gallery** from the main menu to open the New Gallery dialog.
- b. In the **Categories** tree on the left, select **General**, then **Applications** to check if it contains **Mobile Application Framework Application**.

Figure 1-2 Verifying MAF Installation



8. Verify whether you have installed the correct version of MAF.
 - a. Click **Help** then **About** to open the About Oracle JDeveloper dialog.
 - b. Click **Extensions**, and search for **Mobile Application Framework** in the extension list entries. Review the Version column and verify the version number of the MAF extension.

After you have installed the MAF extension, you need to install other software for the platforms (Android, iOS, or UWP). Subsequent chapters in this guide describe the specific steps for each platform.

2

Setting Up the Development Tools for the iOS Platform

This chapter provides information on setting up and configuring development tools for the iOS platform. Install Xcode and the iOS SDK so that you can deploy the MAF applications developed in JDeveloper to an iOS device or simulator. This chapter includes the following sections:

- [Installing Xcode and iOS SDK](#)
- [Specifying the iOS Settings in MAF](#)
- [Setting Up an iOS Device or Simulator](#)
- [Testing the iOS Environment Setup](#)

Installing Xcode and iOS SDK

Download and install Xcode. The download includes the iOS SDK.

Download Xcode from <http://developer.apple.com/xcode/>.

After installing Xcode, you have to run it at least once and complete the Apple licensing and setup dialogs. If these steps are not followed, the build and deploy cycle from JDeveloper to Xcode or a device simulator fails with a "Return code 69" error.

Note:

Older versions of Xcode and iOS SDK are not available from the Apple App Store. To download older versions:

1. Obtain an Apple ID from <http://appleid.apple.com>.
2. Register the Apple ID with the Apple Developer Program to access the Apple developer site at <http://developer.apple.com>.

Specifying the iOS Settings in MAF

Specify the iOS platform settings in JDeveloper so that you can deploy a MAF application to the iOS platform.

For the iOS platform, you can specify preferences using the GUI or command line.

Using the GUI

To configure your environment for the iOS platform:

1. Ensure Xcode and iOS SDK are installed.

2. In JDeveloper, click **Tools**, and then click **Preferences**.
3. In the Preferences dialog, click **Mobile Application Framework** and then click **iOS Platform**.
4. Enter the signing information and export options.
for more information, see Setting the Device Signing Options in *Developing Mobile Applications with Oracle Mobile Application Framework*.

Using the Command Line

You can set MAF preferences required to develop MAF applications by specifying startup parameters when you start JDeveloper.

To launch JDeveloper from the command line with startup parameters, use the `-J-D` options. All strings must be enclosed in double-quotes, as shown in the examples.

The following example shows how to override the provisioning profile name.

```
./jdev -J-Doracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.iosProvisioningProfileName="Oracle ENT1 2017"
```

These are the startup parameters you can use to set iOS preferences from the command line:

- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.iosProvisioningProfileName`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.iosProvisioningProfileTeamIdentifier`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.iosCertificate`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.iosExportMethod`

Setting Up an iOS Device or Simulator

To deploy MAF applications, connect an iPhone or iPad to your development computer or configure external network access to use iOS simulators, included in XCode downloads.

In your MAF application development and deployment, you can use either the iOS-powered device itself or its simulator.

Deployment method	Description
Device (iOS-powered)	<p>Deploying to an iPhone or iPad is preferable for testing.</p> <p>You must connect the device to your computer to establish a link between the two devices. Also, you need to have an iOS-powered device with a valid license, certificates, and distribution profiles. See <i>Deploying Mobile Applications</i> in <i>Developing Mobile Applications with Oracle Mobile Application Framework</i>.</p>
Simulator	<p>Deploying to a simulator is usually much faster than deploying to a device, and it also means that you do not have to sign the application first. Before attempting to deploy your application from JDeveloper to a device simulator, you must first run the simulator. A simulator can be invoked automatically, without any additional setup.</p>

 **Note:**

Since the Apple licensing terms and conditions may change, ensure that you understand them, comply with them, and stay up to date with any changes.

Testing the iOS Environment Setup

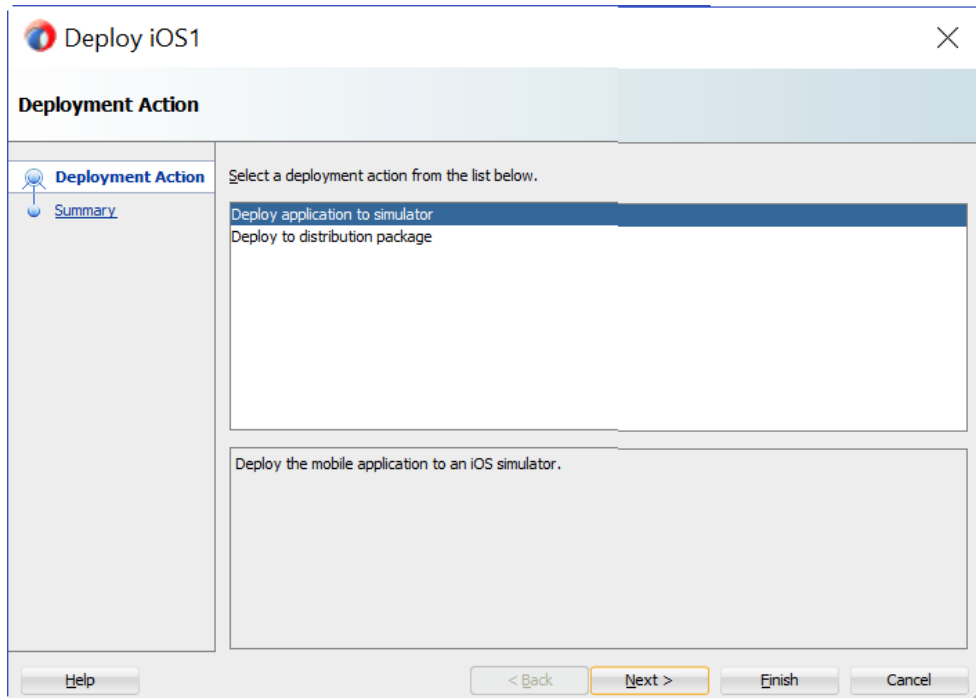
Deploy a MAF sample application to test that you set up your iOS environment successfully.

You can test your environment setup as follows:

1. In JDeveloper, open the HelloWorld sample application.
See MAF Sample Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*.
2. Select **Application**, and then click **Deploy** from the main menu.
3. From the dropdown menu, select the deployment profile for the iOS platform.
4. Select **Deploy application to simulator** in the **Deploy** dialog.

Using an iOS-powered device simulator test the environment setup is preferable because it does not require signing of the application.

Figure 2-1 Selecting Deployment Action for iOS



5. Click **Next** on the Deploy dialog to verify the Summary page, and then click **Finish**.

See Deploying Mobile Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*.

After a successful deployment (which might take a few minutes), the device to which you had deployed the application displays the launch screen of the HelloWorld application, and then displays the default application feature.

3

Setting Up Development Tools for the Android Platform

This chapter provides information on setting up and configuring development tools for the Android platform. Install the Android SDK and the emulator accelerator so that you can deploy the MAF applications developed in JDeveloper to a configured Android device or emulator.

This chapter includes the following sections:

- [Installing the Android SDK](#)
- [Specifying the Android Settings in MAF](#)
- [Installing an Emulator Accelerator](#)
- [Creating an Android Virtual Device](#)
- [Setting Up Your Android Device to Install an App from Your Development Machine](#)
- [Testing the Android Environment Setup](#)

Installing the Android SDK

Install the Android SDK to deploy a MAF application to Android devices.

Android Studio, Google's IDE for Android development, includes the Android SDK in its installation and provides wizard options that simplify the management of the SDK platforms and tools that you need.

Install Android Studio, and the Android SDK that it includes, by downloading the installation file from <https://developer.android.com/studio/index.html>. The Android Developer's website provides installation instructions for Windows, Mac, and Linux. See <https://developer.android.com/studio/install.html>.

The Android SDK provides:

- Tools that build and package your application into an .APK file (the file type that installs applications on Android devices)
- An emulator to create Android Virtual Devices (AVD) where you can test your application if you do not have access to a physical Android device
- An OEM USB driver to connect your development machine to a physical Android device through a USB cable if you do have a device (enables you to deploy an application from your development machine to the Android device)

Specifying the Android Settings in MAF

Configure the Android-specific settings, such as Android SDK, build tools location, and signing information to package and deploy applications to the Android platform.

For the Android platform, you can specify preferences using the GUI or command line.

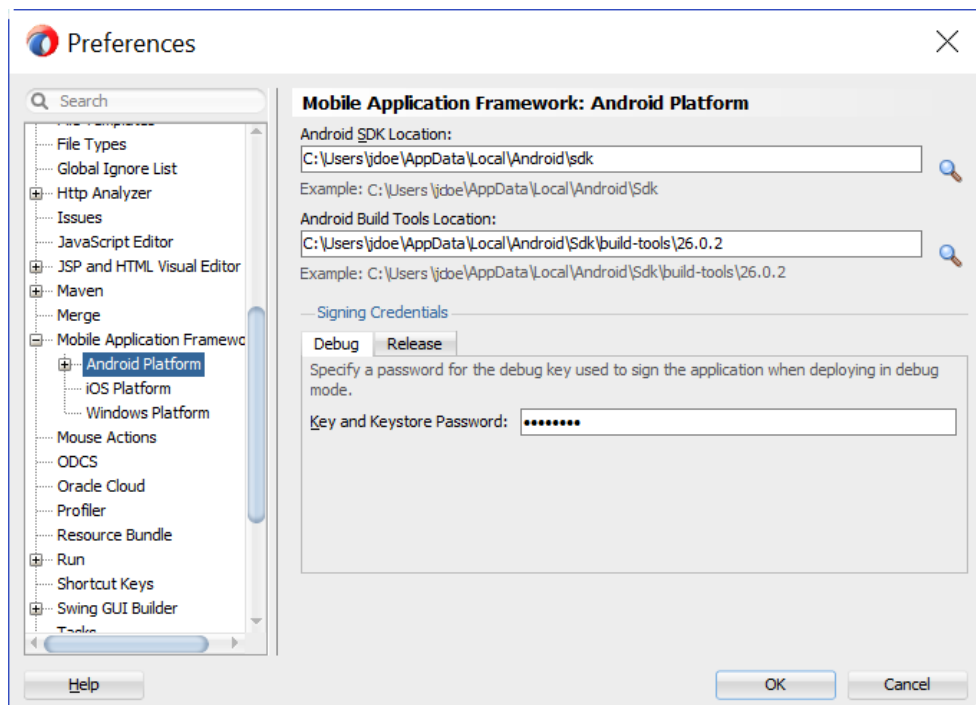
Using the GUI

To configure your environment for the Android platform:

1. Ensure the Android SDK is installed.
2. In JDeveloper, click **Tools**, and then click **Preferences**.
3. In the Preferences dialog, click **Mobile Application Framework**, and then click **Android Platform**.
4. Specify the location of the Android SDK on your computer.
5. Specify the location of the Android build tools on your computer.
6. Provide information on the signing credentials.

For more information about creating a key and keystore, see [How to Sign a MAF Application that You Deploy to the Android Platform in *Developing Mobile Applications with Oracle Mobile Application Framework*](#).

Figure 3-1 Configuring Platform Preferences for Android



Using the Command Line

You can set MAF preferences required to develop MAF applications, such as the Android SDK location, by specifying startup parameters when you start JDeveloper.

To launch JDeveloper from the command line with startup parameters, use the `-J-D` options. All strings must be enclosed in double-quotes, as shown in the examples.

The following example shows how to override the location of the Android SDK:

```
jdeveloper.exe -J-
Doracle.admf.framework.dt.preferences.PlatformSDKsPrefs.androidPlatformDir="C:
```

```
\<my_Android_SDK_path> "
```

These are the startup parameters you can use to set Android preferences from the command line:

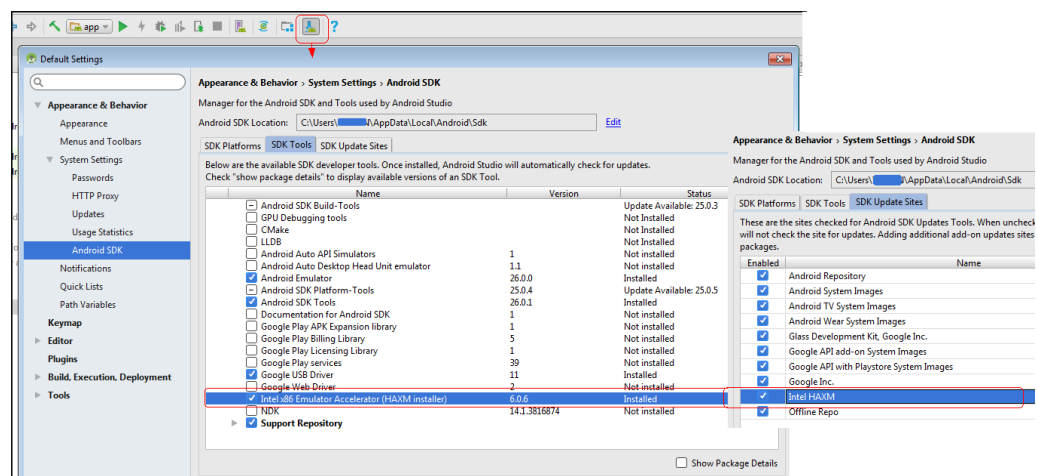
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.androidSdkDir`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.androidBuildToolsDir`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.androidReleaseSigningKeyStorePath`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.androidReleaseSigningKeyStorePath`

Installing an Emulator Accelerator

You can accelerate the performance of the emulator that renders AVDs by installing an emulator accelerator.

Once installed, the emulator accelerator speeds up the performance of the emulator and the AVDs that it emulates by allocating additional resources from your development machine. You specify the amount during installation of the accelerator. Once installed, the accelerator appears in the SDK Tools list of the SDK Manager that you can launch from Android Studio, as shown in the following image. The Intel x86 Emulator Accelerator (HAXM) is one type of emulator accelerator that is available.

Figure 3-2 Install Emulator Accelerator



Make sure that the update site for the emulator accelerator that you want to download is selected in the **SDK Update Sites** tab shown in the previous image. Once downloaded, execute the installer. See <https://developer.android.com/studio/run/emulator-acceleration.html#accel-vm>.

Creating an Android Virtual Device

An Android Virtual Device (AVD) replicates an Android device on your development computer. It is a useful option for testing, especially if you only have access to one or a limited range of physical Android devices.

The AVD Manager that you launch from Android Studio by clicking **Tools** then **Android**, and then **AVD Manager** has a range of ready-to-use virtual devices. It includes most of those devices developed by Google itself, such as the Nexus and Pixel XL range. Google maintains documentation describing how to manage AVDs (see <https://developer.android.com/studio/run/managing-avds.html>).

Other Android device vendors, such as Samsung, provide specifications on their websites that you can use to create the AVD yourself.

To create an AVD:

1. In Android Studio, launch the Android Virtual Device Manager by selecting **Tools**, then **Android**, and then **AVD Manager**.
2. In the Your Virtual Devices screen, click **Create Virtual Device**.
3. In the Select Hardware screen, select a phone device, such as Pixel, and then click **Next**.
4. In the System Image screen, click **Download** for one of the recommended system images. Agree to the terms to complete the download.
5. After the download completes, select the system image from the list and click **Next**.
6. On the next screen, leave all the configuration settings unchanged and click **Finish**.
7. In the Your Virtual Devices screen, select the device you just created and click **Launch this AVD in the emulator**.

Setting Up Your Android Device to Install an App from Your Development Machine

You can install your app directly from your development machine to your Android device by configuring the Android device and connecting it to your development machine using a USB cable.

To set up your Android device:

1. Connect your device to your development machine with a USB cable.
If you are developing on Windows, you might need to install the appropriate USB driver for your device. For help installing drivers, see the [OEM USB Drivers](#) document.
2. Enable USB debugging on your device by going to **Settings > Developer options**.

 **Note:**

Developer options is hidden by default. To make it available, go to **Settings > About phone** and tap **Build** number seven times. Return to the previous screen to find **Developer options**.

Testing the Android Environment Setup

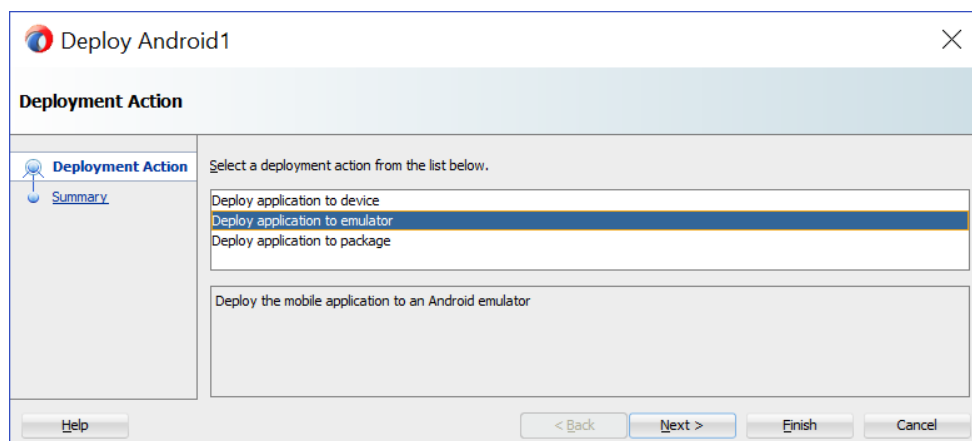
Deploy a MAF sample application to test that you set up your Android environment successfully.

You can test your environment setup as follows:

1. In JDeveloper, open the HelloWorld sample application.
See MAF Sample Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*.
2. Select **Application** then **Deploy** from the main menu.
See Deploying Mobile Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*.
3. From the dropdown menu, select the deployment profile for the Android platform.
4. Select **Deploy application to emulator** in the **Deploy** dialog.

Using an Android-powered device emulator to test the environment setup is preferable because it does not require signing of the application. Ensure that the emulator is running before you start the deployment.

Figure 3-3 Selecting Deployment Action for Android



5. Click **Next** on the Deploy dialog to verify the Summary page, and then click **Finish**.

See Deploying Mobile Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*.

After a successful deployment (which might take a few minutes), the device to which you had deployed the application displays the launch screen of the HelloWorld application, and then displays the default application feature.

4

Setting Up Development Tools for the Universal Windows Platform

This chapter provides information on setting up and configuring development tools for the Universal Windows platform. Install Visual Studio along with needed SDKs and create and install the PFX file so that you can deploy the MAF applications developed in JDeveloper to the Universal Windows Platform.

This chapter includes the following sections:

- [Installing Visual Studio](#)
- [Creating a PFX File for MAF Applications](#)
- [Installing a PFX File on Windows 10](#)
- [Specifying the UWP Settings in MAF](#)
- [Enabling Developer Mode on Windows 10](#)
- [Testing the Windows Environment Setup](#)

Installing Visual Studio

Install Visual Studio 2017 along with the required Windows 10 SDKs.

The Visual Studio installer also allows you to install the Windows SDKs that enable deployment of applications to the UWP. Here are the other components you must install.

- MSBuild 15.0 (automatically installed with Visual Studio 2017)
- Universal Windows Platform development tools (a workload available when installing Visual Studio 2017)
- Windows 10 SDK 10.0.10586.0 (an optional component available when installing Visual Studio 2017)
- Windows 10 SDK 10.0.14393.0 (an optional component available when installing Visual Studio 2017)

To install Visual Studio 2017:

1. Ensure MAF and JDeveloper (12.2.1.3.0) are installed on a computer with x86 architecture running the Windows 10 operating system (version 1511 or later).
2. Download an edition of Visual Studio 2017 available at: <https://www.visualstudio.com/downloads/>.

Select the Visual Studio edition that you require: Community, Professional, or Enterprise. All editions provide the required software to develop and deploy a MAF application to the UWP. Visit the Visual Studio Community product page for information about the eligibility criteria to use Visual Studio Community edition.

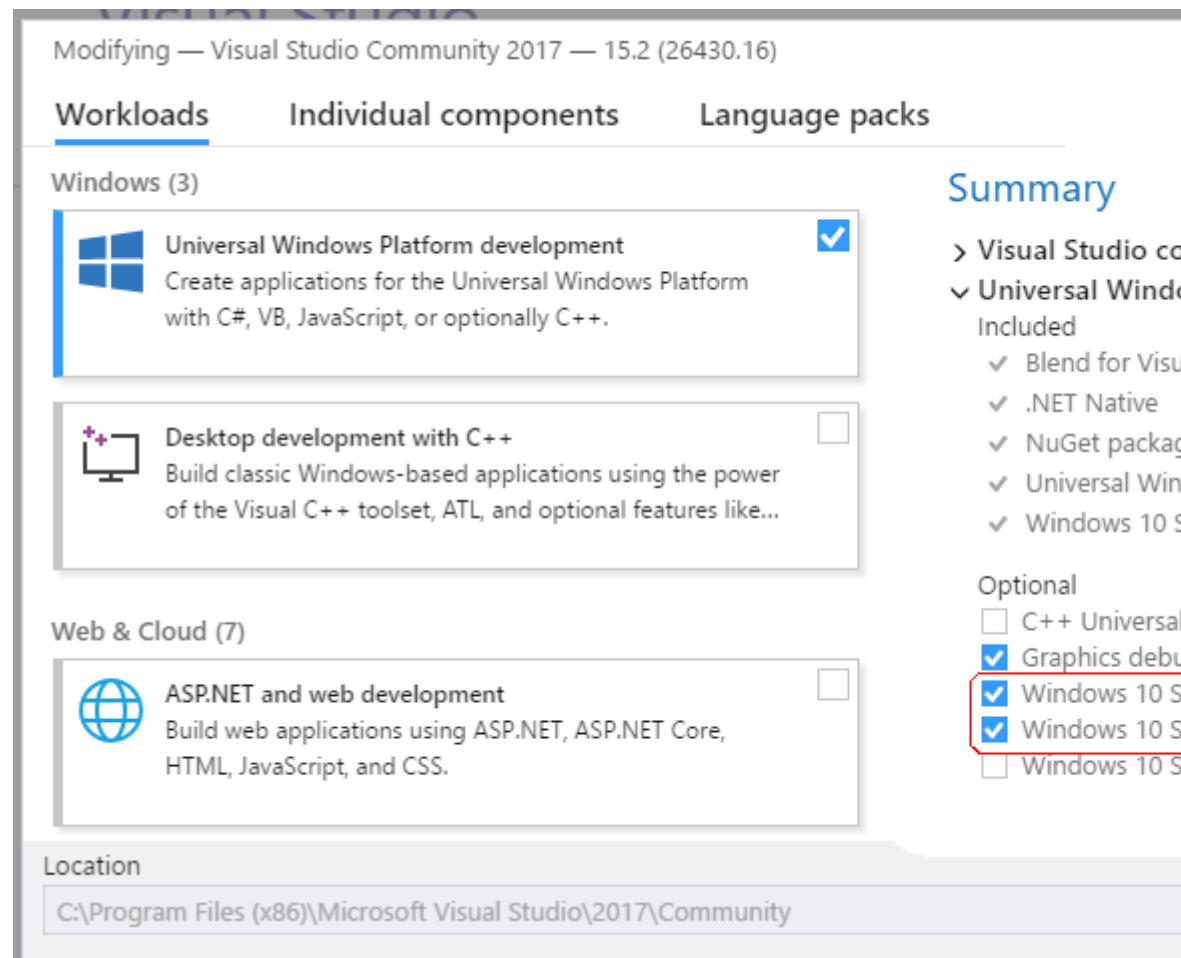
3. During the Visual Studio 2017 installation, select these options.
 - a. Select the **Universal Windows Platform development** workload.

- b. Select **Windows 10 SDK (10.0.10586.0)** and **Windows 10 SDK (10.0.14393.0)** available under the Individual components.

 **Note:**

If you are migrating from an existing MAF and Visual Studio 2015 installation, you might already have the required Windows 10 SDKs installed. If you are performing a fresh installation of Visual Studio 2017, you must install these Windows 10 SDKs.

Figure 4-1 Installing Visual Studio 2017



For more information on the Windows 10 SDKs, see: <https://developer.microsoft.com/en-us/windows/downloads/sdk-archive>.

Creating a PFX File for MAF Applications

MAF applications deployed to the UWP must be digitally signed using a Personal Information Exchange (PFX) file.

Use Microsoft's `New-SelfSignedCertificate` cmdlet in a PowerShell console with administrator privileges to create a PFX file.

To create a PFX file:

1. Click **Start**, type **PowerShell**, right-click **Windows PowerShell**, and then click **Run as administrator**.

The PS `C:\windows\system32>` prompt is displayed.

2. Change the value for the `-Subject` parameter, as required and run the following command.

```
New-SelfSignedCertificate -CertStoreLocation cert:\localmachine\my -Subject
'CN=Example,OU=MAF,O=Oracle,C=US' -KeyAlgorithm RSA -KeyLength 2048
-Provider 'Microsoft Enhanced RSA and AES Cryptographic Provider' -
KeyExportPolicy Exportable -KeyUsage DigitalSignature
-Type Custom -TextExtension
@('2.5.29.37={text}1.3.6.1.5.5.7.3.3,1.3.6.1.4.1.311.10.3.13', '2.5.29.19={text}CA
=False')
```

In the command:

- `-CertStoreLocation`: Specifies the certificate store in which to save the new certificate.
- `-Subject`: Specifies the string that appears in the subject of the new certificate.
- `-KeyAlgorithm`: Specifies the algorithm used to create the asymmetric certificate key.
- `-KeyLength`: Specifies the length (in bits) of the certificate key.
- `-Provider`: Specifies the name of the KSP or CSP that the cmdlet uses to create the certificate.
- `-KeyExportPolicy`: Specifies the policy that governs the export of the private key.
- `-KeyUsage`: Specifies the key usages set in the key usage extension of the certificate.
- `-Type`: Specifies the type of certificate that the cmdlet creates.
- `-TextExtension`: Specifies an array of certificate extensions (strings) the cmdlet includes in the new certificate.

For more information on the parameters, see [Microsoft's documentation](#).

If successful, the command prints the thumbprint ID and subject of the certificate. Here is a sample output.

```
PSParentPath: Microsoft.PowerShell.Security\Certificate::LocalMachine\my
```

```
Thumbprint                               Subject
-----
EA8DA38619D6FF49C9BBE51651DDD6950EF767AE  CN=Example, OU=MAF, O=Oracle, C=US
```

3. Enter the following command.

```
$pwd = ConvertTo-SecureString -String '<password>' -Force -AsPlainText
```

In the command, replace <password> with a suitable password.

4. Enter the following command to create the PFX file.

```
Export-PfxCertificate -cert cert:\localMachine\my\<thumbprint ID> -FilePath  
<path> -Password $pwd
```

In the command:

- Change <thumbprint ID> to the value output to the console (for example, EA8DA38619D6FF49C9BBE51651DDD6950EF767AE)
 - Change <path> to your preferred path, such as c:\someDir\MyPFX.pfx
5. Navigate to the specified path and verify whether the PFX file was created.

Installing a PFX File on Windows 10

Install a Personal Information Exchange (PFX) file in a certificate store on a computer so that the certificate can be used for application signing.

A Software Publisher Certificate (SPC), with its private and public keys, is used for application signing and is stored in a Personal Information Exchange (.pfx) file. A PFX file has to be copied or installed to a certificate store, where the operating system keeps all certificates.



Note:

The installation has to be completed once, manually, for every PFX file on a given computer.

To install a PFX file in a certificate store:

1. Locate and double-click the .pfx file to open the file in the Certificate Import Wizard.
2. Select **Current User** as the Store Location, and then click **Next**.

When you install the PFX file in the Local Machine store, the Windows User Access Control dialog is opened. Click **Yes** for **Do you want to allow this app to make changes to your PC?**

3. Verify whether the name in the **File name** field is the one you want, and then click **Next**.



Note:

The default file location is the location of the file that you double-clicked.

4. Enter a password for the private key, if required.
5. Select **Included all extended properties**, and then click **Next**.
6. Select **Place all certificates in the following store**, and click **Browse**.

7. In Select Certificate Store, select the certificate store that matches the store location, **Personal**, click **OK**.
8. Click **Next**, and then on Completing the Certificate Import Wizard, click **Finish** to import the certificate.

This procedure installs the PFX file in the **Personal** certificate store.

9. Run the Certificate Import Wizard a second time, select the **Current User** location, and the **Trusted People** certificate store.
10. Run the Certificate Import Wizard a third time, select the **Local Machine** location, and the **Trusted People** certificate store.

Specifying the UWP Settings in MAF

Configure the UWP-specific settings, such as Windows SDK location and PFX file information to package and deploy applications to the UWP platform.

To package and deploy applications to the platforms supported by MAF, JDeveloper needs the name of the platform and the names of the directories containing platform-specific tools and data. For convenience, MAF populates JDeveloper Preferences with these settings. Each platform-specific page hosts the preferences for the platform SDK (Android, iOS, or Windows), collecting information, such as the path that MAF needs to compile and deploy Android, iOS, or Windows projects.

To configure your environment for the UWP platform:

1. Ensure the needed Windows SDKs are installed when installing Microsoft Visual Studio 2017.
2. In JDeveloper, click **Tools** then **Preferences**.
3. In the Preferences dialog, click **Mobile Application Framework** then **Windows Platform**.
4. Specify the location of the Windows SDK files (MSBuild version 15.0).
5. Provide the certificate (PFX file) location and password.

You can use the same PFX file to run your application in the Release and Debug modes on your computer. We recommend that you use a certificate issued by a trusted authority, such as your internal CA, if you want to distribute your application and run it on other devices within your organization.

Enabling Developer Mode on Windows 10

Enable Developer Mode on the Windows 10 development computer to side-load applications and to run them in the Debug mode.

Windows 10 runs UWP applications from a trusted source. Since the certificates you imported are self-signed, they will not run by default. If you want to develop and deploy MAF applications to the UWP you must enable Developer Mode on the Windows 10 computer that you use. Developer Mode is required for the following reasons:

- Side-load, or install and run applications, from unofficial sources.
- Run an application in the Debug mode.

To enable Developer Mode:

1. Press the Windows key, search for Settings, and select Settings - Modern application from the displayed results.
2. Select **Update & Security**, then **For developers**, and click **Developer mode**.

 **Note:**

If you create an application in Visual Studio, the system prompts you with a dialog to enable Developer Mode.

3. Configure settings for the Windows platform in JDeveloper as described in [Specifying the UWP Settings in MAF](#).

Testing the Windows Environment Setup

Deploy a MAF sample application to test that you set up your Windows environment successfully.

You can test your environment setup as follows:

1. In JDeveloper, open the HelloWorld sample application.
See MAF Sample Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*.
2. Select **Application** then **Deploy** from the main menu.
See Deploying Mobile Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*.
3. From the dropdown menu, select the deployment profile for the Windows platform.
4. Select **Deploy application to local machine** in the **Deploy** dialog.
5. Click **Next** on the Deploy dialog to verify the Summary page, and then click **Finish**.

See Deploying Mobile Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*.

After a successful deployment (which might take a few minutes), the device to which you had deployed the application displays the launch screen of the HelloWorld application, and then displays the default application feature.

5

Migrating Your Application to MAF 2.5.1

This chapter provides information that you may need to know if you migrate an application created using an earlier release of MAF to MAF 2.5.1. This chapter includes the following sections:

- [Migrating an Application to MAF 2.5.1](#)
- [Using Xcode 9.x with MAF 2.5.1](#)
- [Migrating Cordova Plugins from Earlier Releases to MAF 2.5.1](#)
- [Evaluating EL Expressions in the Java VM Layer](#)
- [Configuring Application Features with AMX Content to Use WKWebView on iOS](#)
- [Security Changes in Release 2.4.0 and Later of MAF](#)
- [Security Changes in Release 2.2.1 and Later of MAF](#)
- [Migrating an Application Developed Using AMPA to MAF 2.5.1](#)
- [Migrating MAF Applications that Use Customer URL Schemes to Invoke Other Applications](#)
- [Migrating to JDK 8 in MAF 2.5.1](#)
- [Retaining Legacy Behavior When Navigating a MAF Application Using Android's Back Button](#)
- [Migrating to New cacerts File for SSL in MAF 2.5.1](#)

Migrating an Application to MAF 2.5.1

Customers who migrate to this release of MAF need to be aware of changes introduced in this release and earlier releases of MAF (for example, MAF 2.3.0) that may affect the applications you migrate.

You must install Android API Level 27 to build and deploy MAF applications to the Android platform. Ensure that you install the Android SDK Platform API Level 27 into the Android SDK Location that you have specified in your MAF preferences, as described in [Setting Up Development Tools for the Android Platform](#).

This release updates the Cordova engine versions that MAF uses (Android: 7.0.0 and iOS: 4.5.4). See [Migrating Cordova Plugins from Earlier Releases to MAF 2.5.1](#).

For information about new features introduced in this release, see What's New in This Guide for MAF Release 2.5.1 in *Developing Mobile Applications with Oracle Mobile Application Framework*.

Previous releases of MAF introduced the following changes that affect the applications you migrate.

Release	Description
MAF 2.5.0	<ul style="list-style-type: none"><li data-bbox="881 275 1385 558">• Required you to install the MAF extension in Oracle JDeveloper 12c (12.2.1.3.0). See Installing Mobile Application Framework with JDeveloper . All MAF applications that you open for the first time in JDeveloper using this release of MAF will prompt you to migrate the application because this release of MAF uses a newer version of JDeveloper (12.2.1.3.0).<li data-bbox="881 562 1385 846">• Required Xcode 9.x to build and deploy MAF applications to the iOS platform. Upgrade your installation to Xcode 9.x, as described in Using Xcode 9.x with MAF 2.5.1. The iOS deployment profile now contains Version and Build input fields where you can specify version and build numbers for the MAF applications you deploy to iOS. See Deploying an iOS Application.<li data-bbox="881 850 1385 1892">• For MAF applications that you deploy to Android, note the following:<ul style="list-style-type: none"><li data-bbox="930 919 1385 1486">– You must install Android API Level 26 to build and deploy MAF applications to the Android platform. Ensure that you install the Android SDK Platform API Level 26 into the Android SDK Location that you have specified in your MAF preferences, as described in Setting Up Development Tools for the Android Platform. As a result of this change, MAF applications can no longer share <code>file://</code> URLs to external application and restrictions apply on the directory location of files that you display to an external viewer using the <code>DeviceFeatures</code> data control's <code>displayFile</code> method. See How to Use the displayFile Method to Enable Displaying Files in Developing Mobile Applications with Oracle Mobile Application Framework.<li data-bbox="930 1491 1385 1892">– MAF now includes a new core plugin "Storage Access" (<code>maf-cordova-plugin-storage-access</code>) that controls the storage permissions of MAF applications you deploy to the Android platform. MAF enables the Storage Access plugin by default which means that MAF applications you deploy to Android devices can access local storage if granted permission by users. MAF enables the Storage Access plugin in MAF applications that you migrate to this release of MAF. You can disable the


Release	Description
	<p>Storage Access plugin in your migrated MAF application. See <i>Enabling a Core Plugin in Your MAF Application</i> in <i>Developing Mobile Applications with Oracle Mobile Application Framework</i>.</p> <ul style="list-style-type: none"> <li data-bbox="1008 449 1458 758">– MAF applications on the Android platform that you migrate to this release of MAF and later enable multidex support by default. You can disable multidex support by modifying the Android deployment profile you use. See <i>Deploying a MAF Application to the Android Platform</i> in <i>Developing Mobile Applications with Oracle Mobile Application Framework</i>. <li data-bbox="959 768 1458 1136">• MAF now uses standard JDeveloper constructs to deploy MAF applications to the Universal Windows Platform, which means that you can deploy your application from the command line, as described in <i>Deploying MAF Applications from the Command Line Using OJDeploy</i>. The number of log levels that you can specify when you deploy an application has also increased. You configure the log level you want to use (Quiet, Minimal, and so on) in the Windows deployment profile. See <i>Working with Deployment Profiles</i>. <li data-bbox="959 1146 1458 1283">• Updated the Cordova engine versions that MAF uses (Android: 6.2.3, iOS: 4.5.0, and Windows: 5.0.0). See Migrating Cordova Plugins from Earlier Releases to MAF 2.5.1. <li data-bbox="959 1293 1458 1774">• MAF applications now evaluate EL expressions in the Java VM layer where previously this evaluation took place in the Web View layer. MAF applications that you migrate to this release use the new behavior (evaluate EL expressions in the Java VM layer). If your MAF application includes custom components, you must now create a <code>/META-INF/amx-tag-libraries.xml</code> file that describes the custom components your MAF application uses. See <i>Creating Custom UI Components</i>. You can configure new or migrated MAF applications to evaluate EL expressions in the Web View layer. See Evaluating EL Expressions in the Java VM Layer.

Release	Description
MAF 2.4.2	Used Visual Studio 2017, in addition to Visual Studio 2015, to set up the development environment that you use to deploy MAF applications to the Universal Windows Platform. Ensure that you have both versions 10.0.10586.0 and 10.0.14393 of the Windows 10 SDK installed in whichever version of Visual Studio you use. See Setting Up Development Tools for the Universal Windows Platform .
MAF 2.4.1	Required Xcode 8.3.x to build and deploy MAF applications to the iOS platform and you must select an export method (Ad Hoc, App Store, Development, or Enterprise) from the Method dropdown list in the iOS platform page of the Preferences dialog, as described in Setting the Device Signing and Export Options of <i>Developing Mobile Applications with Oracle Mobile Application Framework</i> . Note also that you can maintain two separate installations of Xcode, as described in Using Xcode 9.x with MAF 2.5.1 , if you want to maintain two separate development environments for different versions of MAF.

Release	Description
MAF 2.4.0	<ul style="list-style-type: none"> Used Gradle to build and deploy MAF applications to the Android platform. MAF downloads and installs Gradle during the initial deployment of a MAF application to Android. You may need to configure Gradle proxy settings to ensure a successful installation of Gradle. See How to Configure Gradle Proxy Settings in Developing Mobile Applications with Oracle Mobile Application Framework. Updated the Cordova engine versions that MAF used (Android: 6.0.0, iOS: 4.3.0, and Windows: 4.4.3). As a result, you may need to update custom Cordova plugins that you use in your migrated application, as described in Migrating Cordova Plugins from Earlier Releases to MAF 2.5.1. Removed APIs that were deprecated in previous releases. Before you upgrade to this release, review deprecation warnings reported at build time and modify your application to use supported APIs. If you do not do this, your migrated application may fail to build following upgrade to this release. For information about the APIs that MAF supports, see the <i>Java API Reference for Oracle Mobile Application Framework</i>. Defaulted the HTTPS protocol to TLSv1.2 on MAF applications that you deploy to the Android platform. Although not recommended, you can override this default behavior, as described in Security Changes in Release 2.4.0 and Later of MAF.
MAF 2.3.3 and later	<p>Required Xcode 8 to develop and deploy MAF applications to the iOS platform. It also supports deployment of applications to devices running on the iOS 10 platform. See Using Xcode 9.x with MAF 2.5.1.</p>
MAF 2.3.2 and later	<ul style="list-style-type: none"> Replaced the <code>java.security</code> file in your migrated MAF application with a new version generated by MAF. MAF saves the original file with the following filename: <code>java.security.orig</code>. If you had previously made changes to this file you may need to copy those changes to the new version of the <code>java.security</code> file. Included capability to configure migrated applications that run on iOS 9 and later to use WKWebView, as described in Configuring Application Features with AMX Content to Use WKWebView on iOS.

Release	Description
MAF 2.3.1 and later	Included the client data model feature that provides offline read and write support for REST services. If you previously used the A-Team Mobile Persistence Accelerator (AMPA) extension to develop an application with these capabilities, you can migrate it to this release of MAF, as described in Migrating an Application Developed Using AMPA to MAF 2.5.1 .

Release	Description
MAF 2.3.0 and later	<ul style="list-style-type: none"> • Used newer versions of Cordova (4.x). If your migrated MAF application uses a third-party Cordova plugin, verify that it is compatible with the Android and iOS versions of Cordova that this release of MAF uses. See Migrating Cordova Plugins from Earlier Releases to MAF 2.5.1. • Stored the <code>RestServiceAdapter</code> interface to a new package location (<code>oracle.maf.api.dc.ws.rest</code>). The functionality that this interface specifies remains unchanged. For information about creating a REST web service adapter, see <i>Creating a Rest Service Adapter to Access Web Services in Developing Mobile Applications with Oracle Mobile Application Framework</i>. • Removed support for the following features that were deprecated in earlier releases: <ul style="list-style-type: none"> – Mobile-Social authentication server type. Customers are recommended to use another authentication type, such as OAuth, that MAF supports. – SOAP web services. Customers are recommended to use REST web services with JSON objects. See the <i>Using Web Services in a MAF Application in Developing Mobile Applications with Oracle Mobile Application Framework</i>. • No longer bundled the jQuery JavaScript library (starting with MAF 2.3.0). It is no longer used in AMX pages or components. Customers who want to use the jQuery JavaScript library need to explicitly include jQuery using feature includes. • Introduced support for the deployment of MAF applications to the Universal Windows Platform (UWP) (starting with MAF 2.3.0). If your migrated MAF application contains platform-specific code that only executes when the MAF application runs on a specific platform, revise your MAF application to include platform-specific code for the UWP if you want your MAF application to run on this newly-supported platform. For information about deploying a MAF application to the UWP, see <i>Deploying a MAF Application to the Universal Windows Platform in Developing Mobile Applications with Oracle Mobile Application Framework</i>.

Release	Description
MAF 2.2.1	Enabled App Transport Security (ATS) by default for applications that you migrate to this release. See Security Changes in Release 2.2.1 and Later of MAF . If your migrated application uses URL schemes to invoke other applications, configure the migrated application as described in Migrating MAF Applications that Use Customer URL Schemes to Invoke Other Applications .
MAF 2.1.0	<ul style="list-style-type: none"> Used newer versions of Apache Cordova and Java. It also changed the way that JDeveloper registered Cordova plugins in your MAF application. For SSL, it delivered a <code>cacerts</code> file that contained new CA root certificates. <div data-bbox="933 751 1380 1150" style="border: 1px solid #0070C0; padding: 10px; margin: 10px 0;"> <p> Note:</p> <p>If you migrate an application to MAF 2.3.0 or later that was created in MAF 2.1.0 or previously migrated to MAF 2.1.0, MAF will have made already made the changes required by migration to JDK 8, management of Cordova plugins, and a new <code>cacerts</code> file.</p> </div> <ul style="list-style-type: none"> Included an updated SQLite database and JDBC driver. Review, and migrate as necessary, any code in your migrated MAF application that connects to the SQLite database. For information about how to connect to the SQLite database, see <i>Using the Local SQLite Database in Developing Mobile Applications with Oracle Mobile Application Framework</i>.

Using Xcode 9.x with MAF 2.5.1

MAF 2.5.1 requires Xcode 9.x to develop and deploy MAF applications to the iOS platform.

Install or upgrade to Xcode 9.x, as described in [Installing Xcode and iOS SDK](#). Once you install or upgrade to Xcode 9.x, make sure to start it so that you accept the license agreements. Failure to do this may cause deployment errors when JDeveloper attempts to deploy your MAF application to iOS. With this installation, Xcode 9.x replaces the previous installation of Xcode. No other changes are required, since JDeveloper will now use the active Xcode installation. If you want to maintain separate development environments, you can install two instances of Xcode, as described below. For example, install Xcode 9.x for MAF 2.5.1 and install Xcode 8.3.x for MAF 2.4.2.

How To Maintain Separate Xcode 9.x and Xcode 8.3.x Installations

To maintain separate Xcode 9.x and Xcode 8.3.x installations:

1. Rename the preexisting Xcode.app installation for Xcode 8.3.x (For example, Xcode8.app.) and reboot your system before you proceed to the next step.
2. Install Xcode 9.x from the Apple App Store, as described in [Installing Xcode and iOS SDK](#). Make sure that you install, not update, Xcode from the Apple App Store.
3. Once you install Xcode 9.x, make sure to start it so that you accept the license agreements.

After installation, verify that you have the following Xcode installations in your Applications location:

```
Xcode 9.x installation:  
/Applications/Xcode.app
```

```
Xcode 8.3.x installation:  
/Applications/Xcode8.app
```

4. Once the two versions of Xcode have been installed, you must manually control which Xcode installation is active at any given time. Use the `xcode-select` command in a terminal window to perform this procedure, as shown in the following examples:

```
//To make Xcode 9.x active:  
sudo xcode-select -s /Applications/Xcode.app
```

```
//To make Xcode 8.3.x active:  
sudo xcode-select -s /Applications/Xcode8.app
```

```
//To determine which instance of Xcode is currently active:  
xcode-select --print-path
```

Migrating Cordova Plugins from Earlier Releases to MAF 2.5.1

MAF 2.5.1 and later use new versions of Cordova. See the Cordova Engine Version section in the overview editor for the `maf-application.xml` file for the version that each targeted platform uses.

To complete the migration and make sure that your migrated MAF application can continue to use any non-core plugins it used previously, verify that this release of MAF supports the version of the plugin(s) that your MAF application uses. The Cordova Engine Versions displays the versions that your release of MAF uses, as illustrated in [Figure 5-1](#). Obtain a newer version of the plugin if the plugin was created using an earlier release of Cordova than that used by the current release of MAF. Set the relative path to the plugin so that the `maf-plugins.xml` file of the MAF application correctly references the plugin. See [Registering Additional Plugins in Your MAF Application](#) in *Developing Mobile Applications with Oracle Mobile Application Framework*. If the `maf-plugins.xml` file does not correctly reference a plugin using a relative path, the overview editor for the `maf-application.xml` file's **Path*** field which requires a value is empty and the `maf-plugins.xml` displays a validation failure, as shown in [Figure 5-1](#).

MAF applications developed using earlier releases of MAF (prior to MAF 2.1.0) registered plugins in the `maf-application.xml` file. Release MAF 2.1.0 and later registers plugins in the `maf-plugins.xml` file. JDeveloper makes the following changes to an application from an earlier release that uses plugins when you migrate the application:

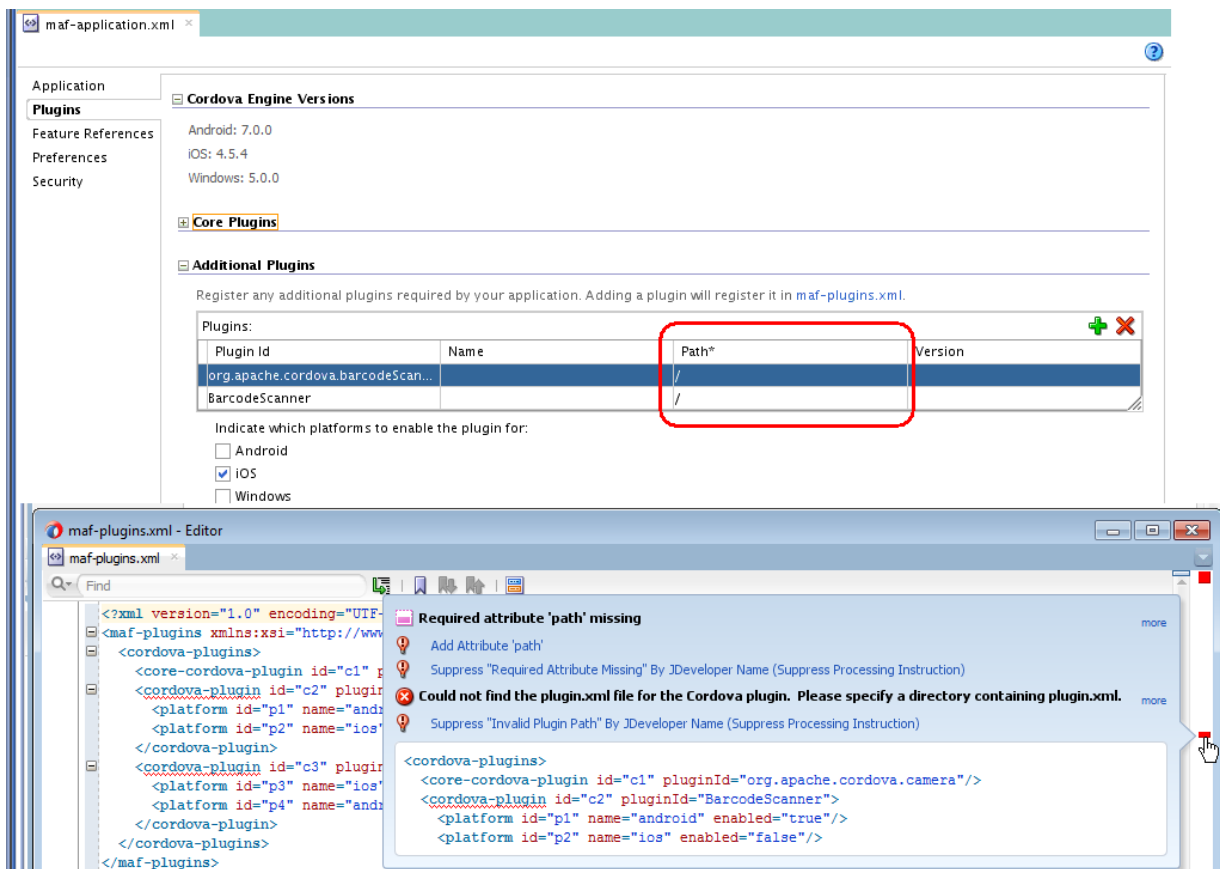
- Comments out entries in the `maf-application.xml` file that referenced plugins. For example, JDeveloper comments out entries such as the following:

```
<!--<adfmf:cordovaPlugins>
  <adfmf:plugin fullyQualifiedClassName="BarcodeScanner"
    implementationClass="com.phonegap.plugins.
      barcodescanner.BarcodeScanner" platform="Android"
      name="BarcodeScanner">
    . . . . .
  </adfmf:cordovaPlugins-->
```

- Registers the plugin in the `maf-plugins.xml` file, as shown in the following example:

```
<cordova-plugins>
  . . .
  <cordova-plugin id="c3" pluginId="org.apache.cordova.barcodeScanner">
    <platform id="p3" name="ios" enabled="true"/>
    <platform id="p4" name="android" enabled="false"/>
  </cordova-plugin>
</cordova-plugins>
```

Figure 5-1 MAF Application that Does Not Specify Path to Plugin



Evaluating EL Expressions in the Java VM Layer

Prior to this release, MAF handled AMX file, EL and AMX node creation in the Web View layer of the MAF application. With this release, MAF performs these tasks in the Java VM layer of new MAF applications that you create.

MAF applications that you migrate to this release also perform these tasks in the Java VM Layer. Note the following as you review MAF applications that you migrate to this release:

- A number of MAF JS APIs have been deprecated. For information about the up-to-date JS APIs that MAF supports, see *JSDoc Reference for Oracle Mobile Application Framework*.
- MAF applications that include custom components must now include a `/META-INF/amx-tag-libraries.xml` file with the metadata for the XML namespace you use to identify the custom components you support and their attributes. See *Creating Custom UI Components in Developing Mobile Applications with Oracle Mobile Application Framework*.
- As EL creation and evaluation has moved to the Java VM layer, EL expressions should not be evaluated in JavaScript and must not be used from component type handlers.

You can configure new and migrated MAF applications to use the legacy behavior where MAF evaluated EL expressions in the Web View Layer, as described in the following section.

How to Move EL Evaluation to the Web View Layer

You move EL evaluation to the Web View layer by setting the value of the `<amxTagHandling>` property in the `maf-config.xml` file to `legacy`.

To move EL evaluation to the Web View layer:

1. In the Applications window, expand the **Application Resources** panel.
2. In the Application Resources panel, expand **Descriptors** and then **ADF META-INF**.
3. Double-click the `maf-config.xml` file and in the source editor that appears, add the following entry:

```
<adfmf-config xmlns="http://xmlns.oracle.com/adf/mf/config">
  ...
  <!-- Use the Java VM layer to handle AMX tags and node hierarchies -->
  <amxTagHandling>legacy</amxTagHandling>
</adfmf-config>
```

Other valid values that you can add to the `maf-config.xml` file to handle EL evaluation include `webview` or `legacy`. These force the use of the Web View layer to handle AMX tags and node hierarchies. A value of `default` or no entry for the `amxTagHandling` property means the Web View layer handles AMX tags and node hierarchies.

Remove the `<amxTagHandling>legacy</amxTagHandling>` entry or set it to `<amxTagHandling>default</amxTagHandling>` if you want the MAF application to use the default behavior of evaluating EL expressions in the Java VM layer.

Configuring Application Features with AMX Content to Use WKWebView on iOS

New MAF applications that you create using the MAF 2.3.2 release and later of MAF use WKWebView by default to render AMX content type when you deploy the MAF application to an iOS device. You can opt to use this web view in MAF applications that you migrate to this release of MAF.

The newer WKWebView offers improved performance compared to UIWebView.

The following example illustrates how you configure an application feature with AMX content in a migrated MAF application to use the newer WKWebView. To revert to using UIWebView, set the `value` attribute to `legacy`. You configure these properties in the `maf-features.xml` file for each application feature with AMX content that you want to use WKWebView.

```
<adfmf:feature id="WKWebViewExample" name="WKWebViewExample">
  <adfmf:constraints>
    <adfmf:constraint property="device.os" operator="contains" value="iOS"
id="c6"/>
  </adfmf:constraints>
  <adfmf:content id="WKWebViewExample.1">
    <adfmf:amx file="WKWebViewExample/home.amx"/>
  </adfmf:content>
  <adfmf:properties id="wkpl">
    <adfmf:property id="wkpl-1" name="iOSWebView" value="modern" />
    <!-- To revert to using UIWebView, set to legacy -->
    <!-- name="iOSWebView" value="legacy" -->
  </adfmf:properties>
</adfmf:feature>
```

Application features that use local HTML or remote URL content types continue to use the UIWebView as this web view supports the `/~maf.device~/` virtual path to access JavaScript APIs.

When the `iOSWebView` property is missing or is set to `default` then WKWebView is used for AMX content and UIWebView is used for local HTML and remote URL content types. You can specifically opt-in to using WKWebView for the local HTML and remote URL content types by setting the value to `modern` if you do not need the `/~maf.device~/` virtual path.

WKWebView is used on iOS 9+ only. UIWebView will always be used on iOS 8.

Security Changes in Release 2.4.0 and Later of MAF

Starting with MAF 2.4.0, MAF defaults the HTTPS protocol to TLSv1.2 on MAF applications that you deploy to the Android platform.

On supported platforms, you can override this behavior by specifying an alternative value as a Java command-line argument in the `maf.properties` file, as shown by the following example that configures the Java VM layer of your application to use TLSv1.1.

```
// Configure Java VM layer of the MAF app to use TLSv1.1
java.commandline.argument=-Dhttps.protocols=TLSv1.1
```

```
// Configure the HTTPS cipher suite(s) that an application uses by
// providing a comma-separated list as a value
java.commandline.argument=-Dhttps.cipherSuites=TLS_RSA_WITH_AES_256_CBC_SHA
```

Android's authentication mechanism honors the properties in the `maf.properties` file if the Android version supports the legacy protocols. Devices running Android 7+, for example, do not support `TLSv1` and disable RC4-based cipher suites.

On the iOS and Universal Windows Platform, specifying these properties in the `maf.properties` file does not change the authentication mechanism of the application. This is managed by the platform itself. However, application code, such as REST calls, may be affected by these properties.

We recommend that you retain the default MAF behavior in your application. Otherwise you may introduce security risks to your application. Overriding the default behavior is described here to assist you if you need to test your application with servers that use older versions of SSL or deprecated cipher suites.

Security Changes in Release 2.2.1 and Later of MAF

Migrating an application from MAF 2.2.0 or earlier to MAF 2.3.0 and later requires you to make some configuration changes to your migrated application so that it adheres to the latest security standards supported by this release of MAF.

Starting with MAF 2.2.1, use of HTTPS with TLS 1.2 for all connections to the server from MAF applications on iOS is required. Any MAF application that uses non-HTTPS connections and an SSL version lower than TLS1.2 will fail to run on iOS. MAF enforces this behavior to meet the Apple iOS requirement to use App Transport Security (ATS) that requires use of HTTPS with TLS 1.2. You can disable use of ATS, as described below.

MAF applications also adhere to the default behavior enforced by the JVM of Java 8 to use the latest SSL version and cipher suites. While we encourage you to upgrade your servers to use these later versions, you can configure your MAF application to work around SSL errors you may encounter by using servers with older SSL versions, as described below.

Disabling App Transport Security for MAF Applications on iOS Devices

MAF applications that you migrate to this release of MAF enable ATS by default. You can disable ATS in your MAF application as follows:

1. In JDeveloper, choose **Application > Application Properties > Deployment**.
2. In the Deployment page, double-click the iOS deployment profile.
3. Select **iOS Options**.
4. Select **Disable Application Transport Security** and click **OK**.

Note:

We recommend that you do not disable ATS. Apple plans to enforce use of ATS from January 01, 2017. MAF applications that disable ATS will not be approved for publication by the Apple App Store.

SSL Configuration Changes

Customers who use SSL versions lower than TLS 1.2, deprecated cipher suites or deprecated encryption algorithms will see SSL errors like "invalid cipher suite", "close notify", "TLS error", and so on. Java 8 enforces use of the latest SSL version and cipher suites. It disables use of insecure SSL versions by default. We encourage you to update your servers to use the later SSL version. If this is not possible, you can use the following configuration to work around the SSL errors just described:

1. Update `maf.properties` file with the version of SSL that you want to use. For example, add the following entry to the `maf.properties` file to use TLS 1:

```
java.commandline.argument=-Dhttps.protocols=TLSv1
```

2. Update `maf.properties` file with the full list of cipher suites required by the application. For the list of cipher suites that Java supports, see the Cipher Suites section on this [page](#).

For example, to enable `SSL_RSA_WITH_RC4_128_MD5`, add the following:

```
java.commandline.argument=-D SSL_RSA_WITH_RC4_128_MD5
```

3. Update the `java.security` file to enable deprecated algorithms. Existing MAF applications will not have this file so create a new empty MAF application and copy the `java.security` file created in the new MAF application's `/resources/security` to the same directory in the existing application.

For example, the RC4 algorithm is disabled by default per the following entry in the `java.security` file:

```
jdk.tls.disabledAlgorithms=SSLv3, RC4, DH keySize < 768
```

If you use a cipher suite that requires the RC4 algorithm, such as `SSL_RSA_WITH_RC4_128_MD5`, an error is thrown at runtime while establishing the SSL connection. To work around this, change the `java.security` entry as follows to enable the RC4 algorithm:

```
jdk.tls.disabledAlgorithms=SSLv3, DH keySize < 768
```

Migrating an Application Developed Using AMPA to MAF 2.5.1

Describes how to migrate an application developed using the A-Team Mobile Persistence Accelerator (AMPA) extension and an earlier release of MAF to this release of MAF.

MAF 2.3.1 and later incorporates AMPA, a persistence and data synchronization framework, as part of the client data model feature in MAF. Application developers can develop new MAF applications using the design-time and runtime features of the MAF client data model to generate the data model of their application, decide what data objects to persist on end user's devices, plus generate a complete user interface from data controls created from the service objects in the generated client data model. See *Creating the Client Data Model in a MAF Application* in *Developing Mobile Applications with Oracle Mobile Application Framework*.

You can migrate applications developed using the AMPA extension and earlier releases of MAF to this release of MAF by performing the following tasks:

- Change the namespace in the `persistence-mapping.xml` file
- Modify the lifecycle listener in the `maf-application.xml` file
- Change the package names of AMPA classes to use the MAF client data model package names

Change the Namespace in the `persistence-mapping.xml` File

Change the namespace in the `persistence-mapping.xml` file to use the MAF client data model value, as shown in the following example:

```
<mobileObjectPersistence xmlns="http://xmlns.oracle.com/adf/mf/amx/cdm/
persistenceMapping" ...
```

The `persistence-mapping.xml` file is in the following directory of the `ApplicationController` project in your migrated application:

```
/ApplicationController/src/META-INF
```

Modify the Lifecycle Listener in the `maf-application.xml` File

Change the value of the `listener-class` attribute in the `maf-application.xml` file from `oracle.ateam.sample.mobile.lifecycle.InitDBLifeCycleListener` to the MAF client data model value, as shown in the following example:

```
<adf:application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:adf="http://xmlns.oracle.com/adf/mf"
...
    listener-class="oracle.maf.impl.cdm.lifecycle.InitDBLifeCycleListener">
```

The `maf-application.xml` file is in the following directory of your migrated application:

```
./adf/META-INF/maf-application.xml
```

Change AMPA Package Name to MAF Client Data Model Package Names

Revise the package names of Java classes in your migrated application to the package names used by MAF client data model. The mapping between AMPA packages and the MAF client data model packages is as follows:

```
oracle.ateam.sample.mobile          -----> oracle.maf.impl.cdm
oracle.ateam.sample.mobile.v2.security -----> oracle.maf.impl.cdm.security
oracle.ateam.sample.mobile.v2.persistence -----> oracle.maf.impl.cdm.persistence
```

If, for example, the AMPA application that you migrate to this release of MAF contains a Java class that imports `oracle.ateam.sample.mobile.mcs.analytics.AnalyticsEvent`, modify the Java class in your migrated application to import `oracle.maf.impl.cdm.mcs.analytics.AnalyticsEvent`.

Note:

Classes in the `oracle.maf.impl.cdm` package are internal classes of the MAF client data model and subject to change. MAF may refactor some of these classes in later releases but, for now, we recommend that you do not extend these classes.

All the classes in the `oracle.maf.api.cdm...` packages are publicly available classes that you can extend. If, for example, the AMPA application that you migrate to this release of MAF contains a Java class that imports

```
oracle.ateam.sample.mobile.mcs.storage.StorageObject, modify it so that it imports
oracle.maf.api.cdm.mcs.storage.StorageObject.
```

The following list identifies the publicly available classes in the `oracle.maf.api.cdm` package:

```
controller.bean.ConnectivityBean
exception.RestCallException
mcs.storage.StorageObject
mcs.storage.StorageObjectService
persistence.cache.EntityCache
persistence.db.BindParamInfo
persistence.manager.DBPersistenceManager
persistence.manager.MCSPersistenceManager
persistence.manager.RestJSONPersistenceManager
persistence.manager.RestXMLPersistenceManager
persistence.metadata.AttributeMapping
persistence.metadata.AttributeMappingDirect
persistence.metadata.AttributeMappingOneToMany
persistence.metadata.AttributeMappingOneToOne
persistence.metadata.ClassMappingDescriptor
persistence.model.Entity
persistence.service.DataSynchAction
persistence.service.DataSynchService
persistence.service.ValueHolderInterface
persistence.util.EntityUtils
```

See the [Java reference documentation](#) for the AMPA framework to identify the package name in AMPA for the list of publicly available classes above and revise to use the package name in the MAF client data model. For information about publicly available classes in MAF, see *Java API Reference for Oracle Mobile Application Framework*.

Also refer to the above reference documentation for other changes implemented in the MAF client data model since it incorporated AMPA. For example, if you extended `DBPersistenceManager` in an application developed using AMPA and the extended class referenced constants, such as `SQL_SELECT_KEYWORD`, you need to supply your own constants in the extended class as the MAF client data model's implementation of `DBPersistenceManager` no longer provides these constants.

Make the above changes for all Java classes in your migrated application and in `pageDefinition.xml` files that reference Java managed beans using the AMPA package names.

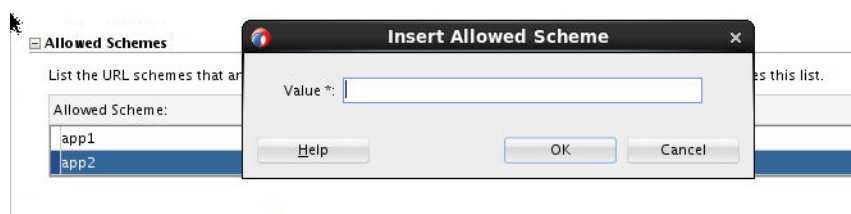
Apart from the above changes, make sure that the application you migrate uses supported classes and methods. For example, AMPA deprecated `oracle.ateam.sample.mobile.util.MCSManager` in a recent release. Any application migrated to this release of MAF which uses the AMPA-deprecated `MCSManager` should be revised to use `oracle.maf.api.cdm.persistence.manager.MCSPersistenceManager`.

Migrating MAF Applications that Use Customer URL Schemes to Invoke Other Applications

If the application you migrate to MAF 2.2.2 or later uses a custom URL scheme to invoke another application, add the scheme(s) to the **Allowed Scheme** list in the Security page of the `maf-application.xml` file's overview editor.

This change addresses the iOS 9 requirement that applications declare any URL schemes they use to invoke other applications. Click the **Add** icon in the Allow Schemes section of the Security page to add the custom URL scheme.

Figure 5-2 Registering a Custom URL Scheme that a MAF Applications Use to Invoke Another Application



Migrating to JDK 8 in MAF 2.5.1

MAF applications that you create in MAF 2.1.0 and later use JDK 8. If you migrate a MAF application that compiled with an earlier version of Java, note that MAF 2.1.0 and later requires JDK 8 and compiles applications using the Java SE Embedded 8 compact2 profile.

When you open an application that you migrated from a pre-MAF 2.1.0 release for the first time, JDeveloper makes the following changes:

- Renames the configuration file that specifies the startup parameters of the JVM from `cvm.properties` to `maf.properties`. For information about the `maf.properties` file, see *How to Enable Debugging of Java Code and JavaScript in Developing Mobile Applications with Oracle Mobile Application Framework*.
- Replaces instances (if any) of the following import statement in the Java source files of the application:

```
com.sun.util.logging
```

With:

```
java.util.logging
```

- Replaces the following entries in the `logging.properties` file of the application

```
.handlers=com.sun.util.logging.ConsoleHandler
.formatter=com.sun.util.logging.SimpleFormatter
```

With:

```
.handlers=java.util.logging.ConsoleHandler
.formatter=java.util.logging.SimpleFormatter
```

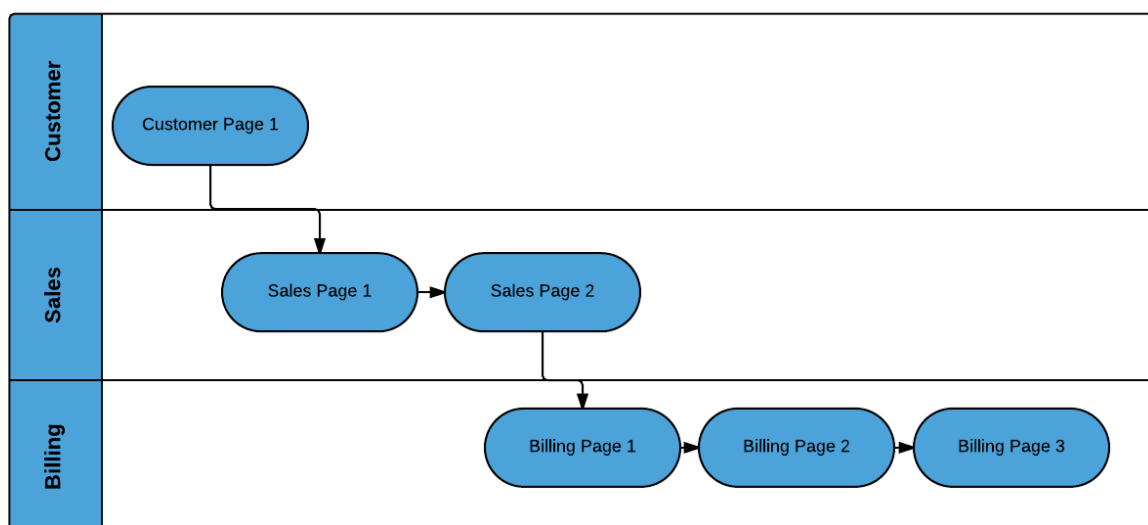
For information about the `logging.properties` file, see *How to Configure Logging Using the Properties File in Developing Mobile Applications with Oracle Mobile Application Framework*.

Retaining Legacy Behavior When Navigating a MAF Application Using Android's Back Button

MAF 2.2.0 introduced a change in the way that MAF applications created using that release respond to usage of the Android system's Back button. A MAF application that you created in a previous release and migrate to MAF 2.2.0 or later uses the new behavior.

The figure shows a navigation flow on a MAF application where an end user has navigated between three application features (Customer, Sales, and Billing) to the Billing Page 3 page of the Billing application feature.

Figure 5-3 Navigation Flow Between Application Features and Pages in a MAF Application



Prior to Release MAF 2.2.0, the default MAF application behavior in response to an end user tapping Android's system Back button on:

- Billing Page 3 was to navigate to the Sales application feature
- Sales application feature was to navigate to the Customers application feature
- Customer application feature was to close the MAF application

In MAF 2.2.0 and later, the default MAF application behavior in response to an end user tapping Android's system Back button on:

- Billing Page 3 is to navigate to Billing Page 2
- Billing Page 2 is to navigate to Billing Page 1

- Billing Page 1 is to hibernate the MAF application

You can customize how your MAF application responds to an end user's tap of the Android system's Back button, as described in Navigating a MAF Application Using Android's Back Button of *Developing Mobile Applications with Oracle Mobile Application Framework*.

You can also configure your MAF application to exhibit the pre-MAF 2.2.0 application behavior (navigate between application features) by setting a property in the `maf-config.xml`, as described in [How to Retain Pre-MAF 2.2.0 Application Behavior in Response to Usage of Android's Back Button](#).

How to Retain Pre-MAF 2.2.0 Application Behavior in Response to Usage of Android's Back Button

You configure the `legacyBack` element in the `maf-config.xml` file to make your MAF application exhibit pre-MAF 2.2.0 behavior when an end user taps Android's Back button.

To Retain Pre-MAF 2.2.0 Application Behavior in Response to Usage of Android's Back Button:

1. In the Applications window, double-click the **maf-config.xml** file.
By default, this is in the Application Resources pane under the Descriptors and ADF META-INF nodes.
2. In the `maf-config.xml` file, set the value of the `legacyBack` element to `true`, as shown in [Example 5-1](#).

Example 5-1 legacyBack element to Retain Pre-MAF 2.2.0 Application Behavior for Usage of Android Back Button

```
<?xml version="1.0" encoding="UTF-8" ?>
<adfmf-config xmlns="http://xmlns.oracle.com/adf/mf/config">
  ...
  <legacyBack>true</legacyBack>
</adfmf-config>
```

Migrating to New cacerts File for SSL in MAF 2.5.1

MAF 2.1.0 delivered a new `cacerts` file for use in MAF applications. Make sure that the `cacerts` file packaged in the application that you publish for end users to install contains the same CA root certificates as the HTTPS server that end users connect to when they use your MAF application.

You may need to import new certificates to the `cacerts` file of your MAF application if the HTTPS server contains certificates not present in the `cacerts` file of your MAF application. Similarly, system administrators for the HTTPS servers that your MAF application connects to may need to import new certificates if your MAF application uses a certificate not present on the HTTPS server.

Use the `keytool` utility of JDK 8 to view and manage the certificates in the `cacerts` file of your MAF application. The following example demonstrates how you might use the `keytool` utility of JDK 8 to display the list of certificates in a `cacerts` file:

```
JDK8install/bin/keytool -list -v -keystore dirPathToCacertsFile/cacerts -storepass
changeit | grep "Issuer:"
```

For information about using the `keytool` utility of JDK 8 to manage certificates, see <http://docs.oracle.com/javase/8/docs/technotes/tools/#security>. For example, to use the `keytool` utility on Windows, see <http://docs.oracle.com/javase/8/docs/technotes/tools/windows/keytool.html>. For UNIX-based operating systems, see <http://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html>.

For information about the `cacerts` file and using SSL to secure your MAF application, see Supporting SSL in *Developing Mobile Applications with Oracle Mobile Application Framework*.

Example 5-2 lists the issuers of CA root certificates included in the MAF 2.1.0 `cacerts` file. Use the `keytool` utility of JDK 8, as previously described, to manage the certificates in this file to meet the requirements of the environment where your MAF application will be used.

Example 5-2 CA Root Certificate Issuers in MAF 2.1.0 cacerts File

```
Issuer: CN=DigiCert Assured ID Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US
Issuer: CN=TC TrustCenter Class 2 CA II, OU=TC TrustCenter Class 2 CA, O=TC TrustCenter GmbH, C=DE
Issuer: EMAILADDRESS=premium-server@thawte.com, CN=Thawte Premium Server CA, OU=Certification
Services Division, O=Thawte Consulting cc, L=Cape Town, ST=Western Cape, C=ZA
Issuer: CN=SwissSign Platinum CA - G2, O=SwissSign AG, C=CH
Issuer: CN=SwissSign Silver CA - G2, O=SwissSign AG, C=CH
Issuer: EMAILADDRESS=server-certs@thawte.com, CN=Thawte Server CA, OU=Certification Services
Division, O=Thawte Consulting cc, L=Cape Town, ST=Western Cape, C=ZA
Issuer: CN=Equifax Secure eBusiness CA-1, O=Equifax Secure Inc., C=US
Issuer: CN=SecureTrust CA, O=SecureTrust Corporation, C=US
Issuer: CN=UTN-USERFirst-Client Authentication and Email, OU=http://www.usertrust.com, O=The
USERTRUST Network, L=Salt Lake City, ST=UT, C=US
Issuer: EMAILADDRESS=personal-freemail@thawte.com, CN=Thawte Personal Freemail CA, OU=Certification
Services Division, O=Thawte Consulting, L=Cape Town, ST=Western Cape, C=ZA
Issuer: CN=AffirmTrust Networking, O=AffirmTrust, C=US
Issuer: CN=Entrust Root Certification Authority, OU="(c) 2006 Entrust, Inc.", OU=www.entrust.net/CPS
is incorporated by reference, O="Entrust, Inc.", C=US
Issuer: CN=UTN-USERFirst-Hardware, OU=http://www.usertrust.com, O=The USERTRUST Network, L=Salt Lake
City, ST=UT, C=US
Issuer: CN=Certum CA, O=Unizeto Sp. z o.o., C=PL
Issuer: CN=AddTrust Class 1 CA Root, OU=AddTrust TTP Network, O=AddTrust AB, C=SE
Issuer: CN=Entrust Root Certification Authority - G2, OU="(c) 2009 Entrust, Inc. - for authorized use
only", OU=See www.entrust.net/legal-terms, O="Entrust, Inc.", C=US
Issuer: OU=Equifax Secure Certificate Authority, O=Equifax, C=US
Issuer: CN=QuoVadis Root CA 3, O=QuoVadis Limited, C=BM
Issuer: CN=QuoVadis Root CA 2, O=QuoVadis Limited, C=BM
Issuer: CN=DigiCert High Assurance EV Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US
Issuer: EMAILADDRESS=info@valicert.com, CN=http://www.valicert.com/, OU=ValiCert Class 1 Policy
Validation Authority, O="ValiCert, Inc.", L=ValiCert Validation Network
Issuer: CN=Equifax Secure Global eBusiness CA-1, O=Equifax Secure Inc., C=US
Issuer: CN=GeoTrust Universal CA, O=GeoTrust Inc., C=US
Issuer: OU=Class 3 Public Primary Certification Authority, O="VeriSign, Inc.", C=US
Issuer: CN=thawte Primary Root CA - G3, OU="(c) 2008 thawte, Inc. - For authorized use only",
OU=Certification Services Division, O="thawte, Inc.", C=US
Issuer: CN=thawte Primary Root CA - G2, OU="(c) 2007 thawte, Inc. - For authorized use only",
O="thawte, Inc.", C=US
Issuer: CN=Deutsche Telekom Root CA 2, OU=T-TeleSec Trust Center, O=Deutsche Telekom AG, C=DE
Issuer: CN=Buypass Class 3 Root CA, O=Buypass AS-983163327, C=NO
Issuer: CN=UTN-USERFirst-Object, OU=http://www.usertrust.com, O=The USERTRUST Network, L=Salt Lake
City, ST=UT, C=US
Issuer: CN=GeoTrust Primary Certification Authority, O=GeoTrust Inc., C=US
Issuer: CN=Buypass Class 2 Root CA, O=Buypass AS-983163327, C=NO
Issuer: CN=Baltimore CyberTrust Code Signing Root, OU=CyberTrust, O=Baltimore, C=IE
Issuer: OU=Class 1 Public Primary Certification Authority, O="VeriSign, Inc.", C=US
```

Issuer: CN=Baltimore CyberTrust Root, OU=CyberTrust, O=Baltimore, C=IE
Issuer: OU=Starfield Class 2 Certification Authority, O="Starfield Technologies, Inc.", C=US
Issuer: CN=Chambers of Commerce Root, OU=http://www.chambersign.org, O=AC Camerfirma SA CIF A82743287, C=EU
Issuer: CN=T-TeleSec GlobalRoot Class 3, OU=T-Systems Trust Center, O=T-Systems Enterprise Services GmbH, C=DE
Issuer: CN=VeriSign Class 3 Public Primary Certification Authority - G5, OU="(c) 2006 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US
Issuer: CN=T-TeleSec GlobalRoot Class 2, OU=T-Systems Trust Center, O=T-Systems Enterprise Services GmbH, C=DE
Issuer: CN=TC TrustCenter Universal CA I, OU=TC TrustCenter Universal CA, O=TC TrustCenter GmbH, C=DE
Issuer: CN=VeriSign Class 3 Public Primary Certification Authority - G4, OU="(c) 2007 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US
Issuer: CN=VeriSign Class 3 Public Primary Certification Authority - G3, OU="(c) 1999 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US
Issuer: CN=XRamp Global Certification Authority, O=XRamp Security Services Inc, OU=www.xrampsecurity.com, C=US
Issuer: CN=Class 3P Primary CA, O=Certplus, C=FR
Issuer: CN=Certum Trusted Network CA, OU=Certum Certification Authority, O=Unizeto Technologies S.A., C=PL
Issuer: OU=VeriSign Trust Network, OU="(c) 1998 VeriSign, Inc. - For authorized use only", OU=Class 3 Public Primary Certification Authority - G2, O="VeriSign, Inc.", C=US
Issuer: CN=GlobalSign, O=GlobalSign, OU=GlobalSign Root CA - R3
Issuer: CN=UTN - DATACorp SGC, OU=http://www.usertrust.com, O=The USERTRUST Network, L=Salt Lake City, ST=UT, C=US
Issuer: OU=Security Communication RootCA2, O="SECOM Trust Systems CO.,LTD.", C=JP
Issuer: CN=GTE CyberTrust Global Root, OU="GTE CyberTrust Solutions, Inc.", O=GTE Corporation, C=US
Issuer: OU=Security Communication RootCA1, O=SECOM Trust.net, C=JP
Issuer: CN=AffirmTrust Commercial, O=AffirmTrust, C=US
Issuer: CN=TC TrustCenter Class 4 CA II, OU=TC TrustCenter Class 4 CA, O=TC TrustCenter GmbH, C=DE
Issuer: CN=VeriSign Universal Root Certification Authority, OU="(c) 2008 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US
Issuer: CN=GlobalSign, O=GlobalSign, OU=GlobalSign Root CA - R2
Issuer: CN=Class 2 Primary CA, O=Certplus, C=FR
Issuer: CN=DigiCert Global Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US
Issuer: CN=GlobalSign Root CA, OU=Root CA, O=GlobalSign nv-sa, C=BE
Issuer: CN=thawte Primary Root CA, OU="(c) 2006 thawte, Inc. - For authorized use only", OU=Certification Services Division, O="thawte, Inc.", C=US
Issuer: CN=Starfield Root Certificate Authority - G2, O="Starfield Technologies, Inc.", L=Scottsdale, ST=Arizona, C=US
Issuer: CN=GeoTrust Global CA, O=GeoTrust Inc., C=US
Issuer: CN=Sonera Class2 CA, O=Sonera, C=FI
Issuer: CN=Thawte Timestamping CA, OU=Thawte Certification, O=Thawte, L=Durbanville, ST=Western Cape, C=ZA
Issuer: CN=Sonera Class1 CA, O=Sonera, C=FI
Issuer: CN=QuoVadis Root Certification Authority, OU=Root Certification Authority, O=QuoVadis Limited, C=BM
Issuer: CN=AffirmTrust Premium ECC, O=AffirmTrust, C=US
Issuer: CN=Starfield Services Root Certificate Authority - G2, O="Starfield Technologies, Inc.", L=Scottsdale, ST=Arizona, C=US
Issuer: EMAILADDRESS=info@valicert.com, CN=http://www.valicert.com/, OU=ValiCert Class 2 Policy Validation Authority, O="ValiCert, Inc.", L=ValiCert Validation Network
Issuer: CN=AAA Certificate Services, O=Comodo CA Limited, L=Salford, ST=Greater Manchester, C=GB
Issuer: CN=America Online Root Certification Authority 2, O=America Online Inc., C=US
Issuer: CN=AddTrust Qualified CA Root, OU=AddTrust TTP Network, O=AddTrust AB, C=SE
Issuer: CN=KEYNECTIS ROOT CA, OU=ROOT, O=KEYNECTIS, C=FR
Issuer: CN=America Online Root Certification Authority 1, O=America Online Inc., C=US
Issuer: CN=VeriSign Class 2 Public Primary Certification Authority - G3, OU="(c) 1999 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US
Issuer: CN=AddTrust External CA Root, OU=AddTrust External TTP Network, O=AddTrust AB, C=SE
Issuer: OU=VeriSign Trust Network, OU="(c) 1998 VeriSign, Inc. - For authorized use only", OU=Class 2

Public Primary Certification Authority - G2, O="VeriSign, Inc.", C=US
Issuer: CN=GeoTrust Primary Certification Authority - G3, OU=(c) 2008 GeoTrust Inc. - For authorized use only, O=GeoTrust Inc., C=US
Issuer: CN=GeoTrust Primary Certification Authority - G2, OU=(c) 2007 GeoTrust Inc. - For authorized use only, O=GeoTrust Inc., C=US
Issuer: CN=SwissSign Gold CA - G2, O=SwissSign AG, C=CH
Issuer: CN=Entrust.net Certification Authority (2048), OU=(c) 1999 Entrust.net Limited, OU=www.entrust.net/CPS_2048 incorp. by ref. (limits liab.), O=Entrust.net
Issuer: OU=ePKI Root Certification Authority, O="Chunghwa Telecom Co., Ltd.", C=TW
Issuer: CN=Global Chambersign Root - 2008, O=AC Camerfirma S.A., SERIALNUMBER=A82743287, L=Madrid (see current address at www.camerfirma.com/address), C=EU
Issuer: CN=Chambers of Commerce Root - 2008, O=AC Camerfirma S.A., SERIALNUMBER=A82743287, L=Madrid (see current address at www.camerfirma.com/address), C=EU
Issuer: OU=Go Daddy Class 2 Certification Authority, O="The Go Daddy Group, Inc.", C=US
Issuer: CN=AffirmTrust Premium, O=AffirmTrust, C=US
Issuer: CN=VeriSign Class 1 Public Primary Certification Authority - G3, OU="(c) 1999 VeriSign, Inc. - For authorized use only", OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US
Issuer: OU=Security Communication EV RootCA1, O="SECOM Trust Systems CO.,LTD.", C=JP
Issuer: OU=VeriSign Trust Network, OU="(c) 1998 VeriSign, Inc. - For authorized use only", OU=Class 1 Public Primary Certification Authority - G2, O="VeriSign, Inc.", C=US
Issuer: CN=Go Daddy Root Certificate Authority - G2, O="GoDaddy.com, Inc.", L=Scottsdale, ST=Arizona, C=US