Oracle® Mobile Application Framework

Installing Oracle Mobile Application Framework

2.4.0.0.0

E81104-01

March 2017

Documentation that describes how to install the Oracle Mobile Application Framework for use with Oracle JDeveloper to create mobile applications that run natively on devices.

ORACLE

Oracle Mobile Application Framework Installing Oracle Mobile Application Framework, 2.4.0.0.0

E81104-01

Primary Authors: Walter Egan, Sujatha Joseph

Contents

3 Migrating Your Application to MAF 2.4.0

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.
italic	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.

What's New in This Guide for Release 2.4.0

Section	Change
Setting Up Development Tools for the Android Platform	Revised to describe the Android Support Repository package that you must install in the Android SDK to ensure a successful deployment to the Android platform.
Setting Up Development Tools from the Command Line Using Startup Parameters	List the startup parameters that you can use to set the MAF preferences required to develop and deploy MAF applications on the Android and iOS platforms.
Security Changes in Release 2.4.0 and Later of MAF	Describes how you can override the default MAF behavior where MAF applications that you deploy to the Android platform use TLSv1.2 of the HTTPS protocol.

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:

1.1 Introduction to Installing the MAF Extension with JDeveloper

The development of MAF applications requires that you install Oracle JDeveloper and the MAF extension. Also configure additional development tools for the platforms on which you intend to deploy the applications.

The first step in starting with MAF application development is to install Oracle JDeveloper and the MAF extension.

Install JDeveloper, and then install the MAF extension in JDeveloper.

Following the installation of the MAF extension in JDeveloper, configure additional development tools for the platforms where you intend to deploy your MAF application. See Setting Up the Development Environment .

Before you create a MAF application using the MAF extension in JDeveloper, ensure that you have the third-party software that is required to develop applications for the platform on which you intend to deploy your MAF application.

Note:

You can deploy a MAF application to all supported platforms without changing any code. You need the third-party software to test, debug, and deploy the MAF application on the target platform.

1.2 Installation Requirements for MAF Applications to be Deployed to the iOS Platform

In addition to a computer running Apple Mac OS X and installations of JDeveloper and MAF, you need Xcode, iOS SDK, JDK1.8 to develop applications and deploy them to the iOS platform.

To develop a MAF application for deployment to the iOS platform, you require the following:

- A computer running Apple Mac OS X

  For the supported versions of operating systems that you can use to develop MAF applications for deployment to the iOS platform, see Certification Information on Oracle Technology Network at:

  http://www.oracle.com/technetwork/developer-tools/maf/documentation/index.html

- Oracle JDeveloper (See Setting Up JDeveloper.)
- Oracle JDeveloper extension for MAF (See Installing the MAF Extension in JDeveloper.)
- Xcode and iOS SDK (See How to Install Xcode and iOS SDK.)
- The most recent version of JDK1.8.

Before you deploy an application to a development environment, see Getting Started with MAF Application Development. Opt for deployment to a mobile device or its simulator. If you want to use a simulator, see How to Set Up an iPhone or iPad Simulator. If you want to use a mobile device, ensure that in addition to the components included in the preceding list, you have the following available:

- Various login credentials. See Deploying MAF Applications.
- iOS-powered device. See How to Set Up an iPhone or iPad.

1.3 Installation Requirements for MAF Applications to be Deployed to the Android Platform

In addition to installations of JDeveloper and MAF, you need Android SDK Manager, JDK1.8, and an Android-powered device to develop applications and deploy them to the Android platform.

To develop a MAF application for deployment to the Android platform, you require the following:

- A computer running one of the following operating systems:
  - Microsoft Windows
  - Mac OS X

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

- The most recent version of JDK 1.8
- Android SDK Manager (See Setting Up Development Tools for the Android Platform.)
- Oracle JDeveloper (See Setting Up JDeveloper.)
- Oracle JDeveloper extension for MAF (See Installing the MAF Extension in JDeveloper.)

Before you deploy an application to a development environment, see Getting Started with MAF Application Development. Opt for deployment to a mobile device or its emulator. If you want to use an emulator, see Setting up Development Tools for the Android Platform. If you want to use a mobile device, ensure that in addition to the components in the preceding list, you have the following:

- Various login credentials. See Deploying MAF Applications.
- Android-powered device. See Setting Up Development Tools for the Android Platform.

1.4 Installation Requirements for MAF Applications to be Deployed to the Universal Windows Platform

In addition to a computer with x86 architecture running the Windows 10 OS and installations of JDeveloper 12.2.1.0.0 and MAF, you need Microsoft Visual Studio 2015 and JDK 1.8 develop applications and deploy them to the Windows platform.

To develop a MAF for deployment to the Universal Windows Platform (UWP), you require the following:

- A computer with x86 architecture running the Windows 10 operating system
- Microsoft Visual Studio 2015 (Enterprise, Professional, or Community edition)
  - MSBuild 14.0 (automatically installed with Visual Studio 2015)
  - Visual Studio Tools for Universal Windows Apps (an optional component when installing Visual Studio 2015)

    Visual Studio 2015 is available at: https://www.visualstudio.com/products/vs-2015-product-editions.
- JDeveloper 12.2.1.0.0
- JDK 1.8
- Oracle JDeveloper extension for MAF

After you have installed JDeveloper and the MAF extension, perform the tasks listed in Setting Up Development Tools for the Universal Windows Platform.

1.5 Setting Up JDeveloper

Follow the steps in the tasks to install JDeveloper on computers running the Windows or the Mac OS X platforms.

- Download and install the latest version of JDK 1.8.

  Note: 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 the two JAR files of Oracle JDeveloper 12c 12.2.1.0.0 Generic/Others available at the Downloads for JDeveloper:

  http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html

To develop Mobile Application Framework (MAF) applications, you need to install Oracle JDeveloper and its MAF extension.

To install JDeveloper from CLI on a computer running the Windows platform:

1. Create an installation folder.
2. Within the installation folder, place the folder that contains the JDeveloper jar files and the folder containing JDK 1.8.
3. Right-click the folder with the JDeveloper jar files, and click **CMD Prompt Here As Administrator**.

   The Command-line interface (CLI) opens.
4. Invoke the JDeveloper installer with the following command:

   ```
   \PathtoJDK8\bin\java.exe -jar jdev_suite_xxxxxx.jar
   ```

   For example: D:\jdevInstall\jdk1.8.0_51\bin\java.exe -jar jdev_suite_122100.jar

   The JDeveloper installation wizard is launched.

To install JDeveloper on a computer running the Mac OS X platform:

1. Open a Terminal window.
2. Set the `JAVA_HOME` to Java 1.8 by running the following command:

   ```
   export JAVA_HOME=$(/Library/Java/JavaVirtualMachines/
   jdk1.8.x_x.jdk/Contents/Home)
   ```
3. Verify that Java 1.8 is used by running the following command:

   ```
   java -version
   ```
4. Using the same Terminal window, install JDeveloper by executing the following:

   ```
   java -jar <JDEV_12.2.1_jar>
   ```

To verify the installation of JDeveloper:

1. Check the `<JDEV_HOME>\jdev\bin\jdev.conf` file and confirm that the `SetJavaHome` property points to JDK 1.8.
2. Start JDeveloper and select the Studio Developer (All Features) role when prompted.
3. Click **Help**, **About**, and then **Version** to view the versions of the JDeveloper and JDK that are installed.

For more information about installing JDeveloper, see Roadmap for Installing Oracle JDeveloper in *Installing Oracle JDeveloper*.

1.6 Installing the MAF Extension in JDeveloper

Download and install the MAF extension using the Check for Updates menu in JDeveloper.

Once you have installed the MAF extension, you need to configure additional development tools for the platforms on which you intend to deploy your MAF application. See Setting Up the Development Environment .

Note: Close any existing MAF applications you have open in JDeveloper before you install a newer version of the MAF extension. Do this to make sure that MAF migrates your existing application to successfully use the new version of MAF. Verify that the application no longer appears in the Applications window of JDeveloper. For more information about migrating a MAF application, see Migrating Your Application to MAF 2.4.0.

To download and install the MAF extension:

1. In JDeveloper, select **Help**, and then **Check for Updates**.

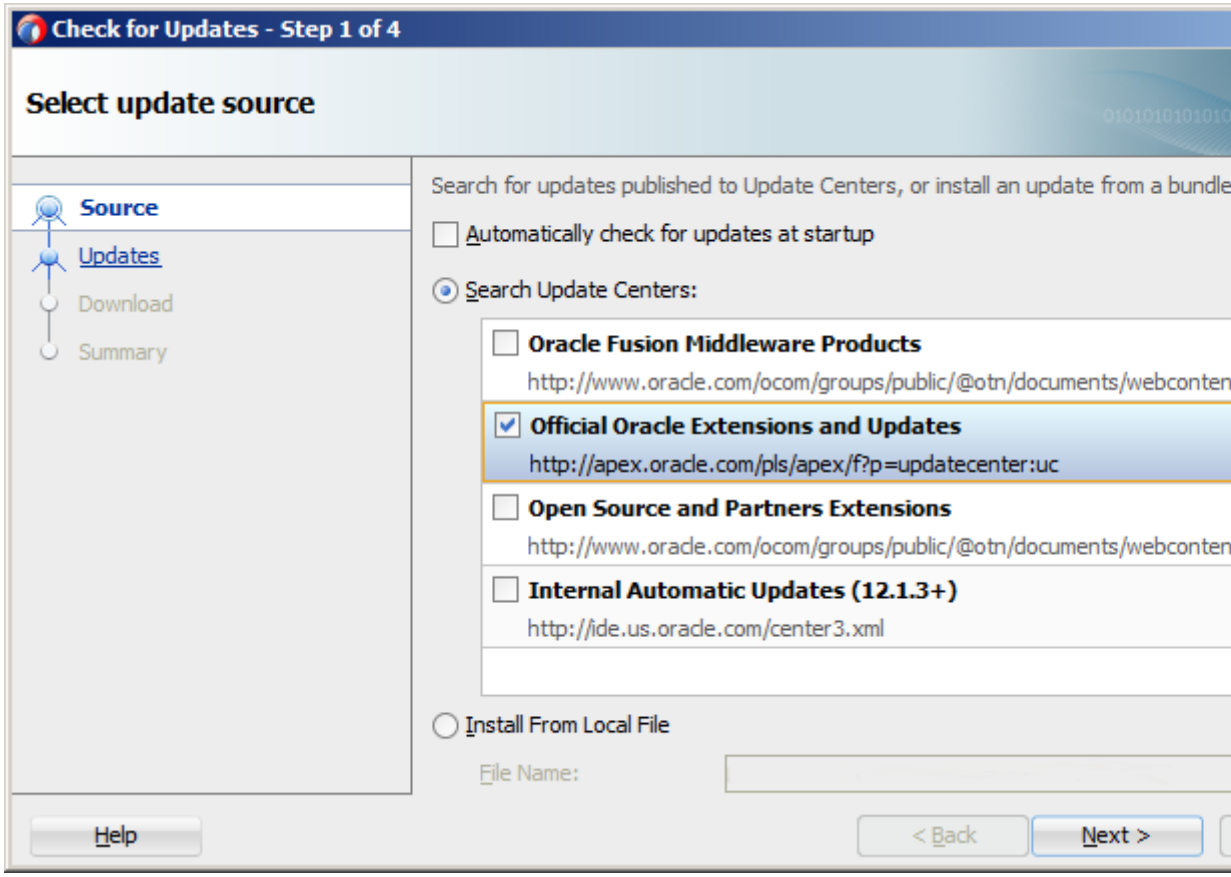
Note:

You might need to configure proxy settings on your development computer. On Windows, select **Tools** and then **Preferences** from the main menu, and then **Web Browser and Proxy** from the tree on the left of the Preferences dialog. On Mac OS X, this option is accessed by selecting **JDeveloper** and then **Preferences**.

---

2. On the Select update source page that Figure 1-1 shows, select **Official Oracle Extensions and Updates** under **Search Update Centers**, and then click **Next**.

Figure 1-1 Checking for Updates in JDeveloper



Alternatively, if network access is unavailable, select the **Install From Local File** option. In this case, you need to point to the MAF extension file that you already downloaded to a directory on your development computer.

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**.

Note:

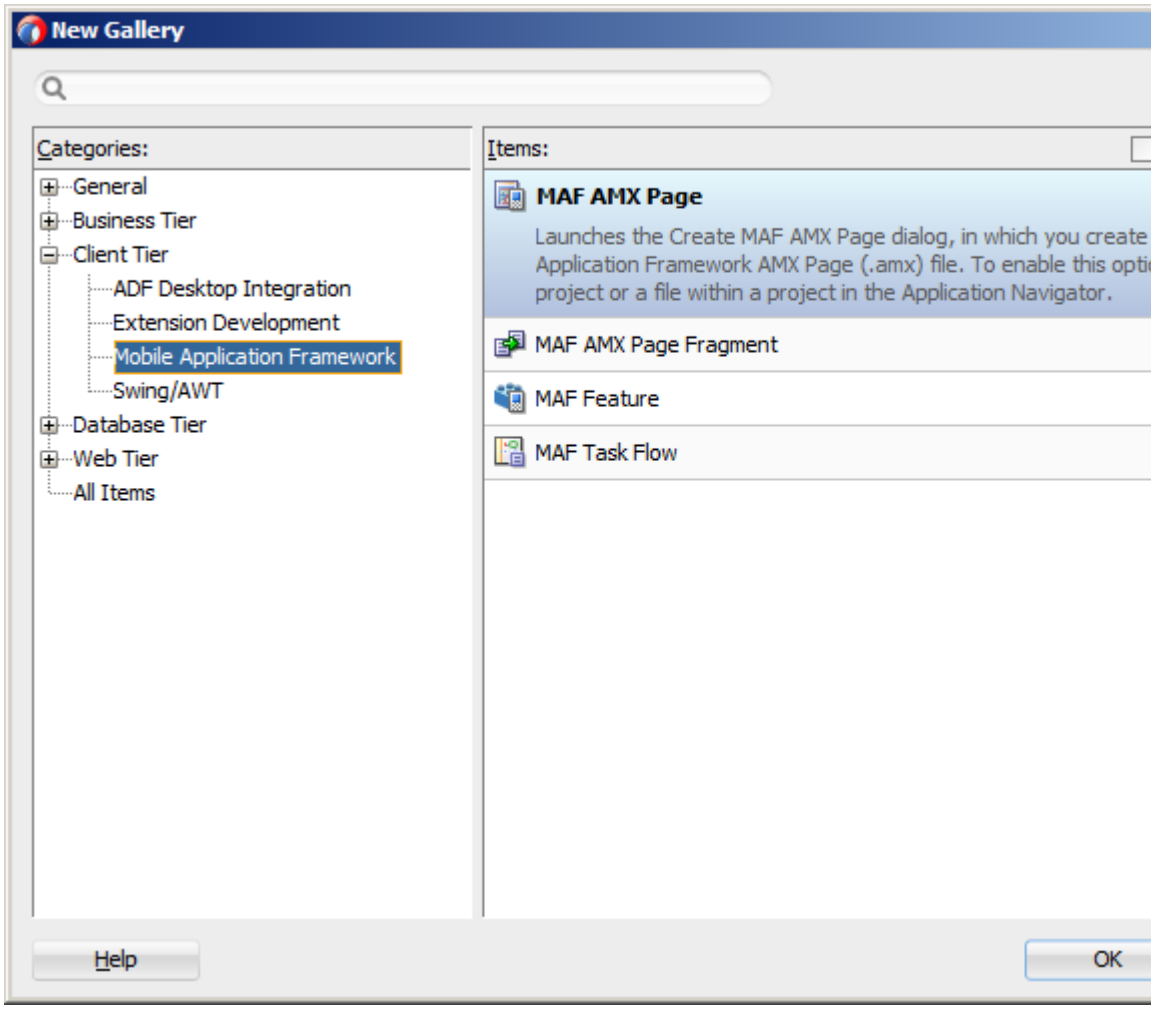
You must comply with all the license terms and conditions with respect to the Oracle Mobile Application Framework Program. See `http://www.oracle.com/technetwork/indexes/downloads/index.html`.

---

5. Click **Next**, and then click **Finish**.
6. Restart JDeveloper.
7. Check whether MAF has been successfully added to JDeveloper:
   - Select **File**, then **New**, and then **From Gallery** from the main menu to open the New Gallery dialog.

- In the **Categories** tree on the left, expand the **Client Tier** node to check if it contains **Mobile Application Framework** (see Figure 1-2).

Figure 1-2 Verifying MAF Installation



To verify whether you have installed the correct version of MAF, in About Oracle JDeveloper, click **Help,** and then **About**. Click **Extensions**, and search for **Mobile Application Framework** in the extension list entries. Select the Version column for inclusion in the table so that the table displays the version number of the MAF extension.

In addition to the preceding steps, the development environment must be configured for target platforms and form factors. See Setting Up the Development Environment .

2

Setting Up the Development Environment

This chapter provides information on setting up and configuring the MAF environment for application development and deployment.

This chapter includes the following sections:

2.1 Introduction to the MAF Development Environment

The development environment for MAF applications requires that you install JDeveloper and the MAF extension, configure form factors, and install and configure third-party tools to package and deploy the applications to supported platforms.

After you install JDeveloper and the MAF extension, as described in Installing Mobile Application Framework with JDeveloper , configure the development environment for the platforms to which you want your MAF application deployed. Configure form factors if you want to test or deploy applications on a particular mobile device. Install and configure third-party tools if you want to package and deploy your MAF application on supported platforms.

For the complete list of supported versions of development and runtime tools, see Certification Information in Oracle Mobile Application Framework Documentation on Oracle Technology Network at: http://www.oracle.com/technetwork/developer-tools/maf/documentation/index.html.

2.2 Configuring the Development Environment for Target Platforms

Configure the platform-specific settings to package and deploy applications to the Android, iOS, or Windows platforms.

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

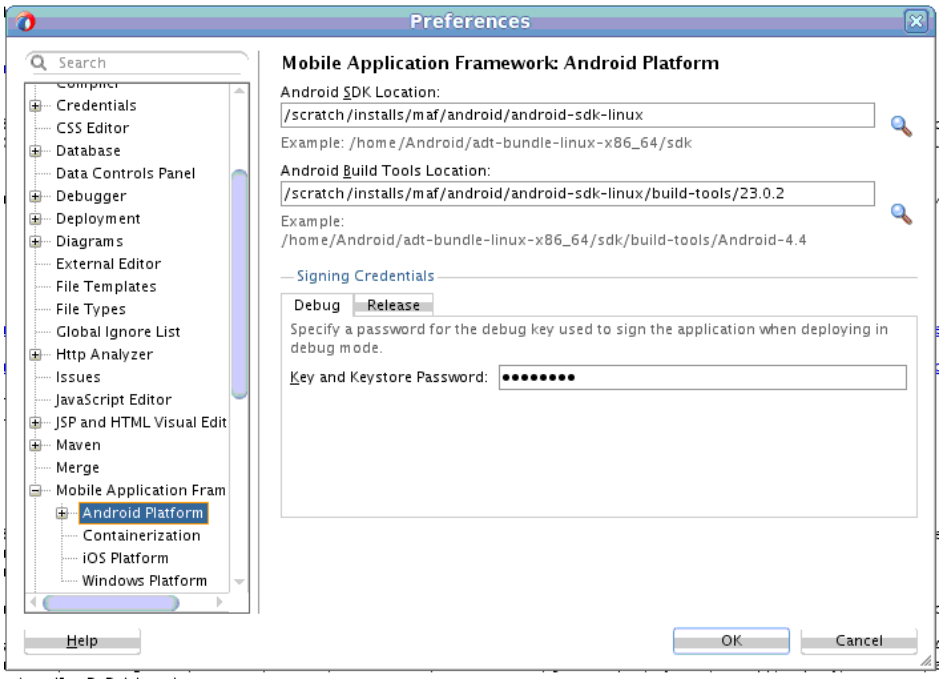
(Android, iOS, or Windows), collecting information such as the path that MAF needs to compile and deploy Android, iOS, or Windows projects. Depending on several factors related to application signing, you may need to edit some of the fields.

- Download and install JDeveloper and the MAF extension. See Installing Mobile Application Framework with JDeveloper .
- For the Android platform, download and configure the Android SDK. (See How to Install the Android SDK.)
- For the iOS platform, download and configure the iOS SDK and Xcode. (See How to Install Xcode and iOS SDK.)
- For the Universal Windows Platform (UWP), download and configure the Windows SDK by installing Microsoft Visual Studio 2015. See Setting Up Development Tools for the Universal Windows Platform.

To configure your environment for target platforms:

1. In JDeveloper, click **Tools** , and then click **Preferences** to open Preferences.
2. In the Preferences dialog, click **Mobile Application Framework** , and then click the platform you want to configure (Android, iOS, or Windows) to open a page that contains the path and configuration parameters for the supported platforms. Fig.1 , for example, shows the page where you configure the platform preferences for Android platform.
    - For the Android platform, specify the following:
        - The Android SDK location on your computer
        - The Android build tools location on your computer
        - Information on the signing credentials

Figure 2-1 Configuring Platform Preferences for Android



- For the iOS platform, specify the following:
  - Location of the iTunes media files, including the mobile applications that are synchronized to the iOS-powered device
  - The iOS-powered device signing information (See Setting the Device Signing Options in *Developing Mobile Applications with Oracle Mobile Application Framework*.)

3. For the Windows platform, specify the following:
   a. In Windows SDK Location, specify the path to the location of MSBuild version 14.0.
   b. Select **Debug** or **Release**, and specify the certificate location and password for the selected mode.

Note: The Hash Algorithm is currently not used.

2.3 Configuring the Development Environment for Form Factors

A form factor is a specific device configuration. Each form factor is identified by a name that you specify for it, and contains information on the specified resolution denoted by pixel width and pixel height.

Since form factors that are defined in preferences are used in the MAF AMX page Preview tab (see Using the Preview in *Developing Mobile Applications with Oracle Mobile Application Framework*), you may choose to perform this configuration if you are planning to include a MAF AMX application feature as part of your MAF application and you do not want to accept the default settings. During development, you can select or switch between various form factors to see how a MAF AMX page is rendered. You can also see multiple form factors applied to the same page using the split screen view.

See About the maf-config.xml File in *Developing Mobile Applications with Oracle Mobile Application Framework*.

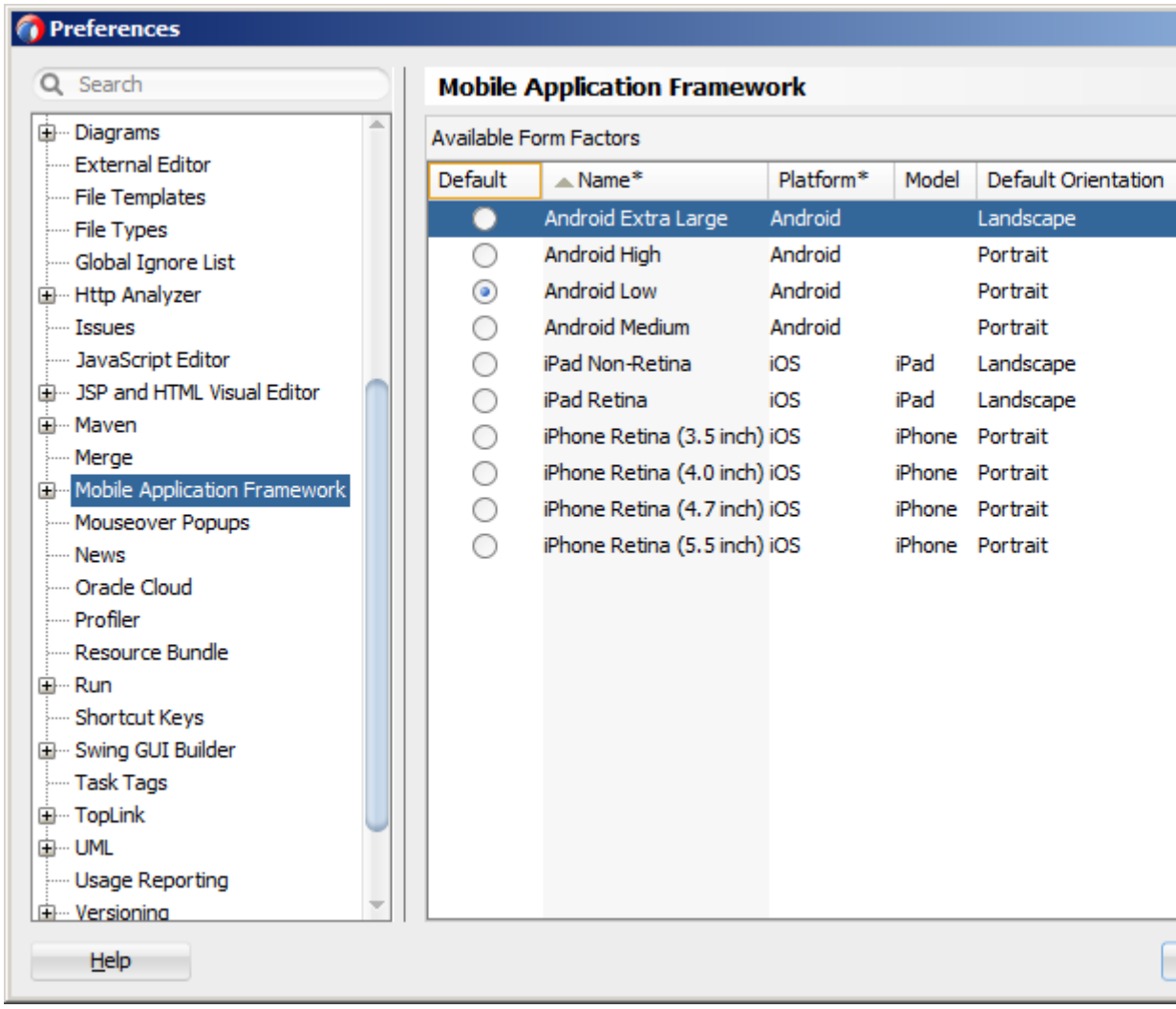
Before you begin:

Download and install JDeveloper and the MAF extension, as described in Installing Mobile Application Framework with JDeveloper .

To configure the form factors:

1. From the main menu in JDeveloper, click **Tools**, and then click **Preferences** .
2. In the **Preferences** dialog that Figure 2-2 opens, select **Mobile Application Framework** from the tree on the left.

Figure 2-2 Defining Form Factors



The **Mobile Application Framework** page is populated with available form factors and the default is set to Android Low.

This preference page allows you to create and manage a set of named form factors that combine a screen resolution size and platform.

3. To create a new form factor, click the green plus sign (New), and then set the following:

   - **Name**: a meaningful string that is used to identify the form factor.
   - **Platform**: the platform of the mobile device.
   - **Model**: the type of the mobile device.
   - **Default Orientation**: the default device orientation used in the MAF AMX page Preview tab. It might be Portrait or Landscape. Select this setting from the drop-down list of values. The default value is Portrait and it is pre-populated during the creation of the new form factor.
   - **Width**: width, in pixels. This value must be a positive integer, and its input is validated.

- **Height**: height, in pixels. This value must be a positive integer, and its input is validated.
- **Scale Factor**: the display scale factor. This value must be either one of 1.0, 2.0, or 3.0.

Note:

If you fail to set the name and resolution for your form, MAF displays an error message.

4. If you need to revert to the default settings, click **More Actions**, and then click **Restore Defaults**.
5. Click **OK** to commit your configuration.

2.4 Setting Up Development Tools for the iOS Platform

Set up the iOS devices or install XCode containing iOS SDK with simulators to equip the environment for the deployment of applications to the iOS platform.

In addition to the general-purpose tools listed in Introduction to Installing the MAF Extension with JDeveloper, you might want to set up an iPhone or iPad when getting ready for the development of a MAF application for the iOS platform (see How to Set Up an iPhone or iPad).

Since iPhone and iPad simulators are included in the iOS SDK installation, which, in turn, is included in the Xcode installation, you need not install them separately. See How to Set Up an iPhone or iPad Simulator.

2.4.1 How to Install Xcode and iOS SDK

Follow the instructions to download Xcode, which includes the iOS SDK.

You download Xcode from `http://developer.apple.com/xcode/`. This download includes the iOS SDK.

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:

Since older versions of Xcode and iOS SDK are not available from the Mac App Store, in order to download them you must obtain an Apple ID from `http://appleid.apple.com`, and then register this Apple ID with the Apple Developer Program to access the Apple developer site at `http://developer.apple.com`.

2.4.2 How to Set Up an iPhone or iPad

Connect an iOS-powered device, with a valid license, certificates, and distribution profile, to your computer to deploy an application to the device.

In your MAF application development and deployment, you can use either the iPhone, iPad, or their simulators (see How to Set Up an iPhone or iPad Simulator). If you plan to use an iPhone or iPad, which is preferable for testing (see Testing MAF

Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*), connect the device to your computer to establish a link between the two devices.

To deploy to an iOS-powered device, you need to have an iOS-powered device with a valid license, certificates, and distribution profiles. See Deploying Mobile Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*.

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.

2.4.3 How to Set Up an iPhone or iPad Simulator

Configure external network access to use iOS simulators, included in XCode downloads, to deploy MAF applications.

In your MAF application development and deployment, you can use either the iOS-powered device itself (see How to Set Up an iPhone or iPad) or its simulator. 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.

A simulator can be invoked automatically, without any additional setup.

Note:

Before attempting to deploy your application from JDeveloper to a device simulator, you must first run the simulator.

If you plan to use web services in your application, and you are behind a corporate firewall, you may need to configure external network access. You do so by modifying the network settings in the System Preferences on your development computer. See Configuring the Browser Proxy Information in *Developing Mobile Applications with Oracle Mobile Application Framework*.

2.5 Setting Up Development Tools for the Android Platform

Once you install the Android SDK, use its SDK manager to install platform tools, a version of the Android platform, such as API 23, and the Android Support Repository.

In addition to the general-purpose tools listed in Introduction to Installing the MAF Extension with JDeveloper, you may want to set up an Android-powered device when getting ready for the development of a MAF application for the Android platform. See How to Set Up an Android-Powered Device.

Since emulators are included in the Android SDK installation, you do not need to separately install them. However, you cannot use an emulator until you configure it. See How to Set Up an Android Emulator.

To develop applications for the Android platform, you can use any operating system that is supported by both JDeveloper and Android.

See the Developer Tools section of the Android Developers website at `http://developer.android.com/tools/index.html`.

2.5.1 How to Install the Android SDK

Android SDK includes development tools that you need to build applications for Android-powered devices. Since the Android SDK is modular, it allows you to download components separately depending on your target Android platform and your application requirements.

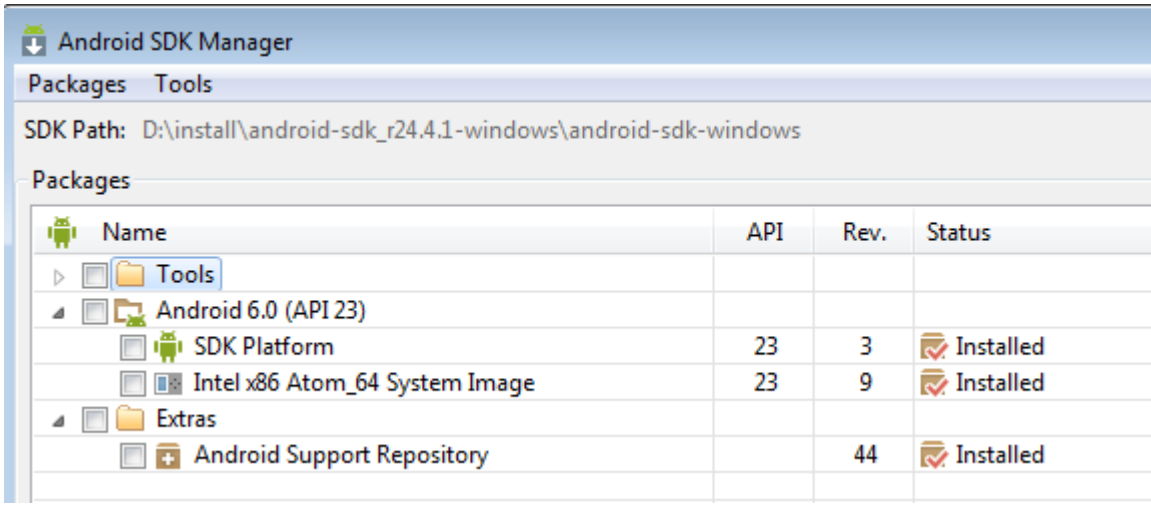
Before you begin:

Ensure that your environment meets the operating system, JDK version, and hardware requirements listed in the Get the Android SDK section of the Android Developers website at `http://developer.android.com/sdk/index.html`.

To install the Android SDK:

1. Download the Android SDK starter package from `http://developer.android.com/sdk/index.html`.
2. Complete the installation by following the instructions provided in the Setting Up an Existing IDE section of the Android Developers website at `http://developer.android.com/sdk/installing.html`.
3. Start the SDK Manager and install the packages that you require to deploy your MAF application to the Android platform. Figure 2-3 shows a list of packages installed that enable deployment of a MAF application to API 23 of the Android platform.

Figure 2-3 SDK Manager with SDK Platform for API 23 and the Android Support Repository Installed



2.5.2 How to Set Up an Android-Powered Device

Follow the instructions on setting up devices on the Android Developers website , and then, following the steps in the task, edit the `android_winusb.inf` file with the values for your device to deploy a MAF application to an Android-powered device.

In your MAF application development and deployment, you can use either the Android device itself, which is preferable for testing (see Testing MAF Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*), or an emulator (see How to Set Up an Android Emulator).

For information on how to set up the Android-powered device, follow the instructions from the Using Hardware Devices section of the Android Developers website at `http://developer.android.com/tools/device.html`.

Note:

You might experience issues when using USB connectivity for the device-based debugging. See Testing and Debugging MAF Applications in *Developing Mobile Applications with Oracle Mobile Application Framework*.

Your target Android-powered device might not be listed in the `.inf` file of the USB device driver, resulting in the failure to install the Android Debug Bridge (ADB). You can eliminate this issue as follows:

1. Find the correct values for your device.
2. Update the `[Google.NXx86]` and `[Google.NTamd64]` sections of the `android_winusb.inf` file.

See Install the Google USB Driver at `http://developer.android.com/sdk/win-usb.html`.

2.5.3 How to Set Up an Android Emulator

Follow the instructions on the Android Developers website to create the Android Virtual Device and review the configuration of the settings.

In your MAF application development and deployment, you can use either the Android device itself (see How to Set Up an Android-Powered Device) or its emulator. Deploying to an emulator is usually much faster than deploying to a device, and it also means that you do not have to sign the application first.

For information on how to create an emulator configuration called Android Virtual Device (AVD), follow the instructions from the Managing Virtual Devices section of the Android Developers website at `http://developer.android.com/tools/devices/index.html`. When creating an AVD through the Create New Android Virtual Device dialog (see Managing AVDs with AVD Manager at `http://developer.android.com/tools/devices/managing-avds.html`), review all the settings to ensure that configuration matches what you are planning to emulate. In particular, you should verify the following:

- The Target field should define the desired Android platform level for proper emulation.
- The CPU/ABI field should reflect the ARM or Intel Atom system image (see Configuring AVD for Intel HAXM).
- The SD card field should be defined based on whether the application uploads files or files install themselves to the SD card.
- The default settings for the Hardware field (see the Hardware Options table at `http://developer.android.com/tools/devices/managing-avds.html#hardwareopts`) must be acceptable for a typical MAF application. For the additional hardware capabilities you may want to use in your application, such as cameras or geolocation services, create new properties.

You need to create an AVD for each Android platform on which you are planning to test your application.

For information on how to use the emulator, see the Using the Android Emulator section in the Android Developers website at `http://developer.android.com/tools/devices/emulator.html`.

2.5.3.1 Configuring the Android Emulator

After the basic Android emulator setup is complete, you may choose to perform the following configurations:

- Save the emulator state (see Saving the Emulator State)
- Create, save, and reuse the SD card (see Creating, Saving, and Reusing the SD Card)
- Configure the network (see Configuring the Network)
- Configure the network proxy (see Configuring the Network Proxy)

2.5.3.1.1 Saving the Emulator State

You can reduce the emulator's load time by saving the emulator state or reusing the saved state. To do so, you manipulate the `avd` files or folders that are located in the `C:\Users\`*`username`*`\.android\avd` directory (on a Windows computer). Each `avd` folder contains several files, such as `userdata.img`, `userdata.qemu.img`, and `cache.img`. You can copy the `cache.img` file to another emulator's `avd` folder to use that state with another emulator.

Alternatively, you can use the command line to run relevant commands, such as, for example, `-snapshot-list`, `-no-snapstorage`, and so on. You can access these commands through `emulator -help` command.

Caution:

When using this utility, keep in mind that in the process of loading, all contents of the system, including the user data and SD card images, will be overwritten with the contents they held when the snapshot was made. Unless saved in a different snapshot, any changes will be lost.

2.5.3.1.2 Creating, Saving, and Reusing the SD Card

The "SD Card Emulation" section of the Android Developers website at `http://developer.android.com/tools/devices/emulator.html#sdcard` lists reasons for creating, saving, and reusing the SD card. You can perform these operations by executing the following commands:

- To create an SD card:

  ```
  C:\android sdk directory\tools>mksdcard -l SD500M 500M C:\Android\sd500m.img
  ```

- To list existing AVDs:

  ```
  C:\android sdk directory\tools>android list avd
  ```

  This produces a listing similar to the following:

```
Name:    AndroidEmulator1
Device:  Nexus S (Google)
Path:    C:\Users\username\.android\avd\AndroidEmulator1.avd
Target:  Android 4.2.2 (API level 17)
Tag/ABI: default/x86
Skin:    480x800
------------
Name:    AndroidEmulator2
Device:  Nexus S (Google)
Path:    C:\Users\username\.android\avd\AndroidEmulator2.avd
Target:  Android 4.2.2 (API level 17)
Tag/ABI: default/armeabi-v7a
Skin:    480x800
Sdcard:  500M
```

- To start the AndroidEmulator2 with the SD card that has just been created:

  ```
  C:\Android\android sdk directory\tools>emulator -avd AndroidEmulator2 -sdcard C:
  \Android\sd500m.img
  ```

- To list the running Android emulator instances:

  ```
  C:\Android\android sdk directory\platform-tools>adb devices
  ```

- To copy a test image to the SD card (this requires the emulator to restart):

  ```
  C:\Android\sdk\platform-tools>adb push test.png sdcard/Pictures
  85 KB/s (1494 bytes in 0.017s)
  ```

For more information, see the Android Tools Help at http://developer.android.com/tools/help/index.html.

2.5.3.1.3 Configuring the Network

From the Android emulator, you can access your host computer through the 10.0.2.2 IP. To connect to the emulator from the host computer, you have to execute the `adb` command from a command line on your development computer or from a script to set up the port forwarding.

To forward socket connections, execute

`adb forward local remote`

using the following forward specifications:

- `tcp:port`
- `localabstract:unix domain socket name`
- `localreserved:unix domain socket name`
- `localfilesystem:unix domain socket name`
- `dev:character device name`
- `jdwp:process pid` (remote only)

For example, an arbitrary client can request connection to a server running on the emulator at port 55000 as follows:

```
adb -e forward tcp:8555 tcp:55000
```

In this example, from the host computer, the client would connect to `localhost:8555` and communicate through that socket.

For more information, see the "Android Debug Bridge" section in the Android Developers website at http://developer.android.com/tools/help/adb.html.

2.5.3.1.4 Configuring the Network Proxy

If your development computer is behind a corporate firewall, you might need to configure a proxy by using one of the following techniques:

1. Execute this command to start the emulator and initiate its connection with the browser:

   ```
   emulator -avd myavd -http-proxy myproxy
   ```

2. Start the emulator and then use its Settings utility as follows:

   a. Click **Settings**, and **More**.

   b. Click **Cellular networks**, and then **Access Point Names**.

   c. Select an access point and set the proxy, port, username, and password.

   Note: Turn Airplane mode off and on to ensure that the changes are applied.

2.5.3.2 Speeding Up the Android Emulator

Use the Android SDK Manager or an Intel location to download Intel HAXM that uses Intel drivers to accelerate the functioning of an Android-powered device emulator.

The Intel Hardware Accelerated Execution Manager (Intel HAXM) is designed to accelerate the Android-powered device emulator by making use of Intel drivers.

The Intel HAXM is available for computers running Microsoft Windows, Mac OS X, and a separate kernel-based virtual machine option (KRM) for Linux. See http://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager to access installation guides and detailed descriptions of system requirements for each operating system.

Regardless of which operating system your development computer is running on, it must have the following:

- The Android SDK installed (see How to Install the Android SDK).
- Intel processor with support for Intel VT-x, EM64T and Execute Disable (XD) Bit functionality at the BIOS level.
- At least 1 GB of available RAM.

To download the Intel HAXM, either use the Android SDK Manager (see *Speeding Up the Android Emulator on Intel Architecture*) or use the following Intel locations:

- Download for Microsoft Windows
- Download for Mac OS X
- Download for Linux

To install the Intel HAXM, follow the steps described in the Speeding Up the Android Emulator on Intel Architecture available at http://software.intel.com/en-us/android/articles/speeding-up-the-android-emulator-on-intel-

`architecture`. It is particularly important to configure AVD (see Configuring AVD for Intel HAXM).

If your development computer is running either Microsoft Windows 8.*n* or later, or Mac OS X 10.9.*n* or later, you have to apply a Hotfix provided by Intel before using the emulator with the Intel HAXM.

Note:

If you do not apply the Hotfix, your computer will freeze and you will lose your work.

To download the Hotfix, use the following locations:

- Download for Microsoft Windows
- Download for Mac OS X

See also:

- *Installation Guide and System Requirements - Windows*
- *Installation Guide and System Requirements - Mac OS X*
- *Installation Guide and System Requirements - Linux*

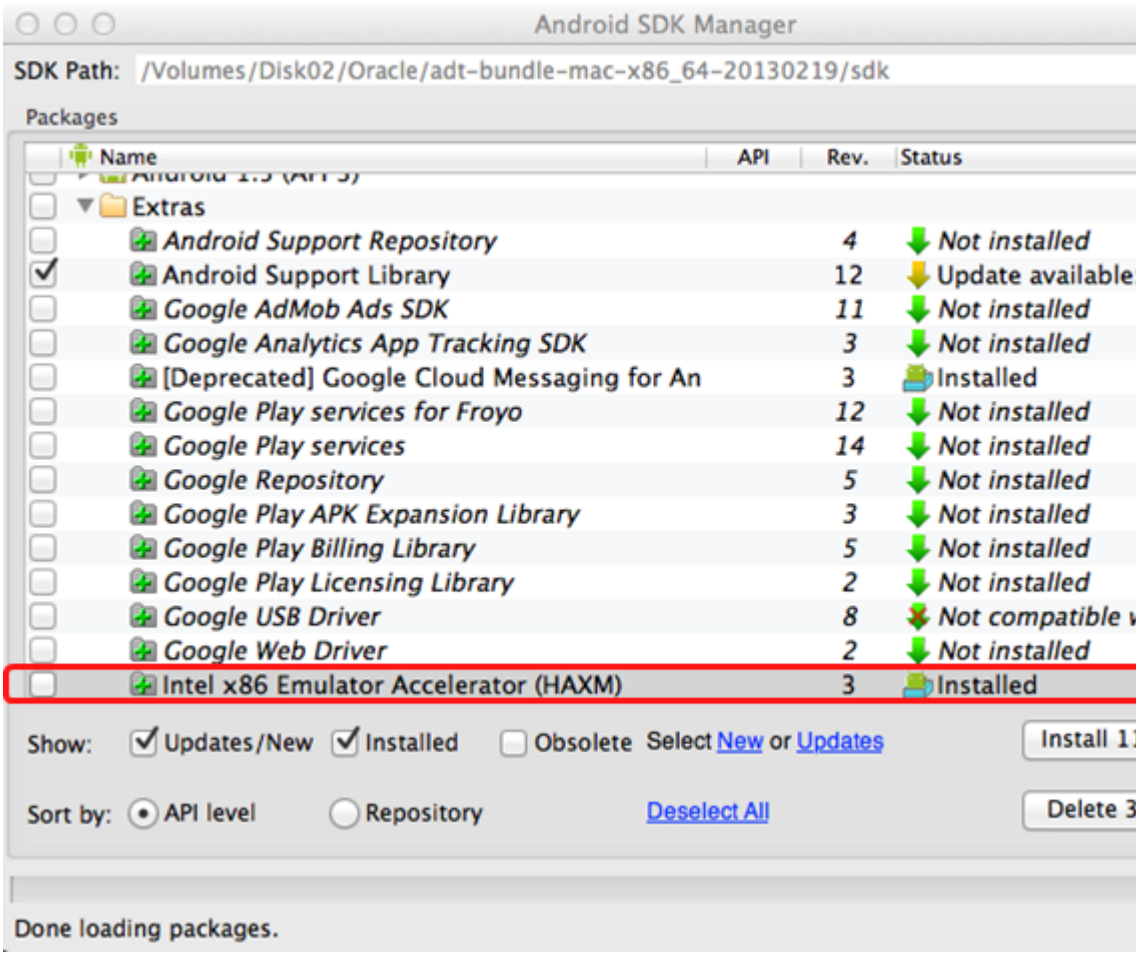
2.5.3.2.1 Configuring AVD for Intel HAXM

When enabling the Intel HAXM, ensure that you download the Intel system image for the Android API level using the Android SDK Manager (see Figure 2-4). The following steps described in *Speeding Up the Android Emulator on Intel Architecture* guide you through the configuration process:

- After you have installed the Android SDK, open the SDK Manager and then find the Intel HAXM in the extras section.
- Select **Intel x86 Emulator Accelerator (HAXM)** and click **Install packages**.

  Once you have installed the package, the status changes to Installed, which is not accurate: the SDK only copies the Intel HAXM executable on your computer; you have to manually install the executable.

Figure 2-4 Downloading Intel System Image in Android SDK Manager

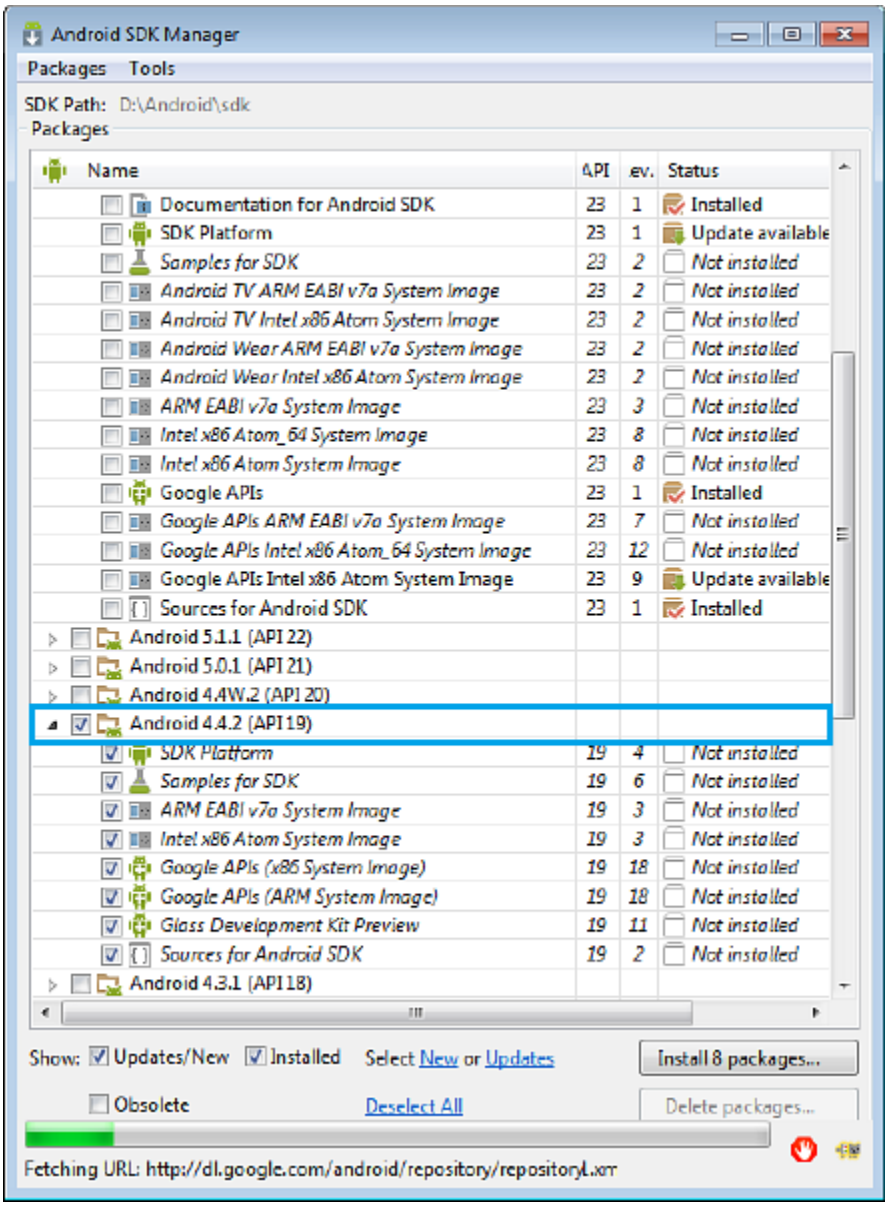


- To install the Intel HAXM executable, depending on your development platform search your hard drive for one of the following:
  - On Windows, search for `IntelHaxm.exe`
  - On Mac OS X, search for `IntelHaxm.dmg`

  If you accepted default settings, the executable should be located at `C:\Program Files\Android\android-sdk\extras\Intel\Hardware_Accelerated_Execution_Manager\IntelHaxm.exe` on Windows.

The Intel HAXM only functions in combination with one of the Intel Atom processor x86 system images, which are available for Android 2.3.3 (API 10), 4.0.3 (API 15), 4.1.2 (API 16), 4.2.2 (API 17), 4.4 (API 19), 4.4W (API 20), 5.0 (API 21). These system images can be installed exactly like the ARM-based images through the Android SDK Manager.

Figure 2-5 Installing Intel Atom System Image

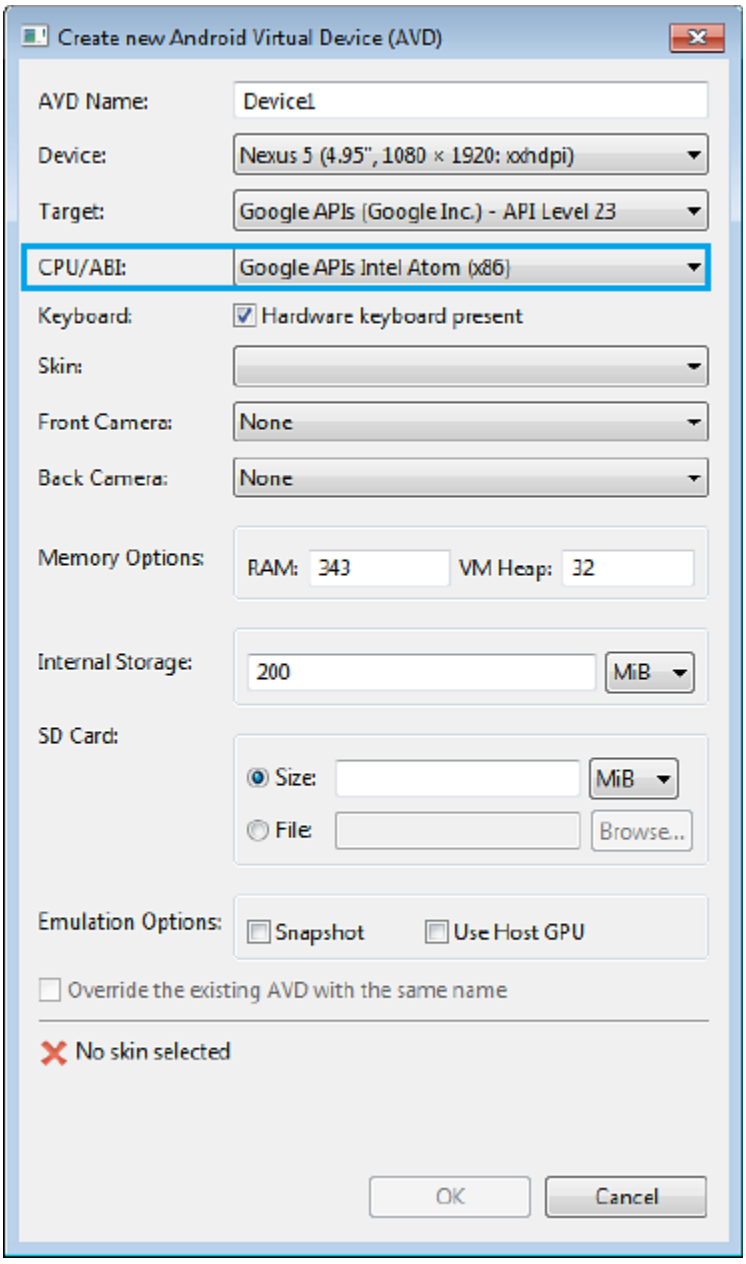


To complete the process, use the AVD Manager to create a new virtual device that has hardware-accelerated emulation by selecting **Intel Atom (x86)** as the CPU/ABI, (see Figure 2-6).

Note:

This option appears in the list only if you have the Intel x86 system image installed.

Figure 2-6 Creating Accelerated AVD



2.6 Setting Up Development Tools for the Universal Windows Platform

Ensure that the development computer meets the installation requirements, install the MAF extension and Visual Studio 2015, create and install the PFX file, and enable Development Mode on the computer to set up a Windows 10 computer for application development.

To set up your development machine so that you can develop and deploy a MAF application to UWP:

- Verify that your computer meets the requirements listed in Installation Requirements for MAF Applications to be Deployed to the Universal Windows Platform.

- Install the MAF extension following the steps in Installing the MAF Extension in JDeveloper.
- Install Visual Studio from Microsoft. The Visual Studio download provides the Windows SDK that enables deployment of applications to the UWP. 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.
- Create and install the PFX file as described in Creating a PFX File for MAF Applications and Installing a PFX File on a Windows 10 Device.
- Enable Development Mode on the computer on which you intend to develop the application as described in Enabling Development Mode in a Windows 10 Device.

After completing these setup tasks, a MAF application can be deployed to UWP. See Deploying a MAF Application to the Universal Windows Platform in *Developing Mobile Applications with Oracle Mobile Application Framework*.

2.6.1 Installing Visual Studio

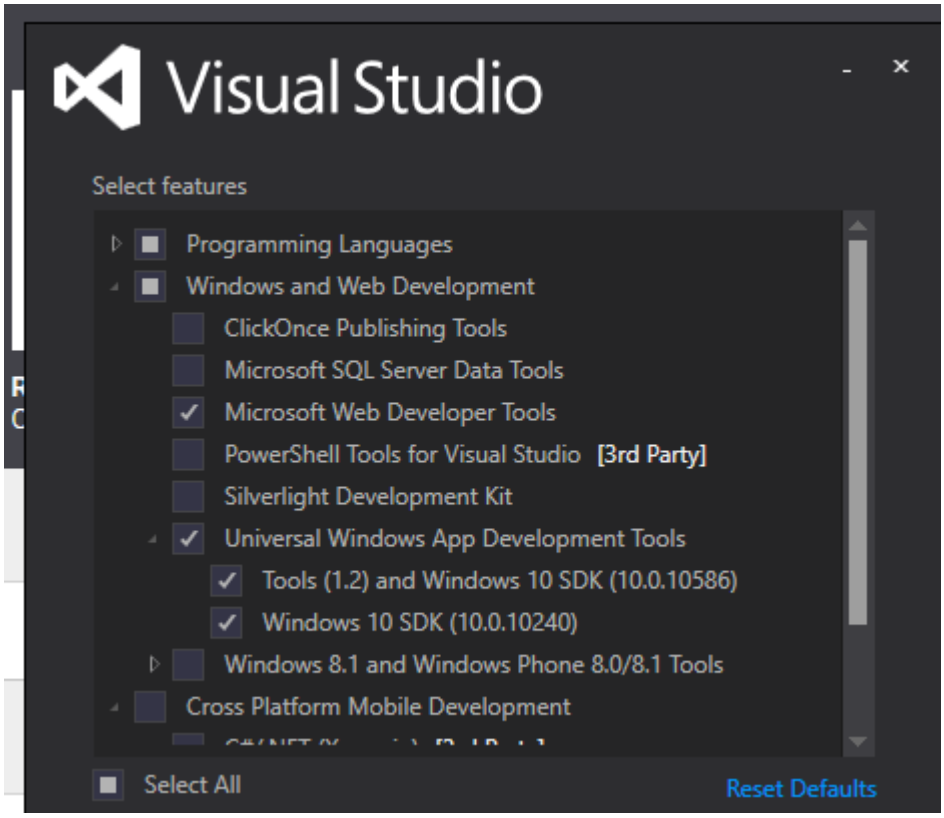
Install Visual Studio 2015, which includes the Windows 10 SDK.

1. Download and install an edition of Visual Studio 2015 available at: https://www.visualstudio.com/products/vs-2015-product-editions. The Visual Studio download includes the Windows 10 SDK.

   Note: The Visual Studio 2015 package includes the Windows 10 SDK.

2. During the Visual Studio 2015 installation, make sure that you select **Universal Windows App Development Tools** and **Windows 10 SDK**, as shown in Figure 2-7.

Figure 2-7 Installing Visual Studio



For the following information, see Windows Software Development Kit (SDK) for Windows 10 at

https://dev.windows.com/en-us/downloads/windows-10-sdk

- What's in the kit
- New APIs
- New and updated tools
- System requirements
- Instructions to install and uninstall
- Known issues

2.6.2 Creating a PFX File for MAF Applications

Follow the steps in the task to create a Personal Information Exchange (PFX) file that is needed to digitally sign Universal Windows Platform (UWP) based MAF applications.

MAF applications based on UWP must be digitally signed before deployment. A PFX file is required to sign an application. A PFX file contains a public x509 certificate file (.cer) and a private key file (.pvk file).

To create a PFX file:

1. Open a Command Prompt window as Administrator.

2. Navigate to the location: C:\Program Files (x86)\Windows Kits\10\bin\x64
3. Run the following command, with variables you want, to create a Windows proprietary private key file (.pvk) and a X.509 certificate file (.cer):

```
makecert.exe -sv c:\somedir\MyKey.pvk -n "CN=Your Name,OU=MAF,O=Oracle,C=US" -r -
h 0 -eku "1.3.6.1.5.5.7.3.3,1.3.6.1.4.1.311.10.3.13" c:\somedir\MyKey.cer
```

   - The "-eku" (Enhanced Key Usage) flag value must not have spaces between the two comma delimited values. The 1.3.6.1.5.5.7.3.3 OID indicates that the certificate is valid for code signing. The 1.3.6.1.4.1.311.10.3.13 indicates that the certificate respects lifetime signing.
   - The "-r" flag creates a self-signed root certificate. This simplifies management of your test certificate.
   - The "-h 0" flag marks the basic constraint for the certificate as an end-entity. This constraint prevents the certificate from being used as a Certification Authority (CA) that can issue other certificates.

4. In the Create Private Key Password window, enter a password and confirm it.
5. In the Enter Private Key Password window that opens, enter the password that was created.

   Verify whether a .pvk and .cer file were created at the specified locations.

6. Run the following command to convert the certificate and the private key files into a PFX file that can be used by Visual Studio.

```
pvk2pfx.exe -pvk c:\somedir\MyKey.pvk -spc c:\someDir\MyKey.cer -pfx c:\someDir
\MyPFX.pfx -pi welcome -po welcome
```

   - -pi : Specify this flag or value if you entered a password for the pvk file that you created. If the pvk file is password protected, and you do not specify the flag, pvk2pfx.exe will prompt you for the password.
   - -po : Specify this flag or value if you want to password-protect the .pfx file being created.

   Verify whether a PFX file was created.

2.6.3 Installing a PFX File on Windows 10

Follow the steps in the task to install a Personal Information Exchange (PFX) file in a certificate store on a computer so that the certificate can be used for application signing.

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

> **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.

2.6.4 Enabling Developer Mode on Windows 10

Follow the steps in the task to 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. In JDeveloper, on the **Tools** menu, click **Preferences**, and from the preferences that are displayed, expand **Mobile Application Framework**, and then click **Windows Platform**.
4. In Mobile Application Framework: Windows Platform, specify the location of the Windows SDK location.
5. In the Certificate Location and Password fields, enter the location of the PFX file, and the password.

Note: 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.

2.7 Testing the Environment Setup

Deploy a MAF sample application to test that you set up your 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*.

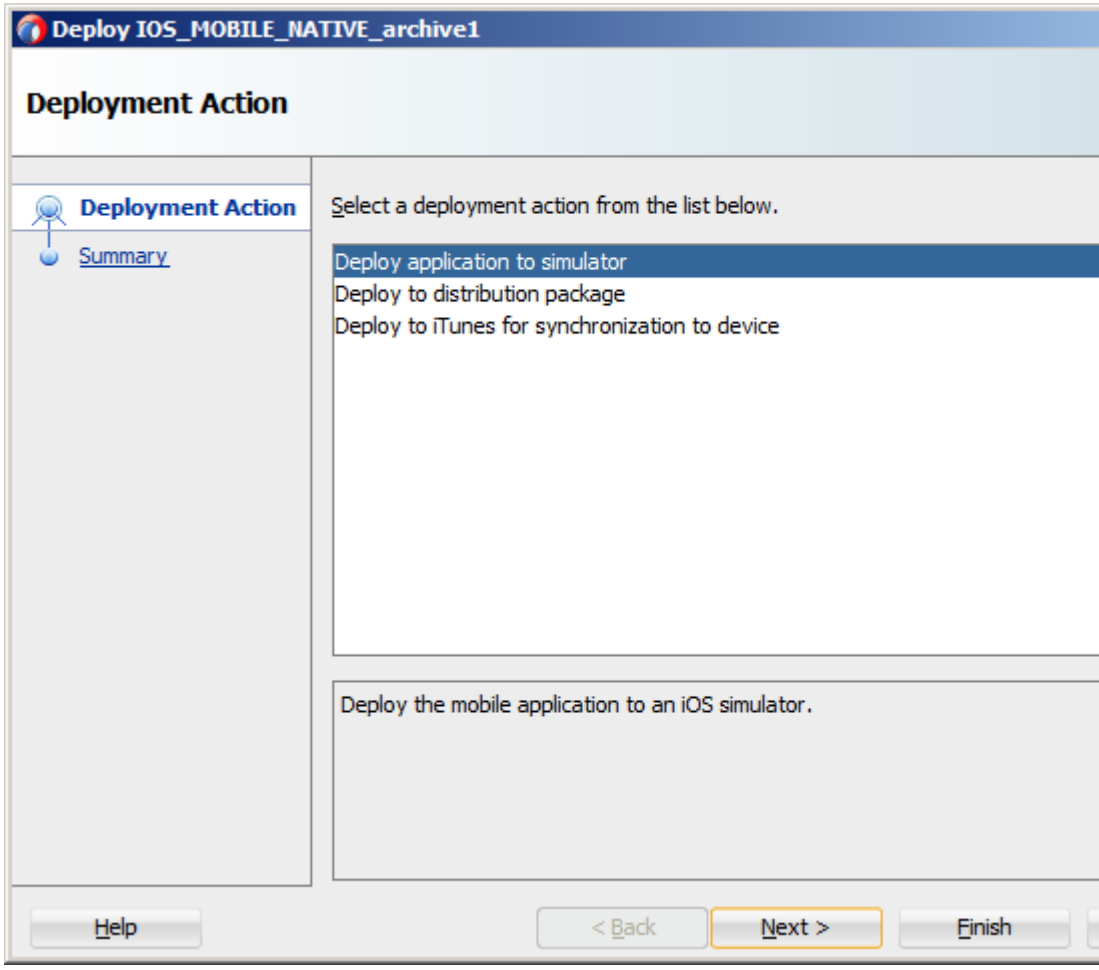
2. Select **Application**, and then **Deploy** from the main menu.

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

3. From the dropdown menu, select the deployment profile for the platform to which you wish to deploy the application.
4. Since using an iOS-powered device simulator or Android-powered device emulator to test the environment setup is preferable because it does not require signing of the application, you should select one of the following deployment actions using the **Deploy** dialog:

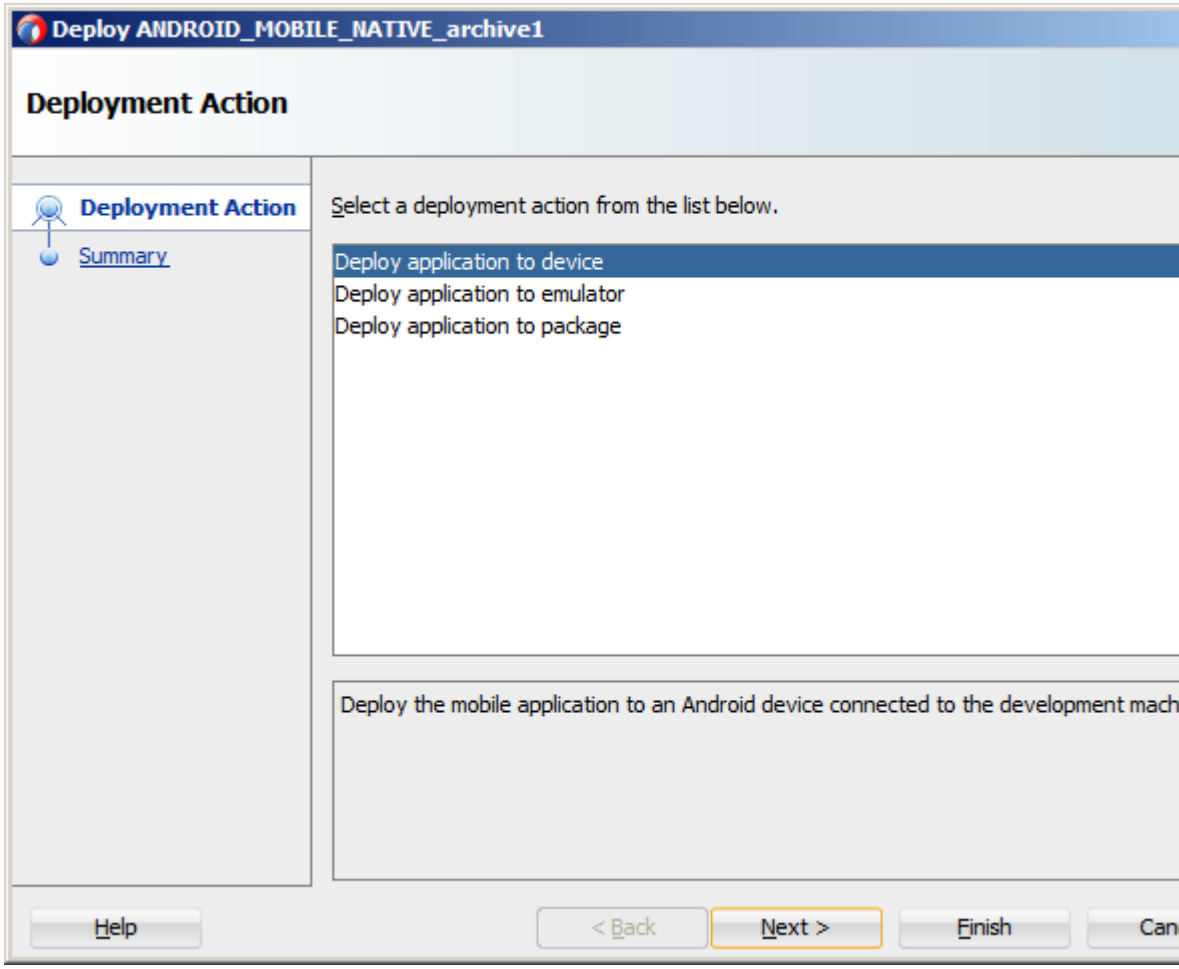
   - For iOS, select **Deploy application to simulator**, as Figure 2-8 shows.

Figure 2-8 Selecting Deployment Action for iOS



- For Android, select **Deploy application to emulator**, as Figure 2-9 shows. Ensure that the emulator is running before you start the deployment.

Figure 2-9 Selecting Deployment Action for Android



- For the Windows platform, use the following steps:

  a. Click **Application**, **Deploy**, and then **Windows1**.

  b. In Deployment Action, select the deployment action to deploy the mobile application to a local Windows machine.

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.

2.8 Setting Up Development Tools from the Command Line Using Startup Parameters

You can set MAF preferences required to develop MAF applications, such as the Android SDK location, by specifying startup parameters when you start JDeveloper. Startup parameters exist to specify the MAF preferences to develop and deploy MAF applications on the Android and iOS platforms.

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.adfmf.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.androidPlatformDir`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.androidBuildToolsDir`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.androidReleaseSigningKeystorePath`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.androidReleaseSigningKeystorePath`

The following example shows how to override the location of the iTunes Media folder:

```
 ./jdev -J-Doracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.iOSiTunesDir="/
Users/<my_username>/Music/iTunes/iTunes Media/Automatically Add to iTunes.localized"
```

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.iosProvisioningProfileTeamName`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.iOSiTunesDir`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.iosCertificate`
- `oracle.adfmf.framework.dt.preferences.PlatformSDKsPrefs.iosProvisioningProfile`

3

Migrating Your Application to MAF 2.4.0

This chapter provides information that you may need to know if you migrate an application created using an earlier release of MAF to MAF 2.4.0.

This chapter includes the following sections:

3.1 Migrating an Application to MAF 2.4.0

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.

MAF now uses 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*.

This release updates the Cordova engine versions that MAF uses (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 2.4.0.

This release of MAF removes 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 more information about the APIs that MAF supports, see the *Java API Reference for Oracle Mobile Application Framework*.

This release also defaults 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.

For information about new features introduced in this release, see What's New in This Guide for MAF Release 2.4.0 in *Developing Mobile Applications with Oracle Mobile Application Framework*.

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

- MAF 2.3.3 and later requires 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. For more information, see Using Xcode 8 and Deploying to iOS 10 with MAF 2.4.0.
- MAF 2.3.2 and later:
  - Replaced the `java.security` file in your migrated MAF application with a new version generated by MAF. MAF saves the original file with the following filename: `java.security.orig`. If you had previously made changes to this file you may need to copy those changes to the new version of the `java.security` file.
  - You can configure migrated applications that run on iOS 9 to use WKWebView, as described in Configuring Application Features with AMX Content to Use WKWebView on iOS 9 and iOS 10.
- MAF 2.3.1 and later includes 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 2.4.0.
- MAF 2.3.0 and later use 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 2.4.0.
- The `RestServiceAdapter` interface has a new package location (`oracle.maf.api.dc.ws.rest`). The functionality that this interface specifies remains unchanged. For more information about creating a REST web service adapter, see Creating a Rest Service Adapter to Access Web Services in *Developing Mobile Applications with Oracle Mobile Application Framework*.
- MAF 2.3.0 removed support for the following features that were deprecated in earlier releases:
  - 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 Using Web Services in a MAF Application in *Developing Mobile Applications with Oracle Mobile Application Framework*.

- As of MAF 2.3.0, MAF no longer bundles the jQuery JavaScript library. 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.

- The MAF 2.3.0 release of MAF introduced support for the deployment of MAF applications to the Universal Windows Platform (UWP). 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 more information about deploying a MAF application to the UWP, see Deploying a MAF Application to the Universal Windows Platform in *Developing Mobile Applications with Oracle Mobile Application Framework*.

MAF enables 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.

The MAF 2.1.0 release introduced significant changes that are also described in this chapter. Use the information in this chapter if you migrate an application created in a pre-MAF 2.1.0 release to MAF 2.3.0.

MAF 2.1.0 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 `cacerts` file that contains new CA root certificates.

If you migrate an application to MAF 2.3.0 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 `cacerts` file.

Read the subsequent sections in this chapter that describe how these changes impact the migration of your MAF application to MAF 2.1.0 or later.

Finally, MAF 2.1.0 delivered 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 more information about how to connect to the SQLite database, see Using the Local SQLite Database in *Developing Mobile Applications with Oracle Mobile Application Framework*.

Close and re-open MAF applications in JDeveloper after you install the MAF extension that delivers this release of MAF. Do this to invoke the migrator that migrates your MAF application to the current release of MAF. After you migrate your MAF application to this release, invoke the JDeveloper Clean All command. This cleans your application of build artifacts from builds prior to migrating to this release. To do this, click **Build** > **Clean All** from the main menu in JDeveloper.

3.2 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.

3.3 Using Xcode 8 and Deploying to iOS 10 with MAF 2.4.0

MAF 2.3.3 and later requires Xcode 8 to develop and deploy MAF applications to the iOS platform.

Install or upgrade to Xcode 8.x, as described in How to Install Xcode and iOS SDK. Once you install or upgrade to Xcode 8.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 8.x replaces Xcode 7.x. No other changes are required, since JDeveloper will now use the active Xcode installation. If you want to maintain separate development environments for MAF 2.3.3 and later (using Xcode 8.x) and MAF 2.3.2 or earlier (using Xcode 7.x), you can install both Xcode 7.x and Xcode 8, as described below.

MAF has made the following additional changes to support use of Xcode 8 and deployment to iOS 10. Review this information and make appropriate modifications to your migrated MAF application to ensure a successful deployment:

- Exposed a new input field (**Team**) that displays the identifier of the development team. MAF automatically populates this input field with a value that it extracts from your provisioning profile. For more information, see Setting the Device Signing Options in *Developing Mobile Applications with Oracle Mobile Application Framework*

- The iOS options page of the MAF for iOS Deployment Profile Properties dialog now displays a **Push Notification Environment** dropdown list from where you must select `Production` or `Development` to register your deployed application with the Apple Push Notification service (APNs) if your deployed application supports push notifications. The default value is `Production`. Applications that you migrate to this release of MAF use the default value. For more information, see Defining the iOS Build Options and Enabling Push Notifications in *Developing Mobile Applications with Oracle Mobile Application Framework*.

- MAF applications deployed to iOS 10 require usage descriptions if the application uses device capabilities, such as the camera, that may access the end user's private

data. For more information, see Providing Usage Descriptions for Plugins that Access Device Capabilities on iOS in *Developing Mobile Applications with Oracle Mobile Application Framework*.

How To Maintain Separate Xcode 8.x and Xcode 7.x Installations

To maintain separate Xcode 8.x and Xcode 7.x installations:

1. Rename the preexisting Xcode.app installation for Xcode 7.x (For example, `Xcode7.app`.)
2. Install Xcode 8.x from the Apple App Store, as described in How to Install Xcode and iOS SDK. Make sure that you install, not update, Xcode from the Apple App Store.
3. Once you install Xcode 8.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 8.x installation:
   /Applications/Xcode.app

   Xcode 7.x installation:
   /Applications/Xcode7.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 8.x active:
   sudo xcode-select -s /Applications/Xcode.app

   //To make Xcode 7 active:
   sudo xcode-select -s /Applications/Xcode7.app

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

3.4 Migrating Cordova Plugins from Earlier Releases to MAF 2.4.0

MAF 2.4.0 and later use new versions of Cordova (4.x). 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 use the plugins it used previously, verify that this release of MAF supports the version of the plugin. The Cordova Engine Versions displays the versions that your release of MAF uses, as illustrated in Figure 3-1. 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. For more information, 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 3-1.

MAF applications developed using earlier releases of MAF (prior to MAF 2.1.0) registered plugins in the `maf-application` 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:

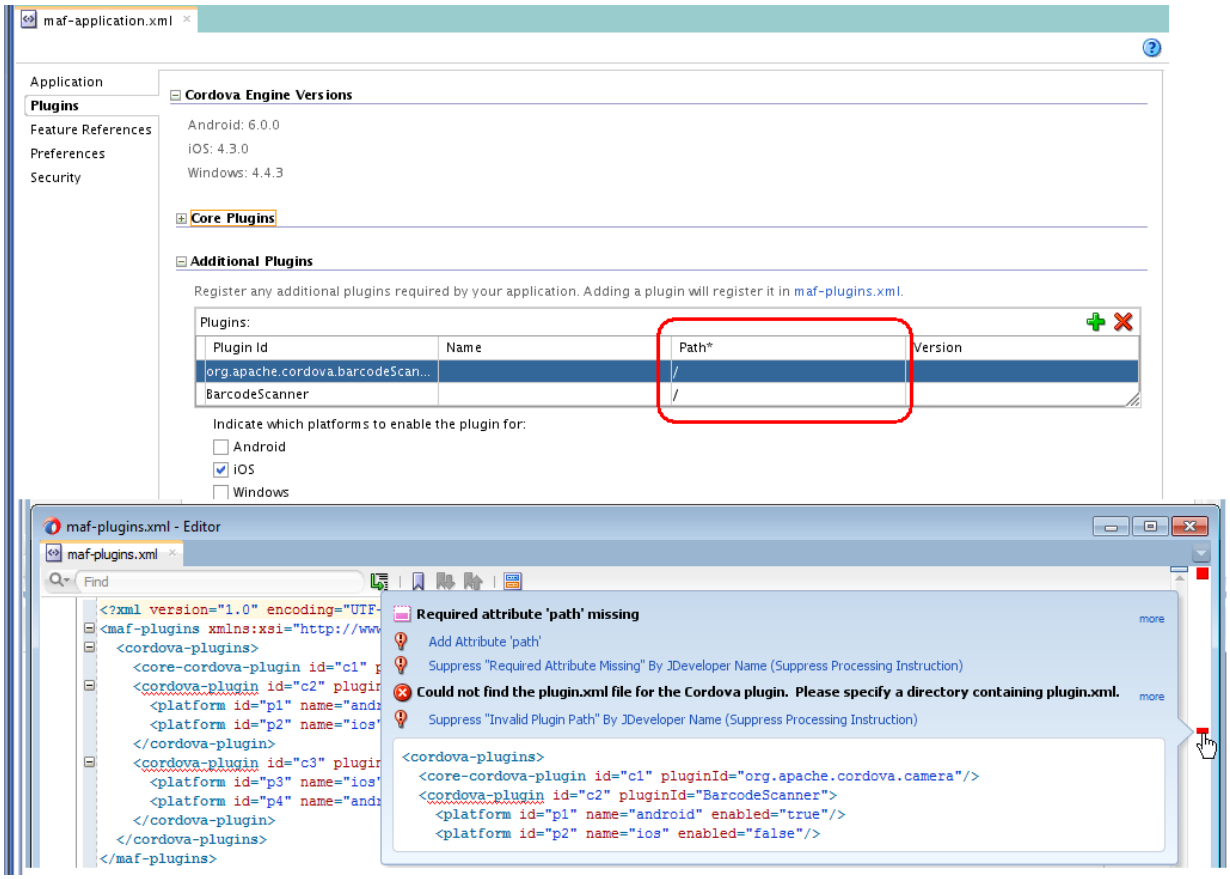
- Comments out entries in the `maf-application.xml` file that referenced plugins. For example, JDeveloper comments out entries such as the following:

```
<!--<adfmf:cordovaPlugins>
    <adfmf:plugin fullyQualifiedName="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 3-1 MAF Application that Does Not Specify Path to Plugin



3.5 Configuring Application Features with AMX Content to Use WKWebView on iOS 9 and iOS 10

New MAF applications that you create using the MAF 2.3.3 release and later of MAF use WKWebView by default to render AMX content type when you deploy the MAF application to an iOS 9 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="wkp1">
      <adfmf:property id="wkp1-1" 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.

3.6 Migrating an Application Developed Using AMPA to MAF 2.4.0

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:

```
<adfmf:application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:adfmf="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`.

3.7 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 9 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
```

3.8 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, as shown in Figure 3-2.

Figure 3-2 Registering a Custom URL Scheme that a MAF Applications Use to Invoke Another Application

3.9 Migrating to JDK 8 in MAF 2.4.0

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 in MAF 2.3.1 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 more 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
```

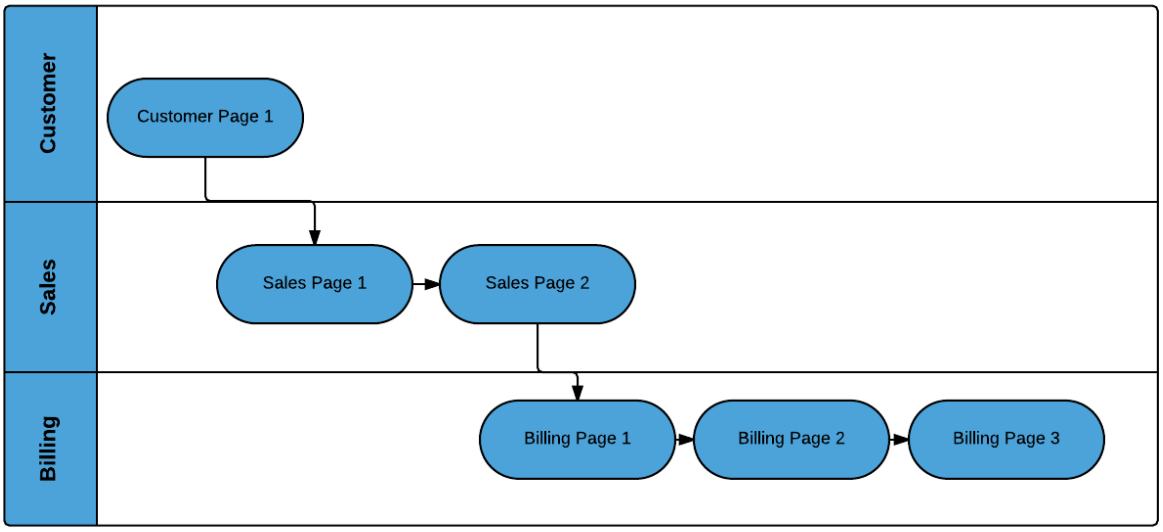
For more information about the `logging.properties` file, see How to Configure Logging Using the Properties File in *Developing Mobile Applications with Oracle Mobile Application Framework*.

3.10 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.

Figure 3-3 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 3-3 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 the "Navigating a MAF Application Using Android's Back Button" section of the *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.

3.10.1 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 3-1.

Example 3-1 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>
```

3.11 Migrating to New cacerts File for SSL in MAF 2.4.0

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 more information about using the `keytool` utility of JDK 8to 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 more 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 3-2 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 3-2 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
```