Oracle® Enterprise Pack

Installing Oracle Enterprise Pack

12*c* (12.2.1.6)

E83407-02

June 2017

Documentation that describes how to install the Oracle Enterprise Pack for Eclipse.

Oracle Enterprise Pack Installing Oracle Enterprise Pack, 12*c* (12.2.1.6)

E83407-02

Primary Author: Sujatha Joseph, Walter Egan

Contents

Preface

Welcome to *Installing Oracle Enterprise Pack*.

Audience

This document is intended for application developers who develop applications using Oracle Enterprise Pack for Eclipse.

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 the following documents:

- *Oracle Enterprise Pack for Eclipse Users Guide*
- *Oracle Enterprise Pack for Eclipse Online Help*
- *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in This Guide for This Release

Section	Change
Using Xcode 8 and Deploying to iOS 10 with MAF 2.3.3.	Added to describe how to configure MAF applications that you migrate to this release of MAF to deploy to the iOS platform that now requires Xcode 8.

1 Installing Oracle Enterprise Pack for Eclipse

This chapter provides information on installing Oracle Enterprise Pack for Eclipse.

It contains the following sections:

1.1 Installing Your OEPE Application

You can install OEPE by itself, or install OEPE with Eclipse.

OEPE is easily installed.

If you are installing Eclipse and OEPE, install using:

- The Eclipse Installer - installs Eclipse and OEPE from an installer launched from a Java plugin running in a browser. See Installing with the Eclipse Installer.
- The distro - download and unzip the kit which includes Eclipse along with OEPE. See Installing with the Distro.

If you already have Eclipse installed:

- You can install OEPE using Eclipse Marketplace. See Installing with Eclipse Marketplace.
- You can install from a repository. See Installing Using the Repository.

1.2 System Requirements

Understand the system requirements and components supported by OEPE.

See the System Requirements link under the **Technical Information** section of the following OEPE page on the Oracle Technology Network (OTN) at `http://www.oracle.com/technetwork/developer-tools/eclipse/learnmore/index.html`.

1.3 Installing with the Distro

Use the procedure to install Eclipse with OEPE from a zip archive.

The distro is a zip archive that contains Eclipse with OEPE already loaded. It is available from the Oracle Technology Network (OTN) web site at `http://www.oracle.com/technetwork/developer-tools/eclipse/downloads/index.html`.

To install using the distro:

1. From OTN, download the archive file appropriate for your operating system and Eclipse version.
2. Extract the archive file to a folder of your choice.

1.4 Installing with the Eclipse Installer

The Network Installer (Eclipse installer) installs Eclipse and OEPE. It also gives you control over the components that you install.

1.4.1 How to install using the Eclipse Installer

Install OEPE using the Eclipse Installer if you want the provision to select the versions of Eclipse and OEPE that suits your requirements.

From the Oracle Technology Network (OTN) web site at `http://www.oracle.com/technetwork/developer-tools/eclipse/downloads/index.html`, launch the Eclipse Installer. The installer downloads and opens.

You can choose the OEPE components to install and the level of guidance during the installation process. See Choosing What to Install.

To install using the installer:

1. On the Eclipse Location page of the installer, enter or browse to the location where you want to install OEPE.
2. On the Guidance Level of the installer, select the level of guidance you want.
3. If you select the **Explore available versions based on the required capabilities** guidance level, the next page is the Versions page. Move the capabilities you want to the Required Capabilities list, then select the Eclipse Version and OEPE version to install.
4. The Java Location page displays the detected JVM. If necessary, browse to another select another Java version.
5. On the Components Page, select the components that you want installed.
6. On the Licenses page of the installer, review the licence terms and accept them.
7. Click Install to download and install the versions of Eclipse and OEPE that you have specified.

1.4.2 Choosing What to Install

Eclipse provides levels of guidance to help you select the OEPE and Eclipse versions for installation.

The Eclipse Installer provides three levels of guidance to help you select the Eclipse and OEPE versions, and you should select the one that best suits your requirements. With each option, you can select the OEPE components to install.

1.4.2.1 Install Eclipse *version* and OEPE *version*

This is the simplest method of installing Eclipse and OEPE. The most recent version of Eclipse and the appropriate OEPE version are selected.

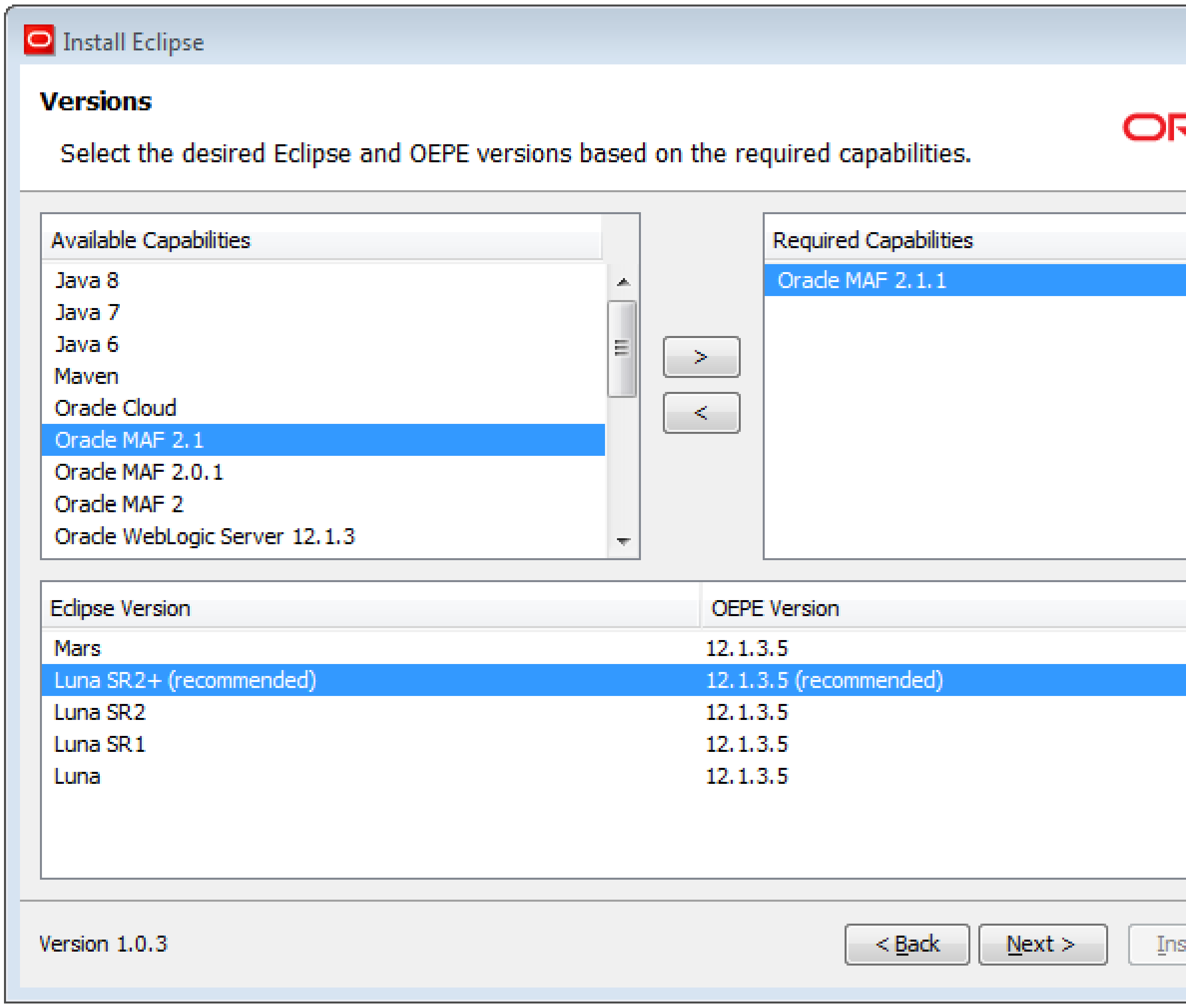
1.4.2.2 Choose an OEPE Version Based on an Eclipse Version

This option allows you to choose the Eclipse version you want, and then choose from the available OEPE versions for that Eclipse version.

1.4.2.3 Explore Available Versions Based on the Required Capabilities

This option allows you specify the capabilities you want. The installer displays the Eclipse and OEPE versions that provide this, as shown in Figure 1-1.

Figure 1-1 Choosing the Eclipse and OEPE Versions



Install Eclipse

Versions

Select the desired Eclipse and OEPE versions based on the required capabilities.

Available Capabilities
Java 8
Java 7
Java 6
Maven
Oracle Cloud
Oracle MAF 2.1
Oracle MAF 2.0.1
Oracle MAF 2
Oracle WebLogic Server 12.1.3

Required Capabilities
Oracle MAF 2.1.1

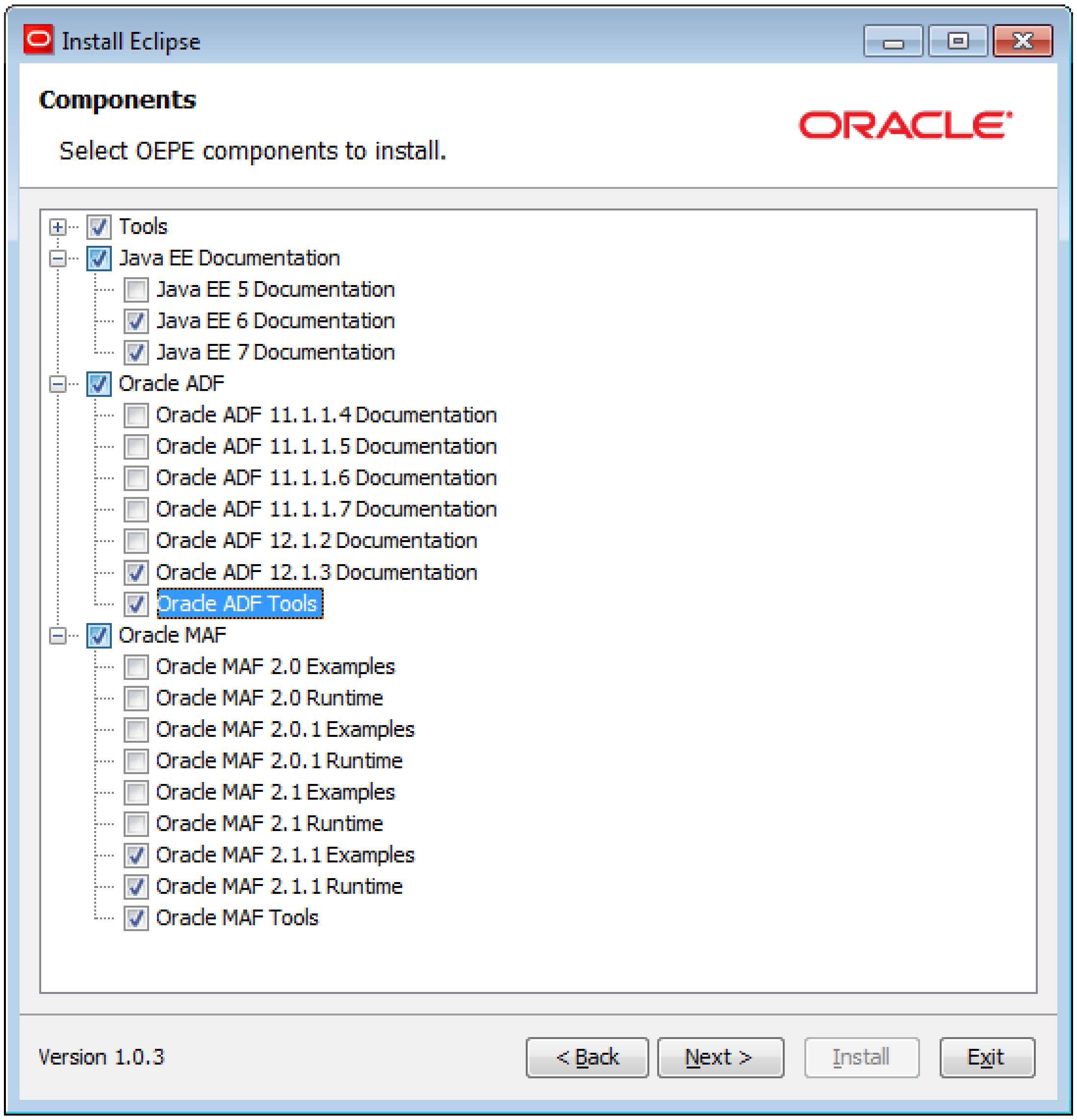
Eclipse Version	OEPE Version
Mars	12.1.3.5
Luna SR2+ (recommended)	12.1.3.5 (recommended)
Luna SR2	12.1.3.5
Luna SR1	12.1.3.5
Luna	12.1.3.5

Version 1.0.3

1.4.3 Selecting the Components to Install

Once you have determined which Eclipse version and OEPE version you want, you can choose the specific components to install, as shown in Figure 1-2.

Figure 1-2 Choosing the OEPE Components


Install Eclipse

Components

Select OEPE components to install.

- [✓] Tools
- [✓] Java EE Documentation
  - [ ] Java EE 5 Documentation
  - [✓] Java EE 6 Documentation
  - [✓] Java EE 7 Documentation
- [✓] Oracle ADF
  - [ ] Oracle ADF 11.1.1.4 Documentation
  - [ ] Oracle ADF 11.1.1.5 Documentation
  - [ ] Oracle ADF 11.1.1.6 Documentation
  - [ ] Oracle ADF 11.1.1.7 Documentation
  - [ ] Oracle ADF 12.1.2 Documentation
  - [✓] Oracle ADF 12.1.3 Documentation
  - [✓] Oracle ADF Tools
- [✓] Oracle MAF
  - [ ] Oracle MAF 2.0 Examples
  - [ ] Oracle MAF 2.0 Runtime
  - [ ] Oracle MAF 2.0.1 Examples
  - [ ] Oracle MAF 2.0.1 Runtime
  - [ ] Oracle MAF 2.1 Examples
  - [ ] Oracle MAF 2.1 Runtime
  - [✓] Oracle MAF 2.1.1 Examples
  - [✓] Oracle MAF 2.1.1 Runtime
  - [✓] Oracle MAF Tools

Version 1.0.3

1.5 Installing with Eclipse Marketplace

Use the procedure to install the latest version of OEPE for a particular version of Eclipse using Eclipse Marketplace.

If you already have an installation of Eclipse, you can install OEPE using Eclipse Marketplace.

Note:

The Eclipse Marketplace only allows you to install the latest OEPE version for a given Eclipse platform. If you want to install a specific version, install from the release repository.

To install using Eclipse Marketplace:

1. In Eclipse, select **Help**, and then **Eclipse Marketplace**.

2. In the Eclipse Marketplace dialog, search for `Oracle`. The Marketplace returns all Oracle entries, such as OEPE, and Oracle Cloud Tools. You can select to install OEPE, which includes all the features, or just the specific components you want.
3. Select the appropriate version of OEPE for the version of Eclipse you are using. For example, if you are using Eclipse Luna, select `Oracle Enterprise Pack for Eclipse Luna`. Click **Install**.
4. On the next page, the features that will be installed are listed. Click **Confirm**.
5. On the Review Licenses page, accept the terms of the license agreements, and click **Finish**.

   The software is installed. You can select to run the installation in the background by clicking **Run in Background**.

1.6 Installing Using the Repository

You can install OEPE from within Eclipse. Use the procedure to install OEPE thus from a repository in Oracle Technology Network.

Use the Install New Software feature from the repository available from the Oracle Technology Network (OTN) web site at `http://www.oracle.com/technetwork/developer-tools/eclipse/downloads/index.html`.

To install using the repository:

1. From the Eclipse main menu, select **Help**, and then **Install New Software**.
2. Click **Add** to add a new update site.
3. In the Add Repository dialog, enter the repository location which you can find under the **Plugin Repository** section on OTN.

   Then click **OK**.

   Note: This URL works only from within Eclipse, and will not work if accessed through a browser.
4. In the software list, select **Oracle Enterprise Pack for Eclipse**, select the subcomponents you want, and then click **Next**.
5. Confirm information presented on the Install Details, and then click **Next**.
6. Review licenses on the Review Licenses page and click **Finish**.

1.7 Updating an Existing Installation

Eclipse allows you to browse for updates and install them, or to uninstall features that are already installed. You control this functionality from the Install/Update page of the Preferences dialog (available from the Window menu).

1.7.1 How to Update Using Check for Updates Option

You can update an OEPE installation. Use the procedure to download new OEPE components.

You can use the Update wizard to download new components.

Updating OEPE Using Check for Updates:

1. In Eclipse, select **Help**, and then **Check for Updates**. This launches an Update wizard.
2. In the wizard, select the appropriate options.
3. Agree to any licences, and start the download.

1.7.2 Troubleshooting Update

An OEPE update fails if you use non-Oracle sites that make available different versions of plugins. Use the procedure to resolve such an update failure.

An OEPE update can fail if conflicting versions of plugins are found on non-Oracle sites. If update fails, de-select all non-Oracle repositories and try again.

To select the appropriate site:

1. Select **Window**, and then **Preferences**, then expand the category **Install/Update**.
2. Select **Available Software Sites**.
3. In the list of Available Software Sites, select just the appropriate Oracle repository, then click **OK**.

Now try using update again.

2

Configuring Mobile Application Framework

This chapter provides information on setting up and configuring the Mobile Application Framework (MAF) environment for application development and deployment.

This chapter includes the following sections:

2.1 Introduction to the MAF Environment

Prepare the environment for application development by setting up Eclipse, platform-specific tools, and a mobile device.

Before developing a MAF application, you must set up your development environment by downloading, installing, and configuring various software components.

To set up a typical MAF development environment that consists of an IDE, mobile platform-specific tools, and, possibly, a mobile device, follow the steps described in Prerequisites for Developing MAF Applications.

For a complete list of supported versions of development and runtime tools, see Oracle Mobile Application Framework Certification Matrix by following the Certification Information link on the MAF documentation page at `http://www.oracle.com/technetwork/developer-tools/maf/documentation/`.

2.2 Prerequisites for Developing MAF Applications

To develop MAF applications for this version of MAF, you need to use the JDK8 JRE.

You need to specify JDK 8's JRE as the default JRE for OEPE, as described in How to Specify JDK 8's JRE as the Default in OEPE.

See the following sections for platform-specific prerequisites if you plan to target one or both of the following platforms:

- What You Need to Develop an Application for the iOS Platform

- What You Need to Develop an Application for the Android Platform
- What You Need to Develop an Application for the Universal Windows Platform

You do not need to install any additional tools for creating specific types of MAF application content (HTML, remote URL, or MAF AMX).

2.2.1 What You Need to Develop an Application for the iOS Platform

A Mac computer, OEPE, Xcode and iOS SDK, JDK 1.8, credentials, and an iOS-powered device are needed for to develop applications for the iOS platform.

Before you start creating a MAF application for iOS, ensure that you have the following available:

- A computer running Mac OS X
- OEPE (see Setting Up OEPE)
- Xcode and iOS SDK (see How to Install Xcode and iOS SDK)
- The most recent version of JDK 8

Before you start deploying your application to a development environment (see Getting Started with Mobile Application Development in *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*), decide whether you would like to use a mobile device or its simulator: if you are to use a simulator, see How to Set Up an iPhone or iPad Simulator; if your goal is to deploy to 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 Mobile Applications in *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*.
- iOS-powered device. See How to Set Up an iPhone or iPad.

2.2.2 What You Need to Develop an Application for the Android Platform

Besides a computer running an OS supported by MAF, you need OEPE, JDK 8, Android SDK with platform and build tools, credentials, and an Android-powered device to develop applications for the Android platform.

Before you start creating a MAF application for Android, ensure that you have the following available:

- A computer running an operating system listed in the Desktop OS Requirements section of the Oracle Mobile Application Framework Certification Matrix that you access by following the Certification Information link on the MAF documentation page at `http://www.oracle.com/technetwork/developer-tools/maf/documentation/`.
- The most recent version of JDK 8
- Android SDK with platform and build tools (see Setting Up Development Tools for the Android Platform)
- OEPE (see Installing Oracle Enterprise Pack for Eclipse .)

2.2.3 What You Need to Develop an Application for the Universal Windows Platform

A computer with x86 architecture running the Windows 10 operating system, JDK 8, Microsoft Visual Studio 2015, MSBuild 14.0, and OEPE are required to develop applications for Universal Windows Platform.

Before you start creating a MAF application for Universal Windows Platform, ensure that you have the following available:

- A computer with x86 architecture running the Windows 10 operating system
- The most recent version of JDK 8
- 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`
- OEPE (see Installing Oracle Enterprise Pack for Eclipse .)

For more information about setting up and configuring your development environment, see Setting Up Development Tools for the Universal Windows Platform.

2.3 Setting Up OEPE

Learn how to specify the JRE and configure OEPE to target the Mobile application platform.

Setting up OEPE to develop MAF applications requires you to:

- Specify the JRE from JDK 8, as described in How to Specify JDK 8's JRE as the Default in OEPE.
- Configure OEPE to target the platform(s) you want the MAF application to run on, as described in How to Configure the Development Environment for Target Platforms.

2.3.1 How to Specify JDK 8's JRE as the Default in OEPE

OEPE requires that JDK 8 be specified as the default JRE for application development. Use the procedure to specify the JRE of JDK 8 as the default in OEPE.

You must specify the JRE packaged with JDK 8 as the default JRE in OEPE so that OEPE adds this JRE to the build path of newly-created MAF applications.

To specify JRE of JDK 8 as the default in OEPE:

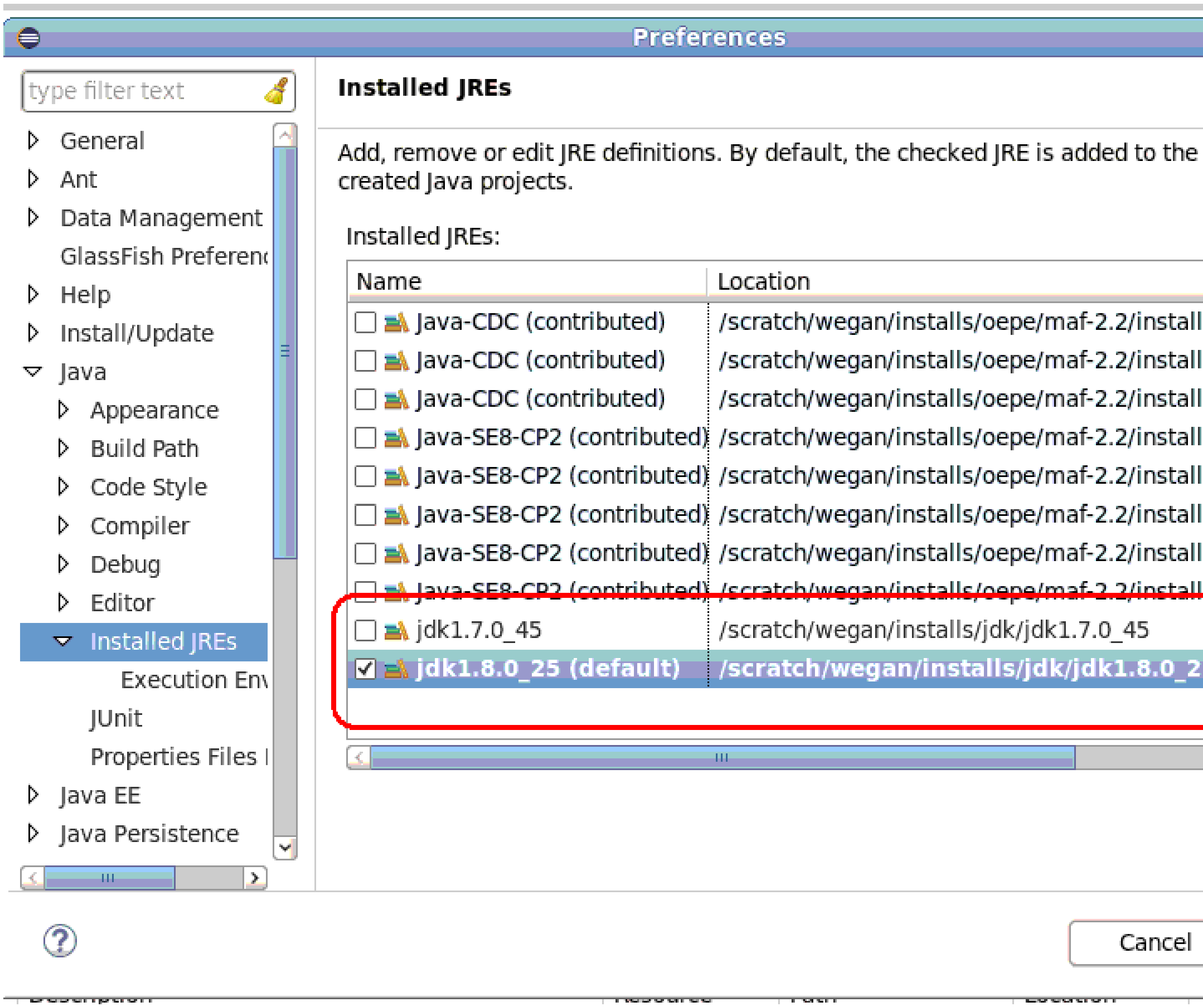
1. Select **Window**, and **Preferences** from the OEPE main menu to open the Preferences dialog.
2. In the **Preferences** dialog, open the folder **Java > Installed JREs** from the tree to open a page that displays a list of installed JREs.
3. If the JDK 8 JRE (for example, `jdk1.8.0_25`) is present, select the associated checkbox to make it the default.

4. If the JDK 8 JRE is not present, do the following:

   - Click **Add**, select **Standard VM** in the dialog that appears and click **Next**.
   - In the Add JRE dialog that appears, click **Directory** to browse to your root JDK 8 directory and select it.
   - Click **Finish**.

5. In the Installed JREs page of the Preferences dialog, verify that the JDK 8 JRE is the default selection, as shown in Figure 2-1.

Figure 2-1 Default JRE Selection in OEPE Preferences



Installed JREs

Add, remove or edit JRE definitions. By default, the checked JRE is added to the created Java projects.

Installed JREs:

Name	Location
☐ Java-CDC (contributed)	/scratch/wegan/installs/oepe/maf-2.2/install
☐ Java-CDC (contributed)	/scratch/wegan/installs/oepe/maf-2.2/install
☐ Java-CDC (contributed)	/scratch/wegan/installs/oepe/maf-2.2/install
☐ Java-SE8-CP2 (contributed)	/scratch/wegan/installs/oepe/maf-2.2/install
☐ Java-SE8-CP2 (contributed)	/scratch/wegan/installs/oepe/maf-2.2/install
☐ Java-SE8-CP2 (contributed)	/scratch/wegan/installs/oepe/maf-2.2/install
☐ Java-SE8-CP2 (contributed)	/scratch/wegan/installs/oepe/maf-2.2/install
☐ Java-SE8-CP2 (contributed)	/scratch/wegan/installs/oepe/maf-2.2/install
☐ jdk1.7.0_45	/scratch/wegan/installs/jdk/jdk1.7.0_45
✓ **jdk1.8.0_25 (default)**	**/scratch/wegan/installs/jdk/jdk1.8.0_2**

2.3.2 How to Configure the Development Environment for Target Platforms

Before you start developing and deploying a MAF application, you may need to configure preferences for appropriate platforms.

2.3.2.1 Configuring the Environment for Target Platforms

OEPE must be provided with information to package and deploy an application to a target platform. Use the procedure to configure the environment for target platforms.

For successful packaging and deployment of your application to target platforms supported by MAF, OEPE must be provided with information such as the name of the platform and directories on your development computer that contain the platform-specific tools and data.

Before you begin:

Depending on your target platform, download and configure the appropriate software for your target platform:

- Android SDK (see Setting Up Development Tools for the Android Platform) or iOS SDK and Xcode (see How to Install Xcode and iOS SDK)
- Visual Studio (see How to Install Visual Studio)

To configure your environment for target platforms:

1. Select **Window > Preferences** from the OEPE main menu to open the Preferences dialog.
2. In the **Preferences** dialog, open the appropriate folder to access a page that contains the path and configuration parameters for the supported platforms:
    - **Oracle > Mobile Application Framework > Android**
    - **Oracle > Mobile Application Framework > iOS**
    - **Oracle > Mobile Application Framework > Windows**

    Each platform-specific page hosts the preferences for the platform SDK, collecting any necessary information such as the path that MAF needs to compile and deploy the projects:

    - For Android platform, specify the Android SDK location on your computer, the local directory of your target Android platform, and provide information on the signing credentials by selecting **Android Keystores** from the Android Platform section of the tree.
    - For iOS platform, specify the location of the `Xcodebuild` utility. See How to Deploy an iOS Application to an iOS Simulator.
    - For the Universal Windows Platform, specify the Windows SDK location on your computer. See How to Create a Deployment Configuration for Universal Windows Platform .

2.4 Setting Up Development Tools for the iOS Platform

Set up an iPhone, iPad, or simulators to which MAF applications can be deployed.

In addition to general-purpose tools listed in Prerequisites for Developing MAF Applications, you might want to set up an iPhone or iPad when getting ready for 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, you do not need to separately install them. See How to Set Up an iPhone or iPad Simulator.

2.4.1 How to Install Xcode and iOS SDK

Download Xcode, which includes the iOS SDK, and run it at least once to ensure that builds and deployments proceed smoothly.

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 performed, any build and deploy cycle from OEPE to Xcode or a device simulator fails with a "Return code 69" error.

Note:

Since older versions of Xcode 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 gain access to the Apple developer site at `http://developer.apple.com`.

2.4.2 How to Set Up an iPhone or iPad

An iOS-powered device with a valid license, certificates, and distribution profile must be connected to your computer for deployment of 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 are planning to use an actual iPhone or iPad, which is preferable for testing (see Testing MAF Applications in *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*), you need to connect it 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 (OEPE Edition)*.

Note:

Since 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 the external network access to use iOS simulators included in XCode downloads for the deployment of 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 OEPE to a device simulator, you must first run the simulator.

If you are planning to use web services in your application and you are behind a corporate firewall, you might need to configure the 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 (OEPE Edition)*.

2.5 Setting Up Development Tools for the Android Platform

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

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

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

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

Once you have installed Android Studio, perform the tasks described in the following topics:

- Install an Emulator Accelerator
- Create an Android Virtual Device
- Set Up Your Android Device to Install an App from Your Development Machine

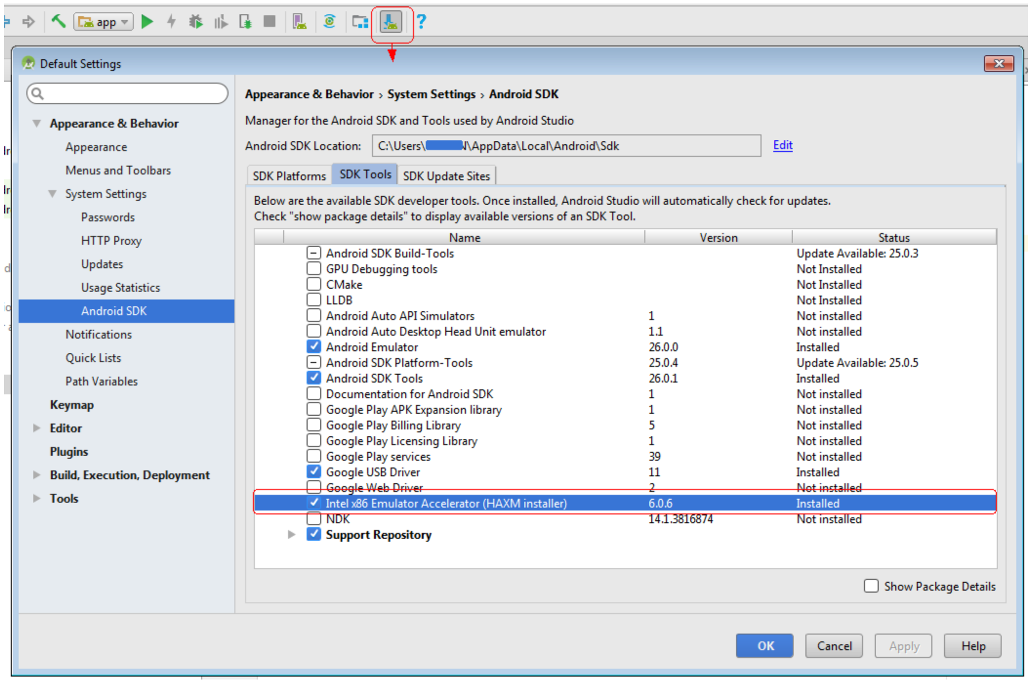
Tip: Android Studio prompts you with wizards to create AVDs, install emulator accelerators, or install the OEM USB driver required to connect to a real device when you attempt to run an app that you create in Android Studio. Consider completing the tasks described in Building Your First App to create a native Android app up to the point where it describes how to run the app on a real device or an emulator. Upon completion of this task, you will have an environment (an AVD on the emulator or a real device) where you can deploy an Android native app. A MAF application is packaged in the same type of installation file (.APK) as the Android native app, so once you have successfully completed this task, you know that you have an environment ready to install a MAF application.

2.5.1 Install an Emulator Accelerator

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

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

Figure 2-2 Install Emulator Accelerator



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: C:\Users\[redacted]\AppData\Local\Android\Sdk   Edit

SDK Platforms | SDK Tools | SDK Update Sites

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
Android SDK Build-Tools		Update Available: 25.0.3
GPU Debugging tools		Not Installed
CMake		Not Installed
LLDB		Not Installed
Android Auto API Simulators	1	Not installed
Android Auto Desktop Head Unit emulator	1.1	Not installed
✓ Android Emulator	26.0.0	Installed
Android SDK Platform-Tools	25.0.4	Update Available: 25.0.5
✓ Android SDK Tools	26.0.1	Installed
Documentation for Android SDK	1	Not installed
Google Play APK Expansion library	1	Not installed
Google Play Billing Library	5	Not installed
Google Play Licensing Library	1	Not installed
Google Play services	39	Not installed
✓ Google USB Driver	11	Installed
Google Web Driver	2	Not installed
✓ Intel x86 Emulator Accelerator (HAXM installer)	6.0.6	Installed
NDK	14.1.3816874	Not installed
✓ **Support Repository**		

Show Package Details

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

2.5.2 Create an Android Virtual Device

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

The AVD Manager that you launch from Android Studio by clicking **Tools**, **Android**, and **AVD Manager** has a range of ready-to-use virtual devices, mostly of those devices developed by Google itself, such as the Nexus and Pixel XL range. Other Android device vendors, such as Samsung, provide specifications on their websites that you can use to create the AVD yourself.

Google maintains documentation describing how to manage AVDs. See https://developer.android.com/studio/run/managing-avds.html.

To create an AVD:

1. In Android Studio, launch the Android Virtual Device Manager by selecting **Tools** > **Android** > **AVD Manager**.

2. In the Your Virtual Devices screen, click **Create Virtual Device**.

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

2.5.3 Set Up Your Android Device to Install an App from Your Development Machine

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

To set up your Android device:

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

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

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 the 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 What You Need to Develop an Application for the Universal Windows Platform.
- 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 How to Create a PFX File for MAF Applications and How to Install a PFX File on Windows 10.
- Enable Development Mode on the computer on which you intend to develop the application as described in How to Enable Developer Mode on Windows 10.

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

2.6.1 How to Install Visual Studio

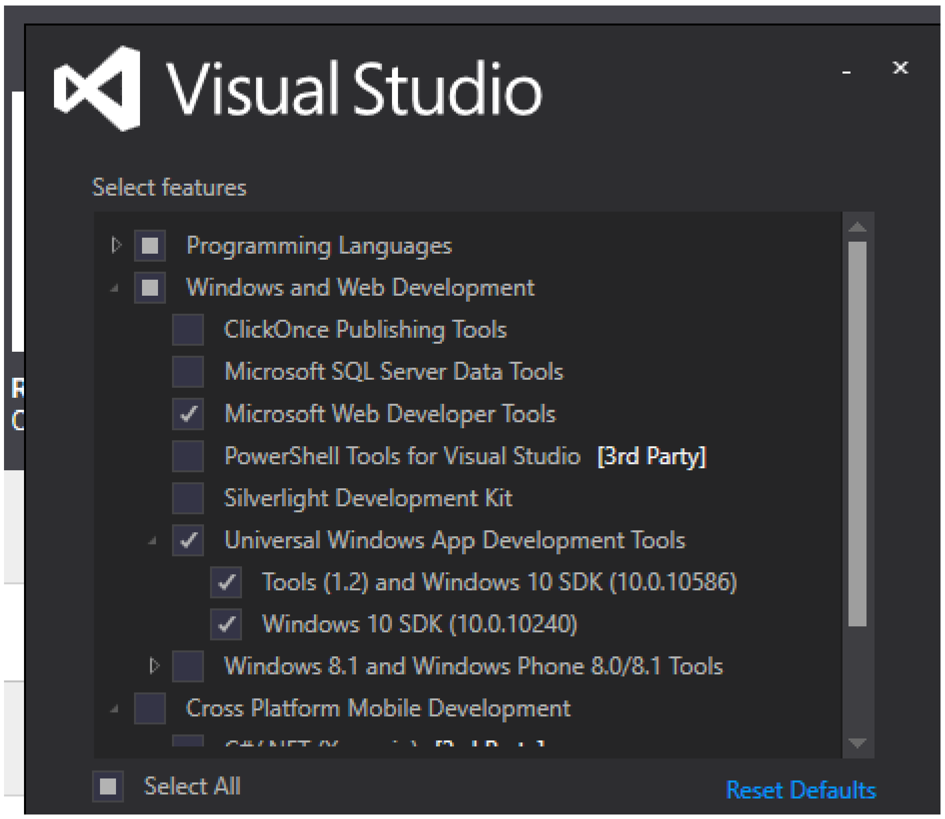
Install Visual Studio 2015 to access the Microsoft tools that are needed to build applications for deployment to Universal Windows Platform. Use the procedure to install Visual Studio.

You download and install Visual Studio from Microsoft Visual Studio web page, see `https://www.visualstudio.com/products/vs-2015-product-editions`. The Visual Studio download includes the Windows 10 SDK.

To install Visual Studio

1. Download and install an edition of Visual Studio 2015.
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 2-3.

Figure 2-3 Installing Visual Studio



Select features

- ■ Programming Languages
- ■ Windows and Web Development
  - ☐ ClickOnce Publishing Tools
  - ☐ Microsoft SQL Server Data Tools
  - ✓ Microsoft Web Developer Tools
  - ☐ PowerShell Tools for Visual Studio [3rd Party]
  - ☐ Silverlight Development Kit
  - ✓ Universal Windows App Development Tools
    - ✓ Tools (1.2) and Windows 10 SDK (10.0.10586)
    - ✓ Windows 10 SDK (10.0.10240)
  - ☐ Windows 8.1 and Windows Phone 8.0/8.1 Tools
- ☐ Cross Platform Mobile Development

■ Select All

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 How to Create a PFX File for MAF Applications

A Personal Information Exchange file is needed to sign Universal Windows Platform based MAF applications. Use the procedure to create a PFX file.

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`).

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.

2.6.3 How to Install a PFX File on Windows 10

Every Personal Information Exchange file on a computer must be manually installed in a certificate store if you want to use it for application signing. Use the procedure to install a PFX file in a certificate store.

Copy or install.

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**.
4. Enter a password, 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** for **Current User**, click **OK**, and then click **Next**.
8. Complete the entries for the dialog.

   This procedure installs the PFX file in the **Personal** certificate store.
9. Repeat the procedure for each of the combinations shown in rows 2 and 3 of the following table.

Store Location	Certificate Store
Current User	Personal
Current User	Trusted People

Store Location	Certificate Store
Local Machine	Trusted People

2.6.4 How to Enable Developer Mode on Windows 10

Enable Developer Mode on the Windows 10 computer so that you can side-load applications. Use the procedure to enable Developer Mode.

Enable Developer Mode to develop MAF applications on Windows 10.

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 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.

2.7 Testing the Environment Setup

Select a sample application from the Samples folder of the MAF extension download, and deploy it either to an iOS-powered device simulator or an Android-powered device emulator to test the environment setup. Use the procedure to test the setup.

Before you begin, set the environment following the instructions in Configuring the Environment for Target Platforms.

You can test your environment setup as follows:

1. In OEPE, open the HelloWorld sample application by selecting **New > Example > MAF Examples**, then click to select the HelloWorld example and click **Finish**.
2. Select **Run > Debug Configurations** from the main menu.
3. From the configuration pane at the left of the Debug Configurations dialog, select MAF Application and click to create a new configuration. If you correctly set the environment, the dialog will show the correct target as shown in Figure 2-4.

   Sometimes **Devices/Emulator** is not listed in the configuration, and clicking **Refresh** still does not display it. In this case, you need to kill and restart it the adb daemon.

   To kill the process, use:

   - Windows: use the process manager

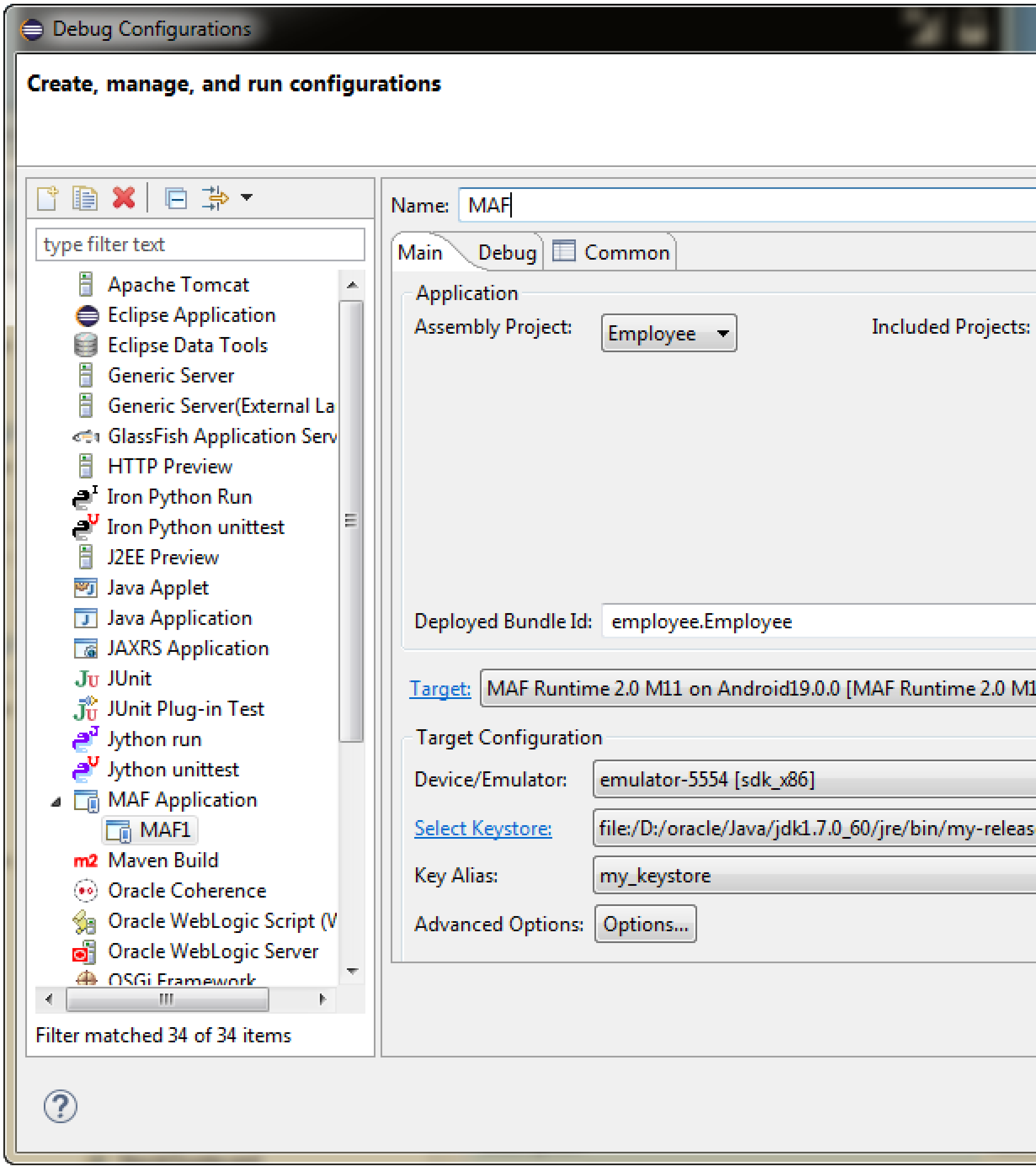
- Mac terminal: use the `kill -9 procID` command

Restart the adb daemon by executing the following command on a terminal:

```
adb devices
```

A deployment that succeeded before adb froze will still be deployed. To debug an application, redeploy it.

Figure 2-4 Debug Configurations Dialog


Debug Configurations

Create, manage, and run configurations

type filter text

Apache Tomcat
Eclipse Application
Eclipse Data Tools
Generic Server
Generic Server(External La
GlassFish Application Serv
HTTP Preview
Iron Python Run
Iron Python unittest
J2EE Preview
Java Applet
Java Application
JAXRS Application
JUnit
JUnit Plug-in Test
Jython run
Jython unittest
MAF Application
MAF1
Maven Build
Oracle Coherence
Oracle WebLogic Script (V
Oracle WebLogic Server
OSGi Framework

Filter matched 34 of 34 items

Name: MAF

Main | Debug | Common

Application
Assembly Project: Employee
Included Projects:
Deployed Bundle Id: employee.Employee

Target: MAF Runtime 2.0 M11 on Android19.0.0 [MAF Runtime 2.0 M1

Target Configuration
Device/Emulator: emulator-5554 [sdk_x86]
Select Keystore: file:/D:/oracle/Java/jdk1.7.0_60/jre/bin/my-releas
Key Alias: my_keystore
Advanced Options: Options...

4. Click **Debug** to deploy the application to the target platform. When the deployment is complete, you will see a BUILD SUCCESSFUL message in the Log Window.

   See the following sections in *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*:

   - Deploying an iOS Application
   - Deploying an Android Application

See Deploying MAF Applications in *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*.

After a successful deployment (which might take a few minutes), your iOS-powered device simulator or Android-powered device emulator will display the HelloWorld application icon that you have to activate to launch the application.

3

Migrating Your Application to MAF 2.3.3

This chapter provides information about migrating applications created using earlier releases of MAF to MAF 2.3.3.

This chapter includes the following sections:

3.1 Migrating an Application to MAF 2.3.3

Learn about the issues you need to be aware of when you migrate a MAF application created in an earlier version to MAF 2.3.3.

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

This release of MAF 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. See Using Xcode 8 and Deploying to iOS 10 with MAF 2.3.3.

Note: An upcoming release of MAF will remove APIs that have been deprecated in previous releases. Please review compiler warnings from your code and the *Java API Reference for Oracle Mobile Application Framework* in order to prepare the transition of your application to use supported APIs.

- 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 also configure migrated applications that run on iOS 9 and iOS 10 to use WKWebView, as described in Configuring Application Features with AMX Content to Use WKWebView on iOS 9 and iOS 10.

- MAF 2.3.0 and later uses 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 2.3.3.

- The `RestServiceAdapter` interface has a new package location (`oracle.maf.api.dc.ws.rest`). The functionality that this interface specifies remains unchanged. For information about creating a REST web service adapter, see Creating a Rest Service Adapter to Access Web Servicesin *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*.

- 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 Using Web Services in a MAF Applicationin *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*.

- 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 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 (OEPE Edition)*.

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 2.3.2. If you migrate an application to MAF 2.3.2 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.

For MAF applications that you migrate to MAF 2.3.2 (irrespective of the release from which you migrate), we recommend that you set the preemptive property to `true` on the following security policies:

- `http_basic_auth_over_ssl_client_policy`
- `wss_http_token_client_policy`
- `wss_http_token_over_ssl_client_policy`

Setting the preemptive property to `true` on these policies inserts a basic authentication header into the first request to a secured REST web service.

Also, if your MAF application uses Web SSO authentication, you must configure the `oracle/http_cookie_client_policy` security policy for all REST/HTTP web service connections used by the application. Without this change, your end users may see errors when accessing services in the application. See Specifying REST Service Connections in *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*.

MAF 2.1.0 used newer versions of Apache Cordova and Java. It also changed the way that OEPE registered plugins in your MAF application. For SSL, it delivers a `cacerts` file that contains new CA root certificates.

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 information about how to connect to the SQLite database, see Using the Local SQLite Databasein *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*.

3.2 Using Xcode 8 and Deploying to iOS 10 with MAF 2.3.3

MAF 2.3.3 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 you attempt to deploy a MAF 2.3.3 application to iOS. With this installation, Xcode 8.x replaces Xcode 7.x. No other changes are required. If you want to maintain separate development environments for MAF 2.3.3 (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. See Setting the Device Signing Options in *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*.
- 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. See How to Create an iOS Deployment Configuration and Enabling Push Notifications in *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*.

- 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. See Providing Usage Descriptions for Plugins that Access Device Capabilities on iOS in *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*.

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. 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
   ```

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

New MAF applications that you create using this release 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, rather than UIWebView, 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.

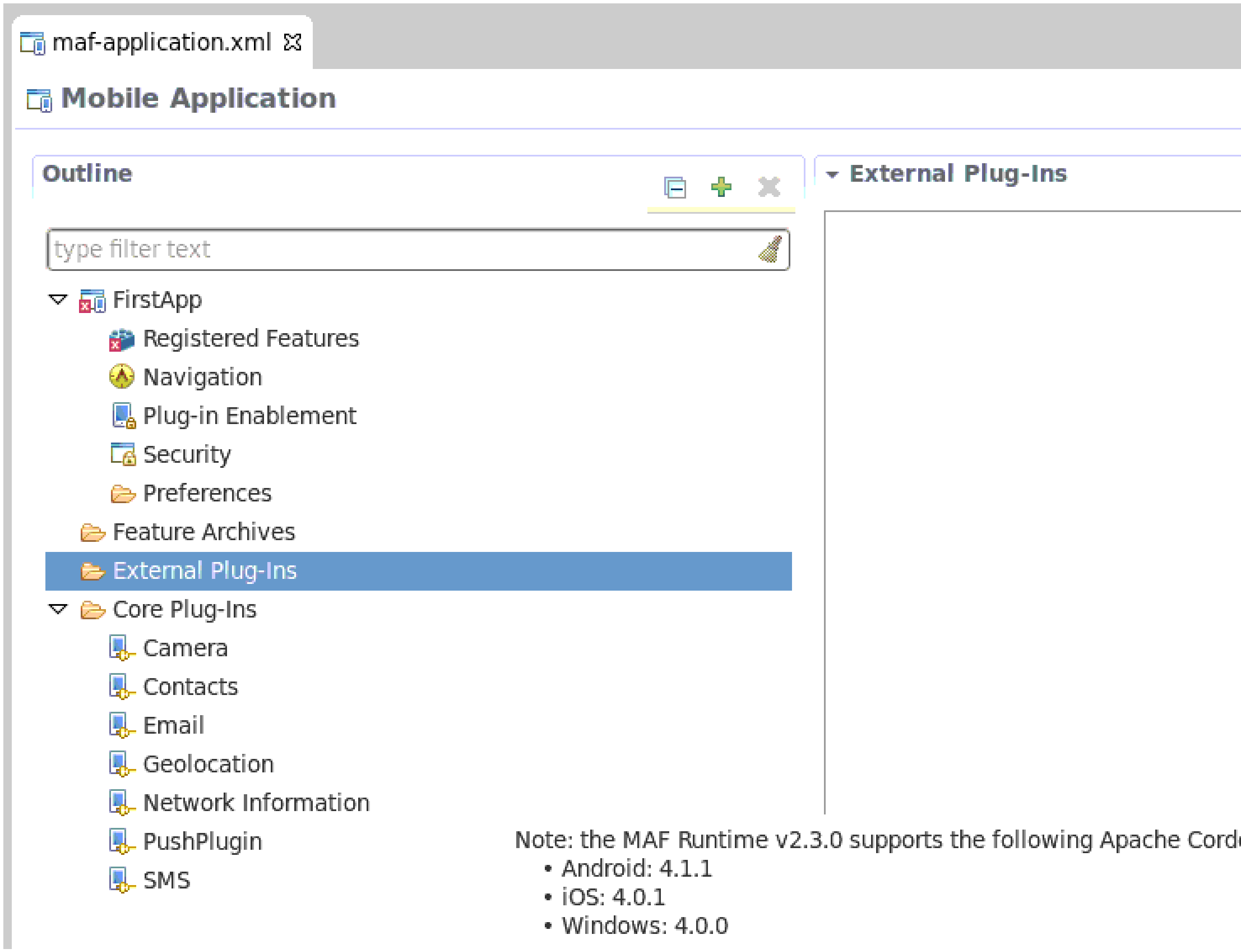
WKWebView is used on iOS 9 only. UIWebView will always be used on iOS 8.

3.4 Migrating Cordova Plugins from Earlier Releases to MAF 2.3.3

Learn how to verify whether your MAF application from an earlier release can use the same plugins.

MAF 2.3.0 and later use new versions of Cordova (4.x). To complete the migration and make sure that your migrated MAF application can use the plugins it used previously, verify that MAF supports the version of the plugin. The External Plug-Ins section 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.

Figure 3-1 Cordova Platform Versions Supported by MAF

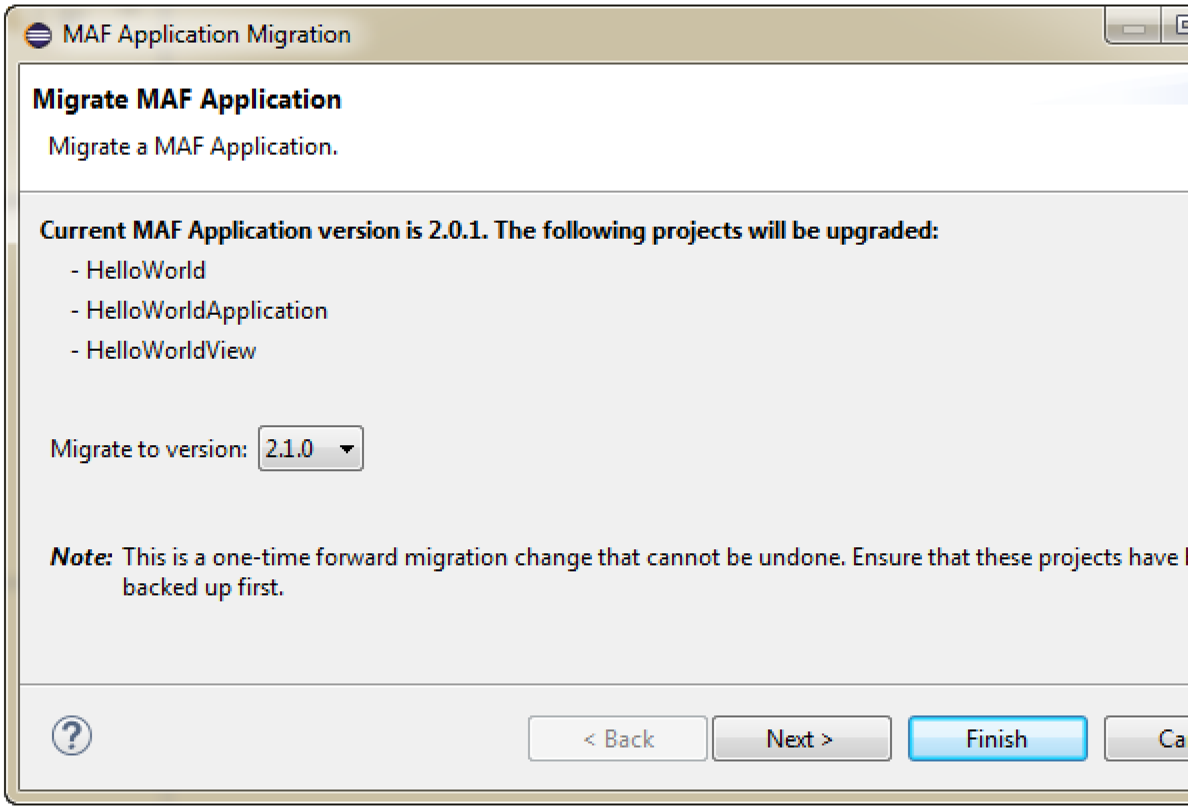


Note: the MAF Runtime v2.3.0 supports the following Apache Cord

- Android: 4.1.1
- iOS: 4.0.1
- Windows: 4.0.0

Using the OEPE migration wizard, shown in Figure 3-2, you can migrate from earlier versions of MAF. For example, if you have a MAF 2.0.0 application you can choose to migrate it to a later version of MAF, such as MAF 2.1 0.

Figure 3-2 Migration Wizard



Migrate MAF Application

Migrate a MAF Application.

Current MAF Application version is 2.0.1. The following projects will be upgraded:

- HelloWorld
- HelloWorldApplication
- HelloWorldView

Migrate to version: 2.1.0

Note: This is a one-time forward migration change that cannot be undone. Ensure that these projects have
backed up first.

3.4.1 How to Migrate an Application

OEPE has a migration wizard that makes it easy to migrate your application. You can choose to migrate to any available version using the wizard.

To migrate an application:

1. Open the application in OEPE.
2. Right click the assembly project and choose **Configure** > **Migrate MAF Application**. The migration wizard opens.
3. Select the MAF version you want to migrate to and click **Next**.
4. The Configure Deployments page of the wizard shows that the initial deployment is disabled. Click **Add** to add a new deployment target. Click **Finish**.

3.4.2 What Happens When you Migrate an Application

MAF applications developed using earlier releases of MAF registered plugins in the MAF Application Editor. This release of MAF registers plugins in the same editor, but due to changes to Apache Cordova the functionality is different.

Examine the application once it has migrated and make any appropriate changes. For example, enable additional core plugins and register external plugins in the MAF Application Editor, and specify the plugins used by features in the MAF Features Editor. See *Using Plugins in MAF Applications*.

To complete the migration and make sure that your migrated MAF application can use the plugins it used previously, verify that the:

- Version of the plugin is supported by MAF.
- Obtain a newer version of the plugin if the plugin was created using an earlier release of Cordova.

3.5 Security Changes in Release 2.2.1 and Later of MAF

Learn about security issues that have changed since MAF 2.2.1.

Starting with MAF 2.3.0, 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 Apple iOS 9's 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 Java 8's JVM 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.

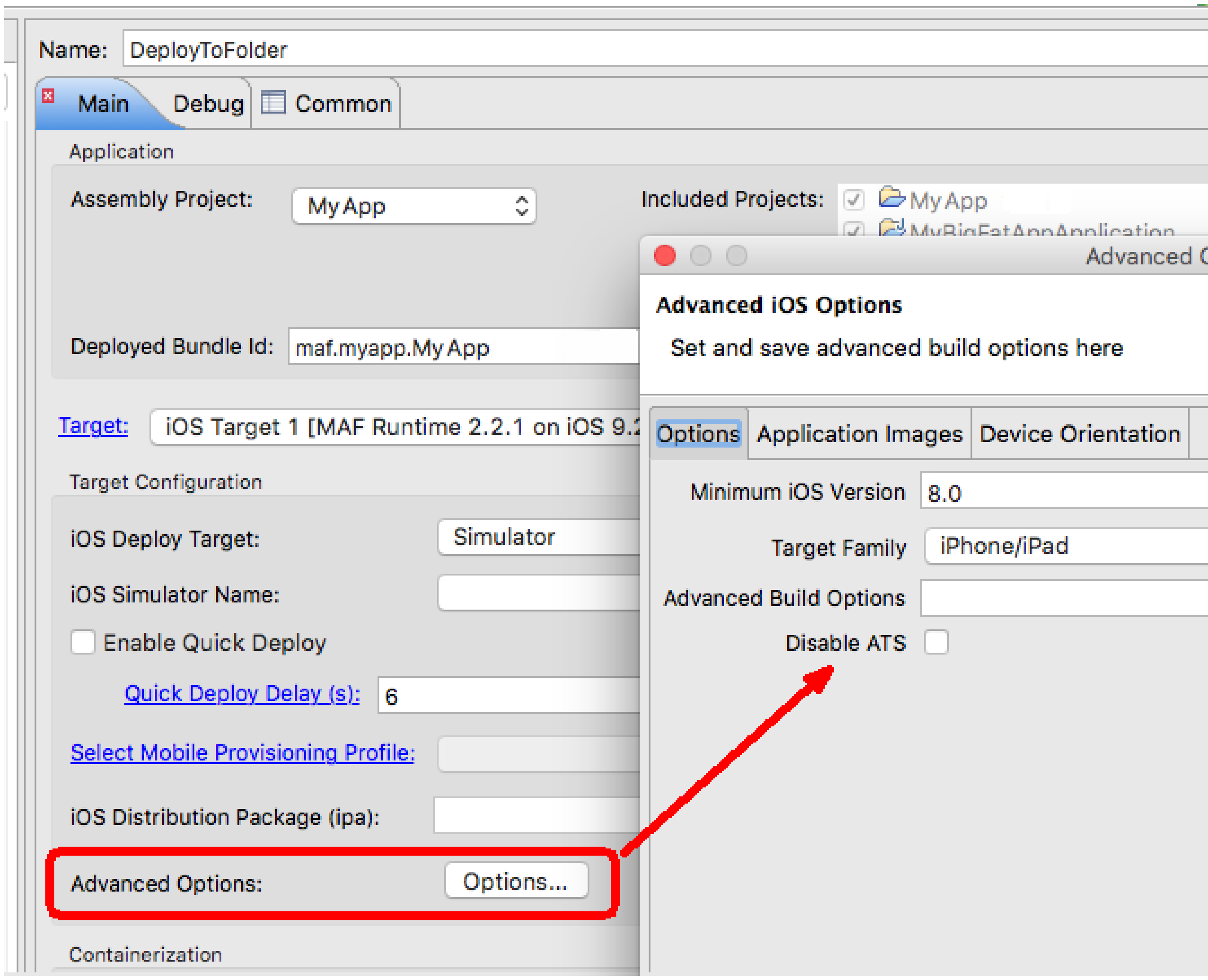
Disabling App Transport Security for MAF Applications on iOS Devices

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 OEPE, click the **debug** button and select **Debug Configurations**.
2. In the Target Configuration pane of the Main tab, click **Options**.
3. In the Advanced iOS Options dialog that appears, select the **Disable ATS** checkbox, as shown in Figure 3-3, 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.

Figure 3-3 Disable ATS Option for MAF Application on iOS Devices



Name: DeployToFolder

Assembly Project: My App

Included Projects: My App

Deployed Bundle Id: maf.myapp.My App

Target: iOS Target 1 [MAF Runtime 2.2.1 on iOS 9.

iOS Deploy Target: Simulator

iOS Simulator Name:

Enable Quick Deploy

Quick Deploy Delay (s): 6

Select Mobile Provisioning Profile:

iOS Distribution Package (ipa):

Advanced Options: Options...

Advanced iOS Options

Set and save advanced build options here

Minimum iOS Version 8.0

Target Family iPhone/iPad

Advanced Build Options

Disable ATS

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:

For the list of cipher suites that Java supports, see the Cipher Suites section athttp://docs.oracle.com/javase/8/docs/technotes/guides/security/SunProviders.html

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 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
   ```

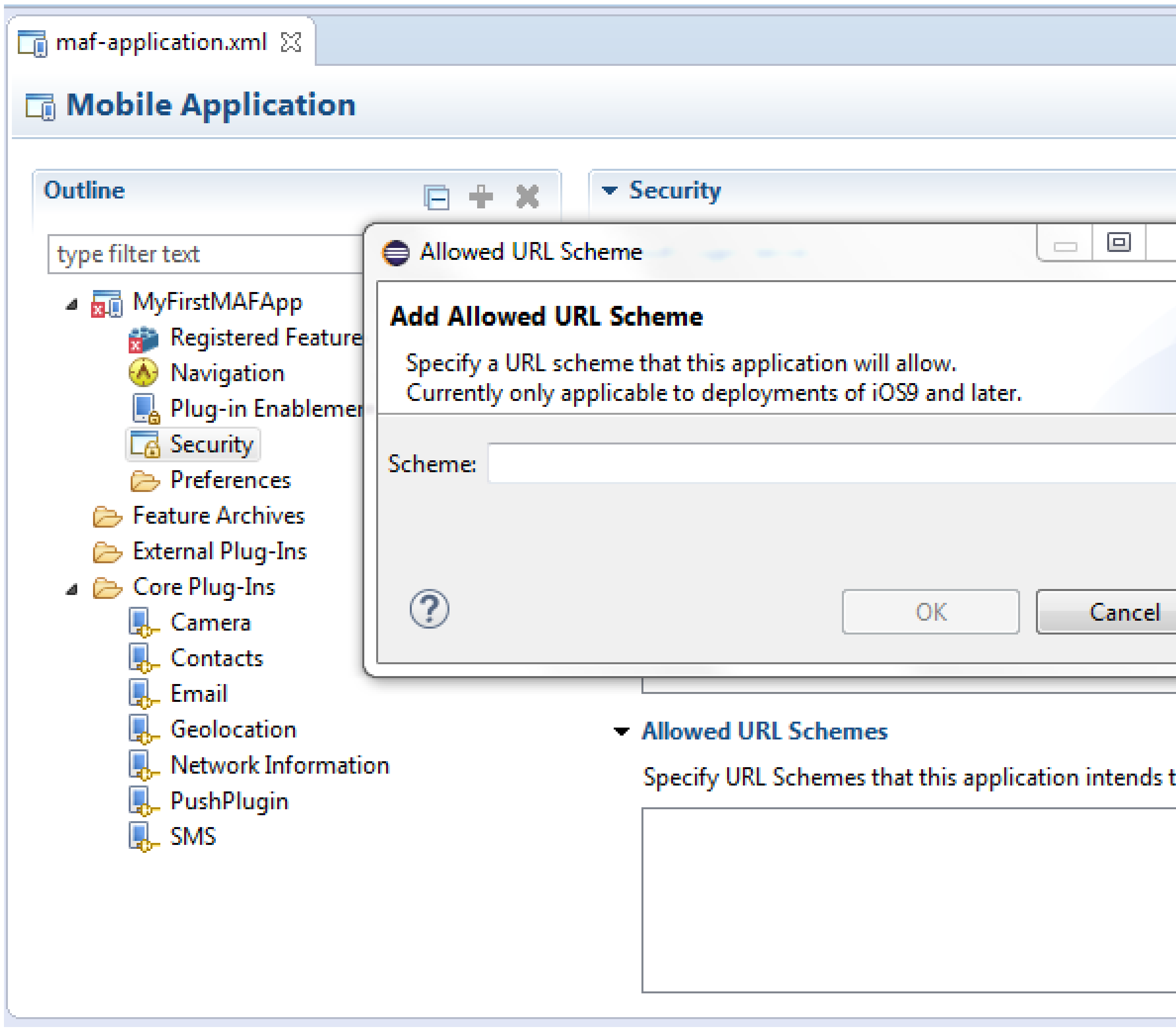
3.6 Migrating MAF Applications that Use Customer URL Schemes to Invoke Other Applications

Learn how to use URL schemes when you migrate MAF applications that invoke another application.

If the application you migrate to MAF 2.3.0 or later uses a custom URL scheme to invoke another application, and you deploy the application to iOS 9 or higher, add the scheme(s) to the Allowed URL Schemes list in the Security page of the `maf-application.xml` file's editor.

This change addresses iOS 9's requirement that applications declare any URL schemes they use to invoke other applications. Click the Add icon in the Allow URL Schemes section of the Security page to add the custom URL scheme, as shown in Figure 3-4.

Figure 3-4 Registering a Custom URL Scheme that a MAF Application Uses to Invoke Another Application



Specify a URL scheme that this application will allow.
Currently only applicable to deployments of iOS9 and later.

Specify URL Schemes that this application intends t

3.7 Migrating to JDK 8 in MAF 2.3.3

Learn about the changes you need to make when you migrate a MAF application compiled in an earlier version of Java to JDK8.

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 2.3.3 for the first time, OEPE makes the following changes:

- Renames the configuration file that specifies the startup parameters of the JVM from `cvm.properties` to `maf.properties`. For information about the `maf.properties` file, see *How to Enable Debugging of Java Code and JavaScript*.

- Replaces instances (if any) of the following import statement in the application's Java source files:

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

  With:

```
java.util.logging
```

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

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

  With:

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

  For information about the `logging.properties` file, see *How to Configure Logging Using the Properties File*.

3.8 Migrating to a New cacerts File for SSL in MAF 2.3.3

Learn about the tasks to perform to migrate to a new `cacerts` file.

Make sure that the `cacerts` file packaged in the MAF application that you publish for your 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 your MAF application's `cacerts` file if the HTTPS server contains certificates not present in your MAF application's `cacerts` file. 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 JDK 8's `keytool` utility to view and manage the certificates in your MAF application's `cacerts` file. The following example demonstrates how you might use JDK 8's `keytool` utility to display the list of certificates in a `cacerts` file:

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

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

For information about the `cacerts` file and using SSL to secure your MAF application, see *Creating Certificates to Access Servers That Use Self-Signed Certificates for SSL*.

Use JDK 8's `keytool` utility, as previously described, to manage the certificates in MAF 2.1.0's `cacerts` file to meet the requirements of the environment where your MAF application will be used. The `cacerts` file lists the issuers of CA root certificates:

```
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
```