

Oracle® Enterprise Pack

Installing Oracle Enterprise Pack

12c (12.2.1.5)

E79538-01

December 2016

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

Oracle Enterprise Pack Installing Oracle Enterprise Pack, 12c (12.2.1.5)

E79538-01

Copyright © 2015, 2016, Oracle and/or its affiliates. All rights reserved.

Primary Author: Sujatha Joseph, Walter Egan

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

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

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

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

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

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

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

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

Contents

Preface	v
Audience	v
Documentation Accessibility	v
Related Documents.....	v
Conventions.....	v
What's New in This Guide for This Release	vii
1 Installing Oracle Enterprise Pack for Eclipse	
1.1 Installing Your OEPE Application	1-1
1.2 System Requirements.....	1-1
1.3 Installing with the Distro.....	1-2
1.4 Installing with the Eclipse Installer.....	1-2
1.4.1 How to install using the Eclipse Installer	1-2
1.4.2 Choosing What to Install.....	1-2
1.4.3 Selecting the Components to Install	1-3
1.5 Installing with Eclipse Marketplace.....	1-4
1.6 Installing Using the Repository	1-5
1.7 Updating an Existing Installation.....	1-5
1.7.1 How to Update Using Check for Updates Option	1-5
1.7.2 Troubleshooting Update	1-6
2 Configuring Mobile Application Framework	
2.1 Introduction to the MAF Environment	2-1
2.2 Prerequisites for Developing MAF Applications.....	2-1
2.2.1 What You Need to Develop an Application for the iOS Platform.....	2-2
2.2.2 What You Need to Develop an Application for the Android Platform	2-2
2.2.3 What You Need to Develop an Application for the Universal Windows Platform	2-3
2.3 Setting Up OEPE	2-3
2.3.1 How to Specify JDK 8's JRE as the Default in OEPE.....	2-3
2.3.2 How to Configure the Development Environment for Target Platforms.....	2-5

2.4	Setting Up Development Tools for the iOS Platform	2-6
2.4.1	How to Install Xcode and iOS SDK	2-6
2.4.2	How to Set Up an iPhone or iPad	2-7
2.4.3	How to Set Up an iPhone or iPad Simulator	2-7
2.5	Setting Up Development Tools for the Android Platform	2-8
2.5.1	How to Install the Android SDK.....	2-8
2.5.2	How to Set Up an Android-Powered Device.....	2-10
2.5.3	How to Set Up an Android Emulator.....	2-10
2.6	Setting Up Development Tools for the Universal Windows Platform	2-18
2.6.1	How to Install Visual Studio.....	2-19
2.6.2	How to Create a PFX File for MAF Applications	2-20
2.6.3	How to Install a PFX File on Windows 10	2-21
2.6.4	How to Enable Developer Mode on Windows 10.....	2-22
2.7	Testing the Environment Setup	2-23

3 Migrating Your Application to MAF 2.3.2

3.1	Migrating an Application to MAF 2.3.2.....	3-1
3.2	Configuring Application Features with AMX Content to Use WKWebView on iOS 9	3-3
3.3	Migrating Cordova Plugins from Earlier Releases to MAF 2.3.2.....	3-4
3.3.1	How to Migrate an Application	3-5
3.3.2	What Happens When you Migrate an Application	3-5
3.4	Security Changes in Release 2.2.1 and Later of MAF	3-6
3.5	Maintaining Separate Xcode Installations for MAF 2.3.2 and MAF 2.2.0.....	3-8
3.6	Migrating MAF Applications that Use Customer URL Schemes to Invoke Other Applications	3-9
3.7	Migrating to JDK 8 in MAF 2.3.2	3-10
3.8	Migrating to a New cacerts File for SSL in MAF 2.3.2.....	3-11

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.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
<code>monospace</code>	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
Migrating an Application to MAF 2.3.2.	Revised to note that this release includes a new version of the <code>java.security</code> file in migrated MAF applications.
Configuring Application Features with AMX Content to Use WKWebView on iOS 9.	Added to describe how MAF applications that you deploy to iOS 9 devices can render AMX content using WKWebView, a web view that offers better performance than UIWebView.
Security Changes in Release 2.2.1 and Later of MAF.	Revised with a recommendation not to disable App Transport Security (ATS). Apple plans to enforce use of ATS from January, 01, 2017.

Installing Oracle Enterprise Pack for Eclipse

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

It contains the following sections:

- [Installing Your OEPE Application](#)
- [System Requirements](#)
- [Installing with the Distro](#)
- [Installing with the Eclipse Installer](#)
- [Installing with Eclipse Marketplace](#)
- [Installing Using the Repository](#)
- [Updating an Existing Installation](#)

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

For information about 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.

To install using the distro:

1. From the Oracle Technology Network (OTN) web site: <http://www.oracle.com/technetwork/developer-tools/eclipse/downloads/index.html>, 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.

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. See [Choosing What to Install](#).
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.

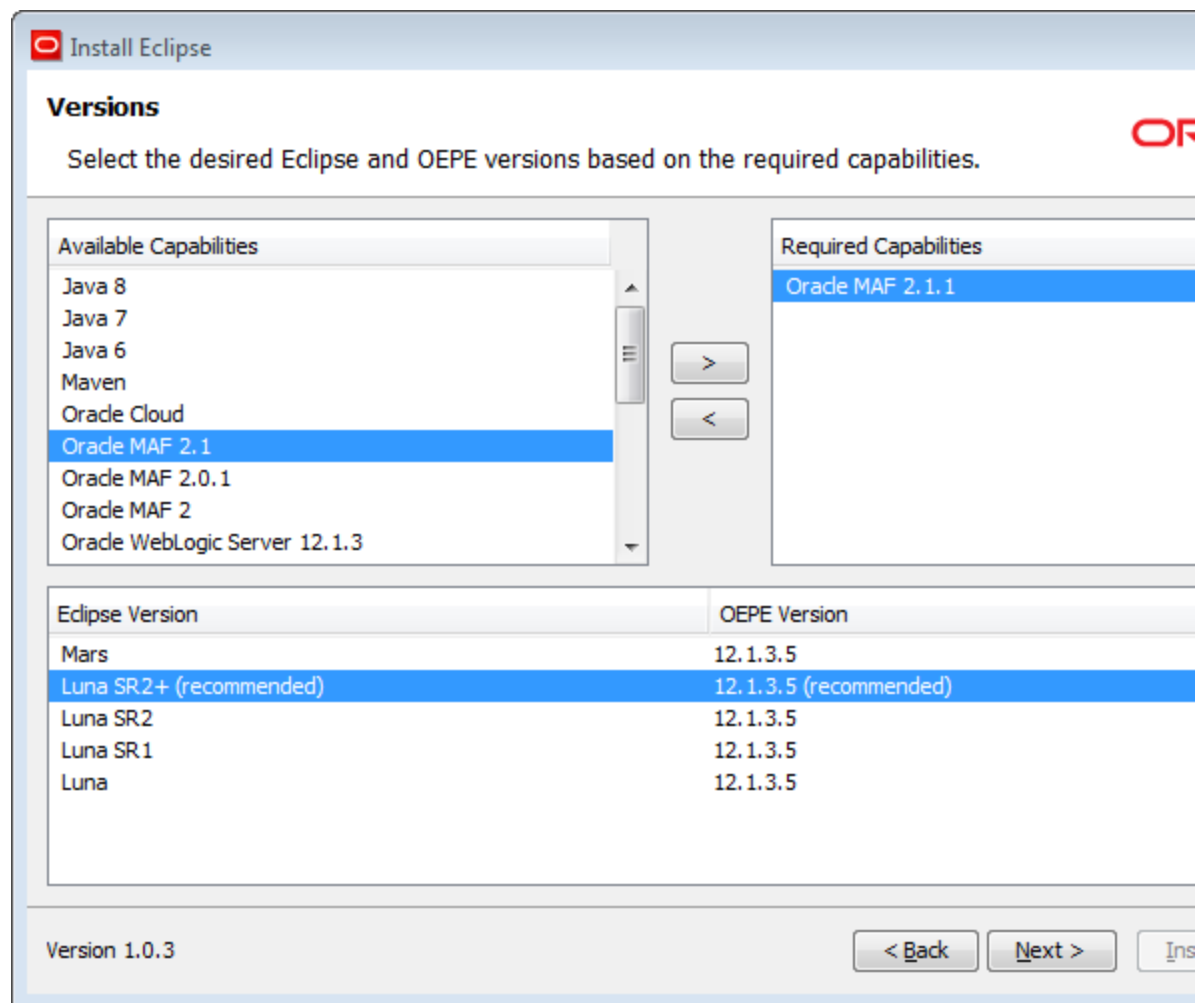
1.4.2.2 Choose an OEPE Version Based on an Eclipse Version

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](#).

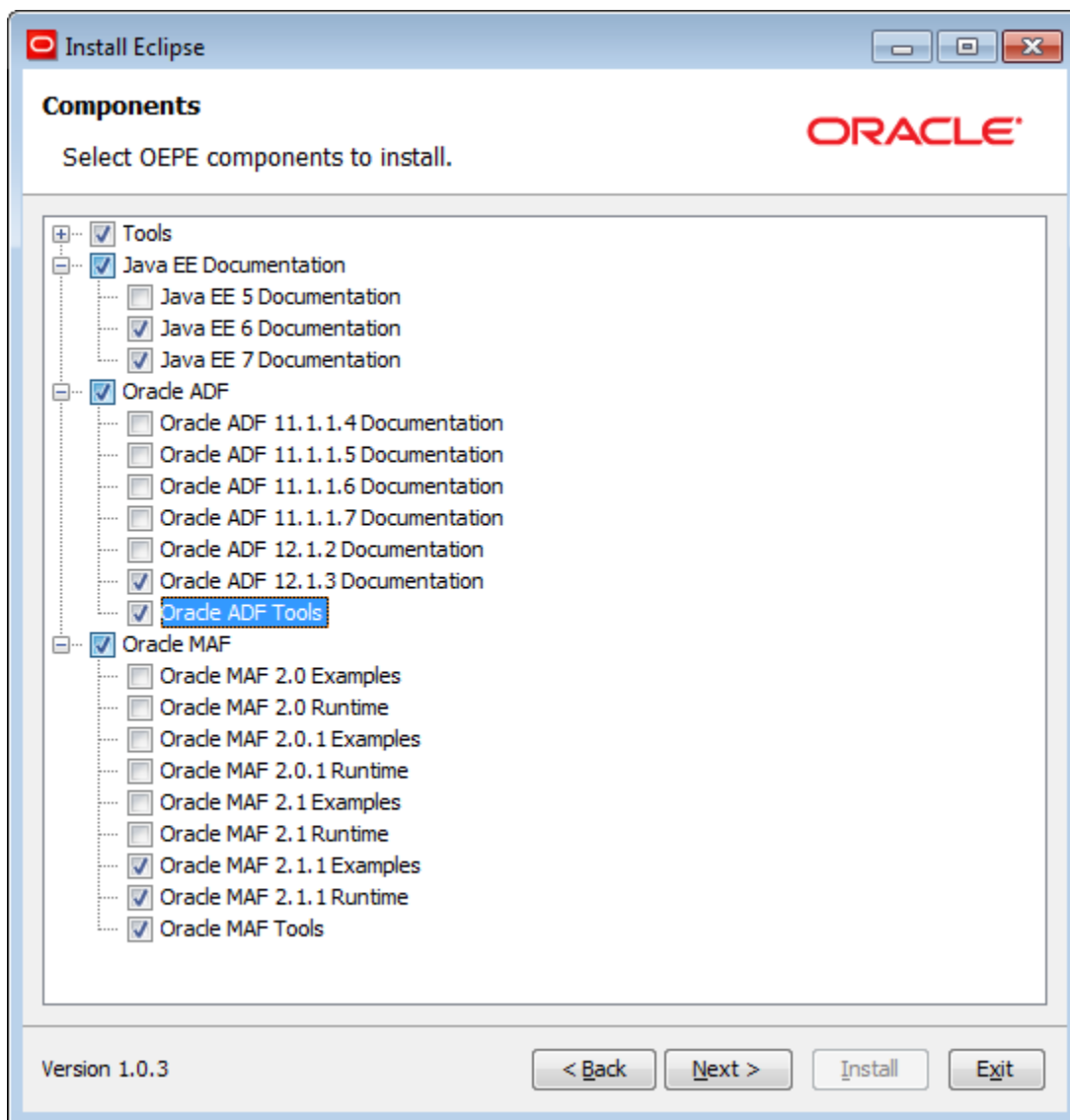
Figure 1-1 *Choosing the Eclipse and OEPE Versions*



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



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.

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 the Oracle Technology Network (OTN) web site at <http://www.oracle.com/technetwork/developer-tools/eclipse/downloads/index.html>.

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.

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:

- [Introduction to the MAF Environment](#)
- [Prerequisites for Developing MAF Applications](#)
- [Setting Up OEPE](#)
- [Setting Up Development Tools for the iOS Platform](#)
- [Setting Up Development Tools for the Android Platform](#)
- [Setting Up Development Tools for the Universal Windows Platform](#)
- [Testing the Environment Setup](#)

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

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 [How to Install the Android SDK](#))
- OEPE (see [Installing Oracle Enterprise Pack for Eclipse](#) .)

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 emulator: if you are to use an emulator, see [How to Set Up an Android Emulator](#); 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)*.
- Android-powered device. See [How to Set Up an Android-Powered Device](#).

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

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.

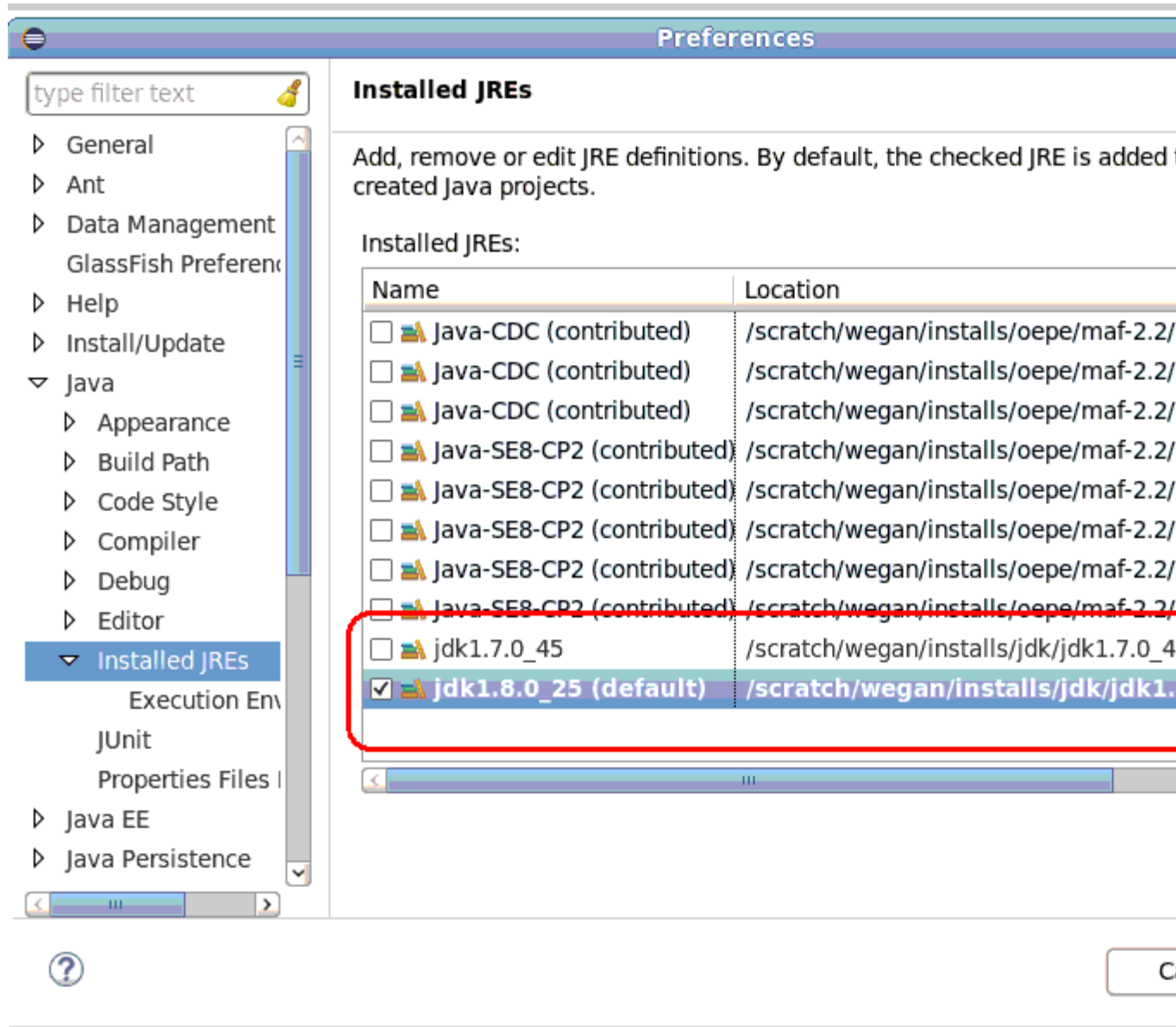
You must specify the JRE packaged with JDK 8 as the default JRE in OEPE so that OEPE adds this JRE to the build path of newly-created MAF applications.

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 Dialog.



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 [How to Install the Android SDK](#)) 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

Using an operating system that is compatible with OEPE and the Android platform, configure the Android SDK if you want to deploy applications to Android devices or emulators.

In addition to the general-purpose tools listed in [Prerequisites for Developing MAF Applications](#), you might want to set up an Android-powered device when getting ready for 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 create its configuration (see [How to Set Up an Android Emulator](#)).

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

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

2.5.1 How to Install the Android SDK

Use the procedure to download the Android SDK starter package from the Android Developers website and install it. Alternatively, install components separately to build applications for Android-powered devices.

Android SDK includes the 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.

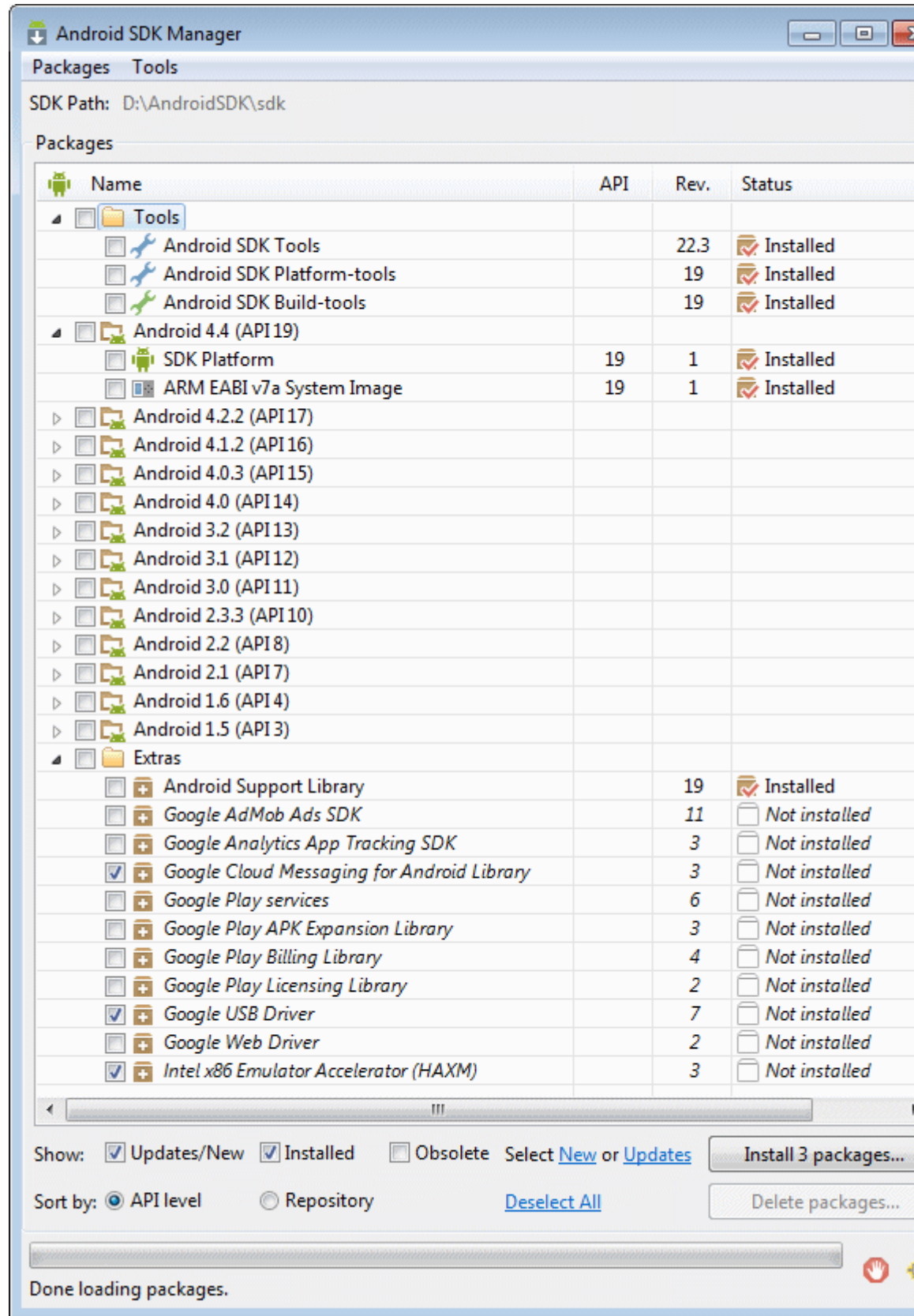
Before you begin:

Make sure that your environment meets the operating system, JDK version, and hardware requirements listed in System Requirements on 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. By default, the Android SDK Tools, Android SDK Platform-tools and several other packages are installed. You can install additional packages using the Android SDK Manager, as shown in [Figure 2-2](#).

Figure 2-2 Android SDK packages to install



3. Complete the installation by following the instructions provided in Installing the Android SDK on the Android Developers website at <http://developer.android.com/sdk/installing/index.html>.

Note:

Do not start the Android SDK Manager when prompted. Instead, follow the instructions on installing the Eclipse plugin called Android Development Tools (ADT), located at <http://developer.android.com/sdk/installing/installing-adt.html>

2.5.2 How to Set Up an Android-Powered Device

Follow the instructions on the Android Developers website to set up devices. Edit the `android_winusb.inf` file with the correct values for the device to set up an Android-powered device for the deployment of MAF applications.

In your MAF application development and deployment, you can use either the Android device itself, which is preferable for testing *Testing MAF Applications*), 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 Using Hardware Devices on 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 (OEPE Edition)*.

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.NTx86]` and `[Google.NTamd64]` sections of the `android_winusb.inf` file.

See Google USB Driver on the Android Developers website 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. Review the configuration of the settings to ensure that it matches your requirements.

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 Managing Virtual Devices 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 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.
- Default settings for the Hardware field (see the Hardware Options table at <http://developer.android.com/tools/devices/managing-avds.html#hardwareopts>) should be acceptable for a typical MAF application. For 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 Using the Android Emulator on the Android Developers website at <http://developer.android.com/tools/devices/emulator.html>.

2.5.3.1 Configuring the Android Emulator

Complete the basic setup of the Android emulator, and then configure the emulator state, SD card, network, and the network proxy.

After the basic Android emulator setup is complete, you may want 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

Save the emulator state and reuse it to save load time. You can also save emulator state from the CLI.

You can reduce the load time of the emulator 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 the avd folder of another emulator 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

Follow the instructions on the Android Developers website to create, save, and reuse an SD card.

SD Card Emulation on 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:

```
Available Android Virtual Devices:
Name: AVD_for_Nexus_S
Device: Nexus S (Google)
Path: C:\Users\username\.android\avd\AVD_for_Nexus_S.avd
Target: Android 5.0.1 (API level 21)
Tag/ABI: default/armeabi-v7a
Skin: 480x800
-----
Name:      AndroidEmulator2
Device:   Nexus S (Google)
Path:     C:\Users\username\.android\avd\AndroidEmulator2.avd
Target:   Android 5.0.1 (API level 21)
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)
```

See the Android Tools Help at <http://developer.android.com/tools/help/index.html>.

2.5.3.1.3 Configuring the Network

An Android emulator requires access to the host computer so that applications can be deployed to the emulator.

From the Android emulator, you can access your host computer through the 10.0.2.2 IP address. 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.

See Android Debug Bridge on in the Android Developers website at <http://developer.android.com/tools/help/adb.html>.

2.5.3.1.4 Configuring the Network Proxy

A computer that is protected by a corporate firewall requires a proxy to connect to an emulator. Use the procedure to start the emulator, and to configure 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. Select **Wireless & Networks**.
- b. Select **Mobile Networks**, and then select **Access Point Names**.
- c. Select the appropriate internet option.
- d. Set the proxy, port, username, and password using the Edit access point list.

2.5.3.2 Speeding Up the Android Emulator

Install Intel HAXM so that Intel drivers quicken deployment to an emulator. Some versions of operating systems require the Intel Hotfix for the emulator to work with Intel HAXM.

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 (KVM) 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:

- Version 17 or later of 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.

Note:

It may be necessary to edit your system BIOS to enable Intel VT-x support. To do this, restart your computer but do not let it boot normally: interrupt your boot process, then select the menu to edit your BIOS. Scroll through the BIOS selections until you see the entry for VT-x, then toggle it to select Enabled.

- At least 1 GB of available RAM.

To download the Intel HAXM, either use the Android SDK Manager (see the Speeding Up the Android Emulator on Intel Architecture article available at <http://software.intel.com/en-us/android/articles/speeding-up-the-android-emulator-on-intel-architecture>) or use the following Intel locations:

- For Microsoft Windows: <http://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager-end-user-license-agreement>
- Mac OS X: <http://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager-end-user-license-agreement-macosx>
- For Linux: <http://software.intel.com/en-us/blogs/2012/03/12/how-to-start-intel-hardware-assisted-virtualization-hypervisor-on-linux-to-speed-up-intel-android-x86-gingerbread-emulator>

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

- For Microsoft Windows: <http://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager-end-user-license-agreement-windows-hotfix>
- Mac OS X: <http://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager-end-user-license-agreement-macos-hotfix>

See the following:

- *Installation Guide and System Requirements - Windows* at <http://software.intel.com/en-us/android/articles/installation-instructions-for-intel-hardware-accelerated-execution-manager-windows>
- *Installation Guide and System Requirements - Mac OS X* at <http://software.intel.com/en-us/android/articles/installation-instructions-for-intel-hardware-accelerated-execution-manager-mac-os-x>
- *Installation Guide and System Requirements - Linux* at <http://software.intel.com/en-us/blogs/2012/03/12/how-to-start-intel-hardware-assisted-virtualization-hypervisor-on-linux-to-speed-up-intel-android-x86-gingerbread-emulator>

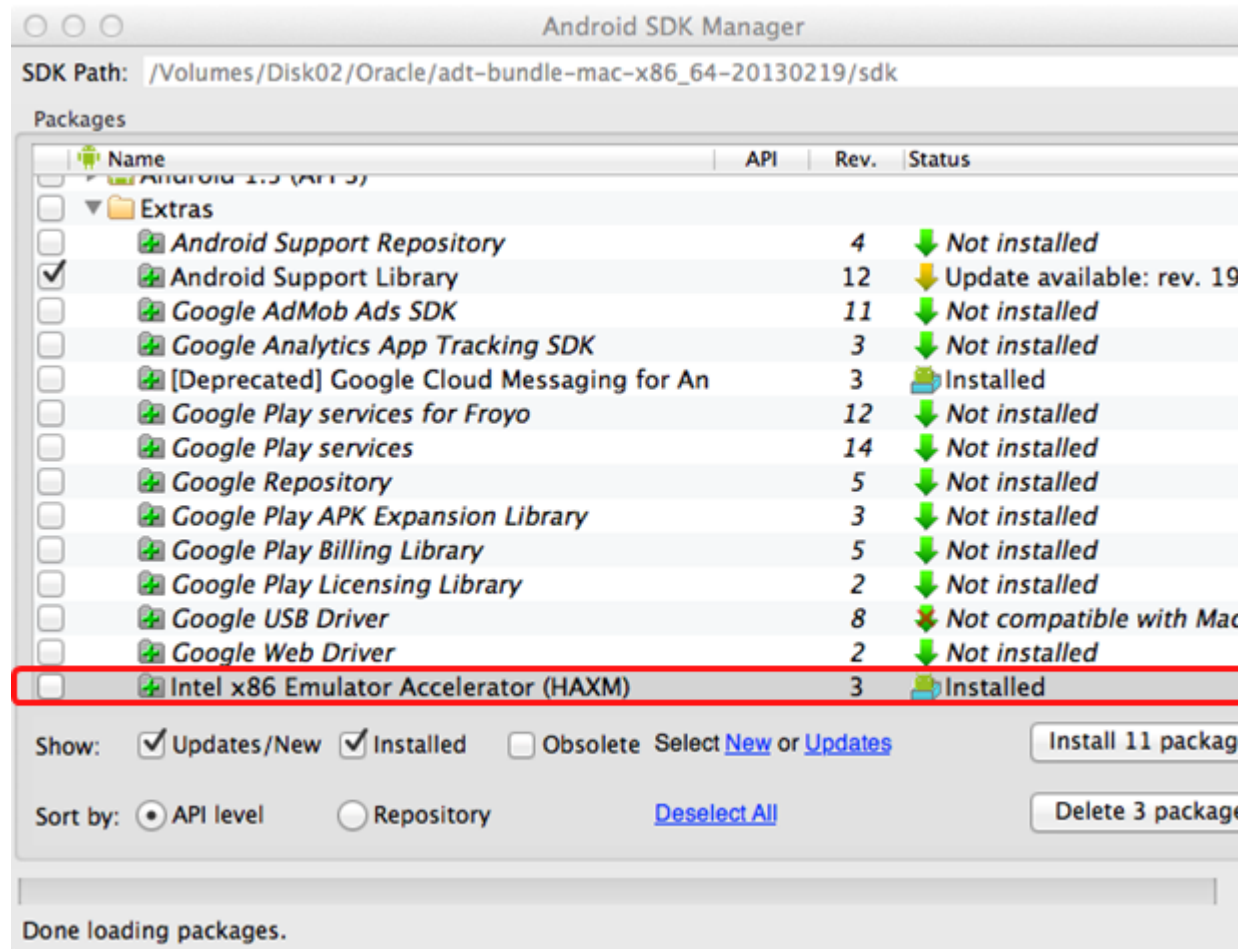
2.5.3.2.1 Configuring AVD for Intel HAXM

Use the Android SDK Manager to install the Intel Atom processor x86 system image for the relevant Android API version. The image is required for an installed Intel HAXM to function.

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 2-2](#)). As described in the [Speeding Up the Android Emulator on Intel Architecture](#) article at <http://software.intel.com/en-us/android/articles/speeding-up-the-android-emulator-on-intel-architecture>:

- 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**.

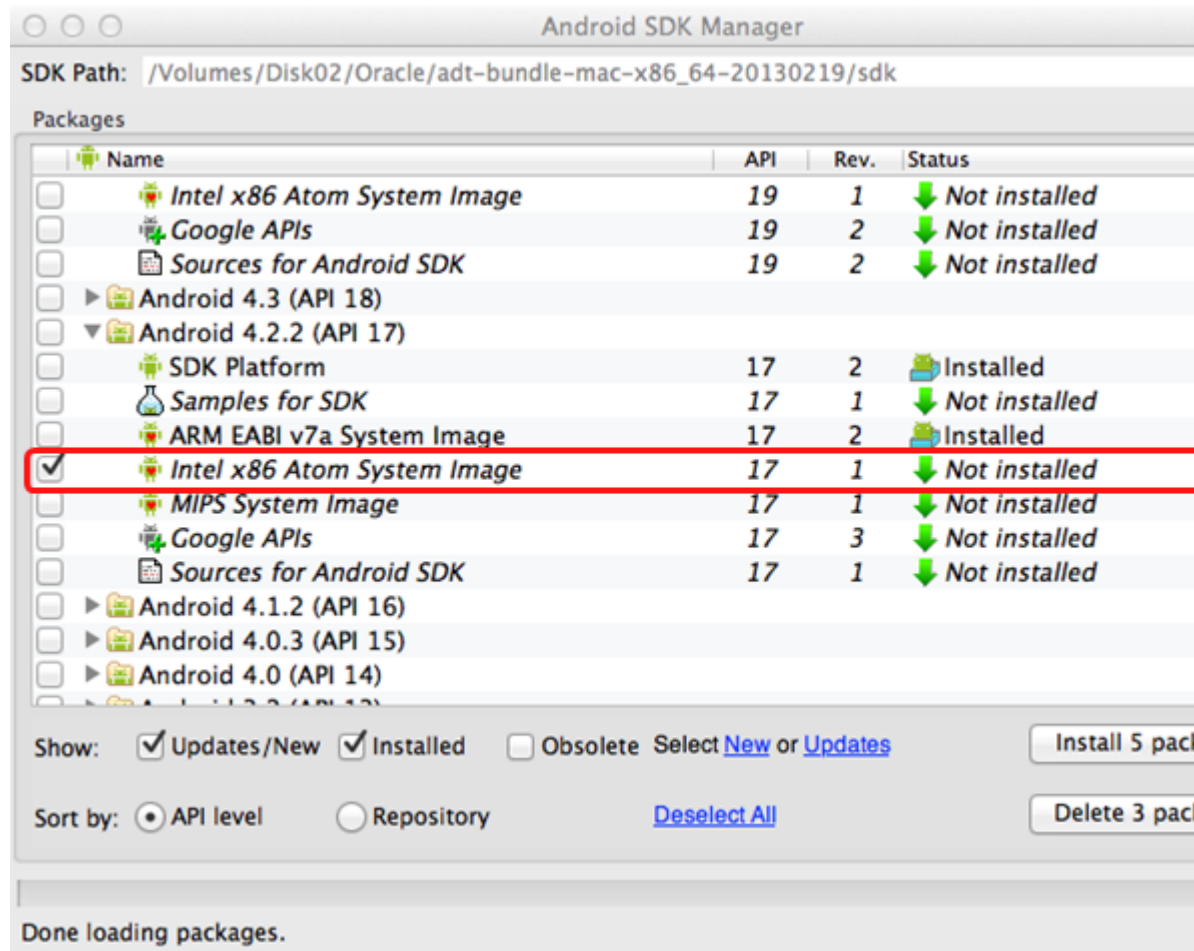
Once you have installed the package, the status changes to Installed, which is not accurate. The SDK only copies the Intel HAXM executable on your computer; you have to manually install the executable.

Figure 2-3 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.

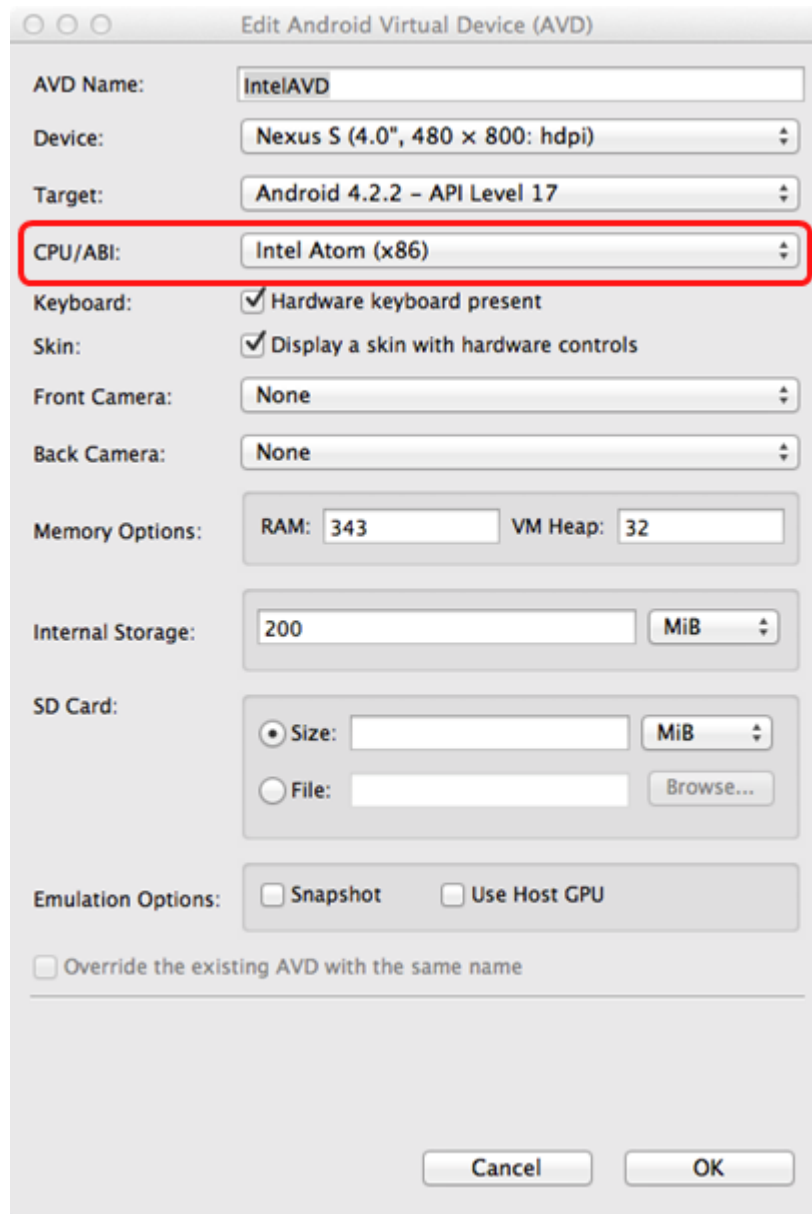
The Intel HAXM only functions in combination with one of the Intel Atom processor x86 system images, which are available for Android 2.3.3 (API 10), 4.0.3 (API 15), 4.1.2 (API 16), 4.2.2 (API 17). These system images can be installed exactly like the ARM-based images through the Android SDK Manager.

Figure 2-4 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 2-5](#)).

Note:

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

Figure 2-5 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 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)*.

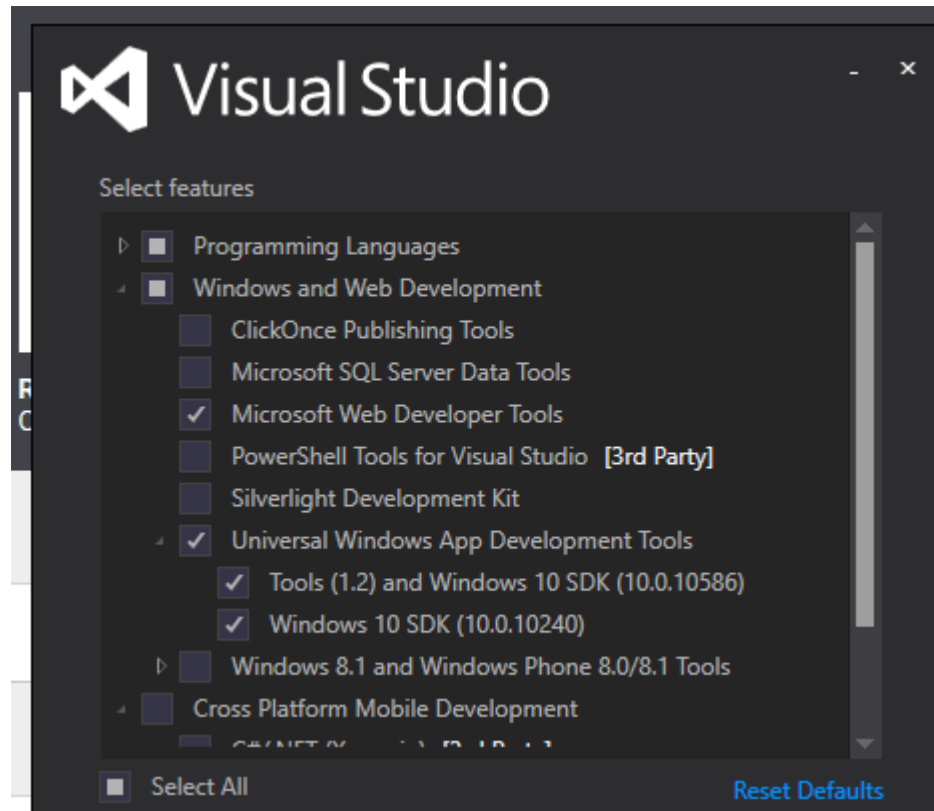
2.6.1 How to Install Visual Studio

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.

To install Visual Studio

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.
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 2-6](#).

Figure 2-6 *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 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
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.

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.

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

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 set the environment following the instructions in [Configuring the Environment for Target Platforms](#) correctly the dialog will show the correct target, as shown in [Figure 2-7](#).

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 the adb daemon.

To kill the process, use:

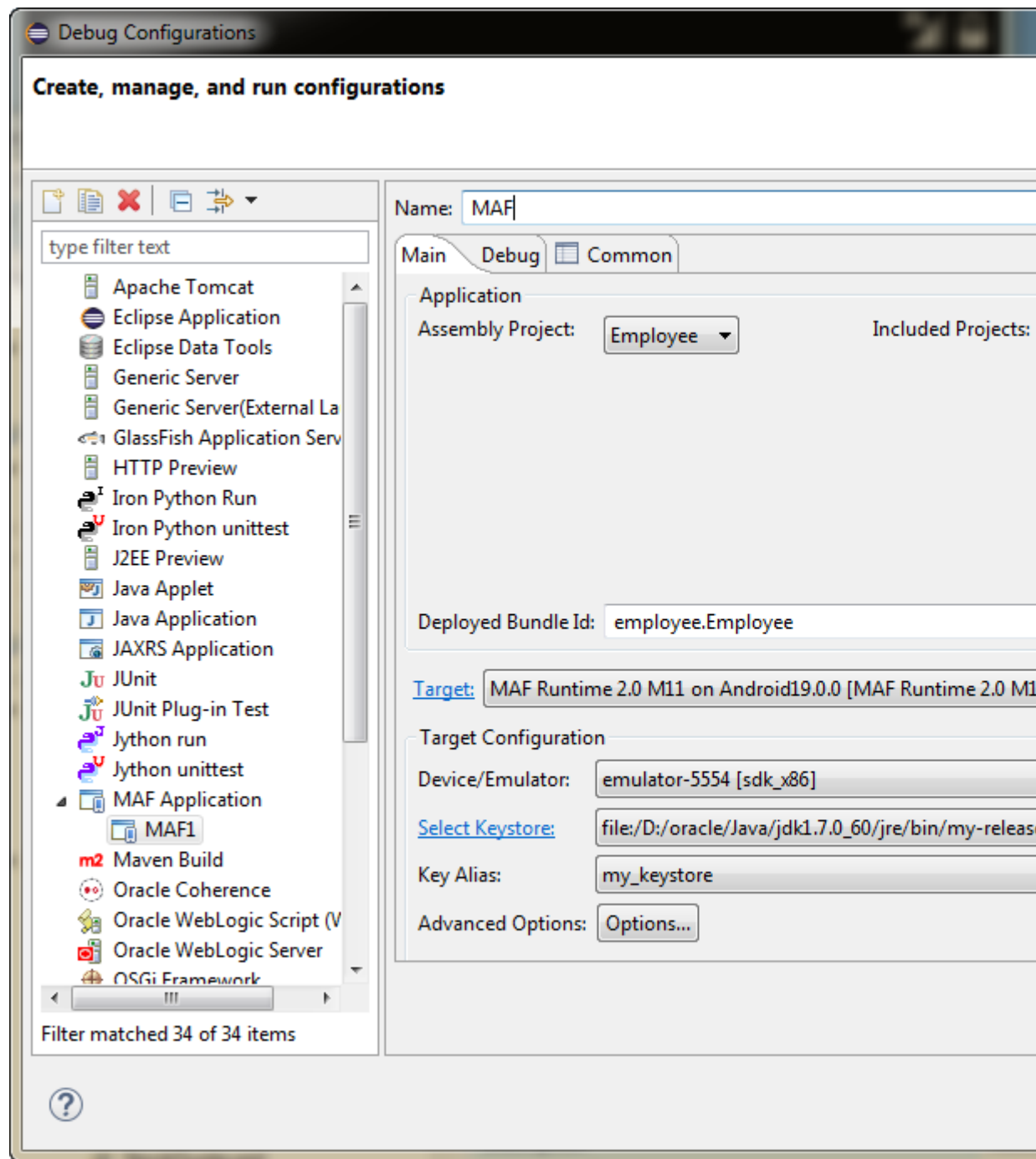
- Windows: use the process manager
- Mac terminal: use the `kill -9 PROCID` command

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 2-7 Debug Configurations Dialog



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

- How to Deploy an iOS Application to an iOS Simulator
- How to Deploy an Android Application to an Android Emulator

See the Deploying Mobile 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.

Migrating Your Application to MAF 2.3.2

This chapter provides information about migrating applications created using earlier releases of MAF to MAF 2.3.2.

This chapter includes the following sections:

- [Migrating an Application to MAF 2.3.2](#)
- [Configuring Application Features with AMX Content to Use WKWebView on iOS 9](#)
- [Migrating Cordova Plugins from Earlier Releases to MAF 2.3.2](#)
- [Security Changes in Release 2.2.1 and Later of MAF](#)
- [Maintaining Separate Xcode Installations for MAF 2.3.2 and MAF 2.2.0](#)
- [Migrating MAF Applications that Use Customer URL Schemes to Invoke Other Applications](#)
- [Migrating to JDK 8 in MAF 2.3.2](#)
- [Migrating to a New cacerts File for SSL in MAF 2.3.2](#)

3.1 Migrating an Application to MAF 2.3.2

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 replaces the `java.security` file in your migrated MAF application with a new version generated by this release of 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.

In this release, you can also 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](#).

- 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. For more information, see [Migrating Cordova Plugins from Earlier Releases to MAF 2.3.2](#).
- 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 the "Creating a Rest Service Adapter to Access Web Services" section

in *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. For more information, see the "Using Web Services in a MAF Application" section in *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 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 (OEPE Edition)*.

MAF enables App Transport Security (ATS) by default for applications that you migrate to this release. For more information, 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.

For more information, see the "Specifying REST Service Connections" section 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 more information about how to connect to the SQLite database, see the "Using the Local SQLite Database" section in *Developing Mobile Applications with Oracle Mobile Application Framework (OEPE Edition)*.

3.2 Configuring Application Features with AMX Content to Use WKWebView on iOS 9

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="wkpl">
    <adfmf:property id="wkpl-1" name="iOSWebView" value="modern" />
    <!-- To revert to using UIWebView, set to legacy -->
    <!-- name="iOSWebView" value="legacy" -->
  </adfmf:properties>
</adfmf:feature>
```

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

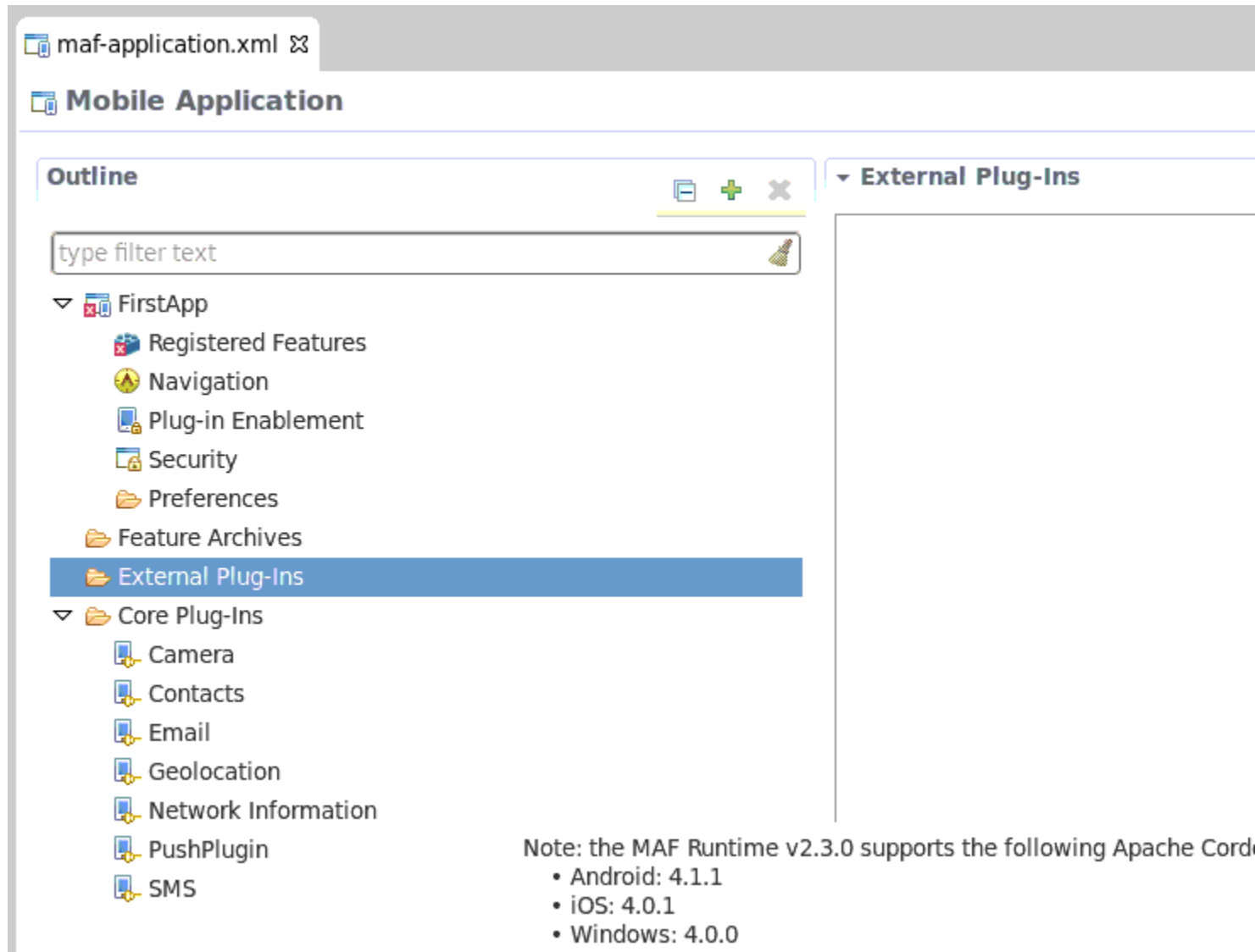
When the `iOSWebView` property is missing or is set to `default` then WKWebView is used for AMX content and UIWebView is used for local HTML and remote URL content types. You can specifically opt-in to using WKWebView for the local HTML and remote URL content types by setting the value to `modern` if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

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

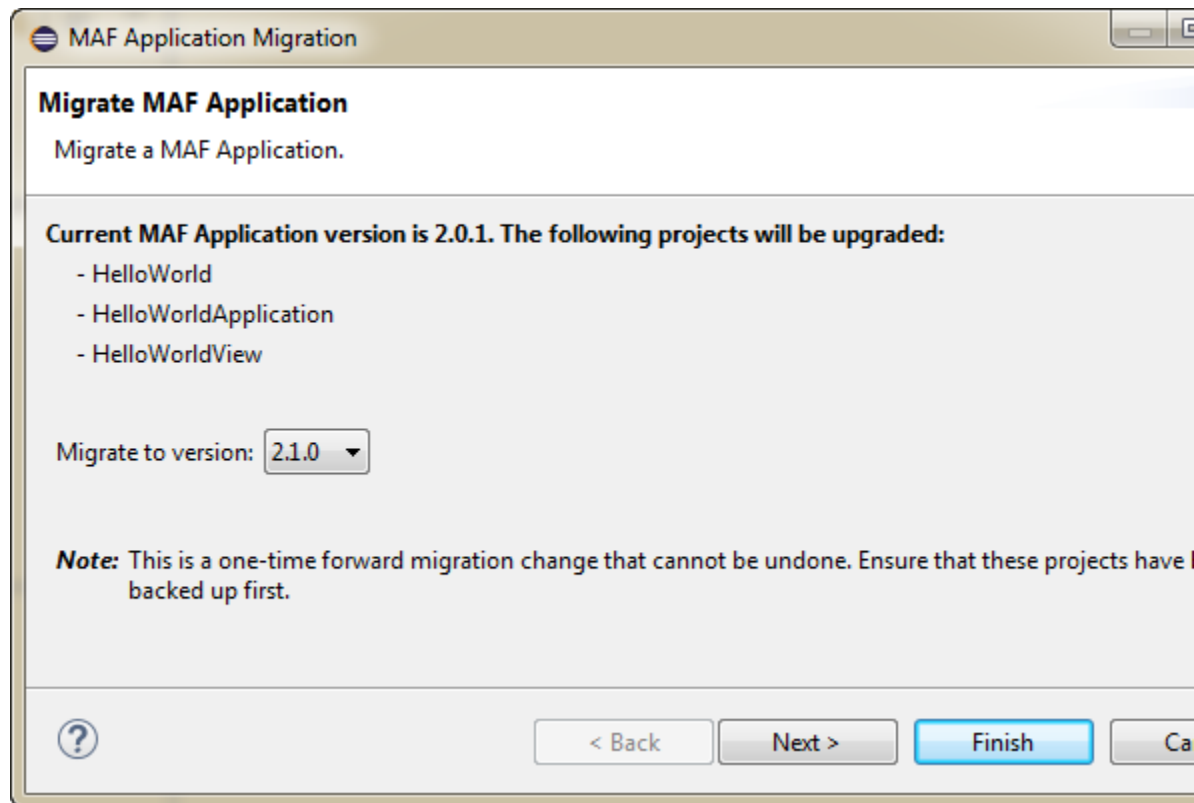
3.3 Migrating Cordova Plugins from Earlier Releases to MAF 2.3.2

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



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

3.3.1 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**.

3.3.2 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. For more information, 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.

3.4 Security Changes in Release 2.2.1 and Later of MAF

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.

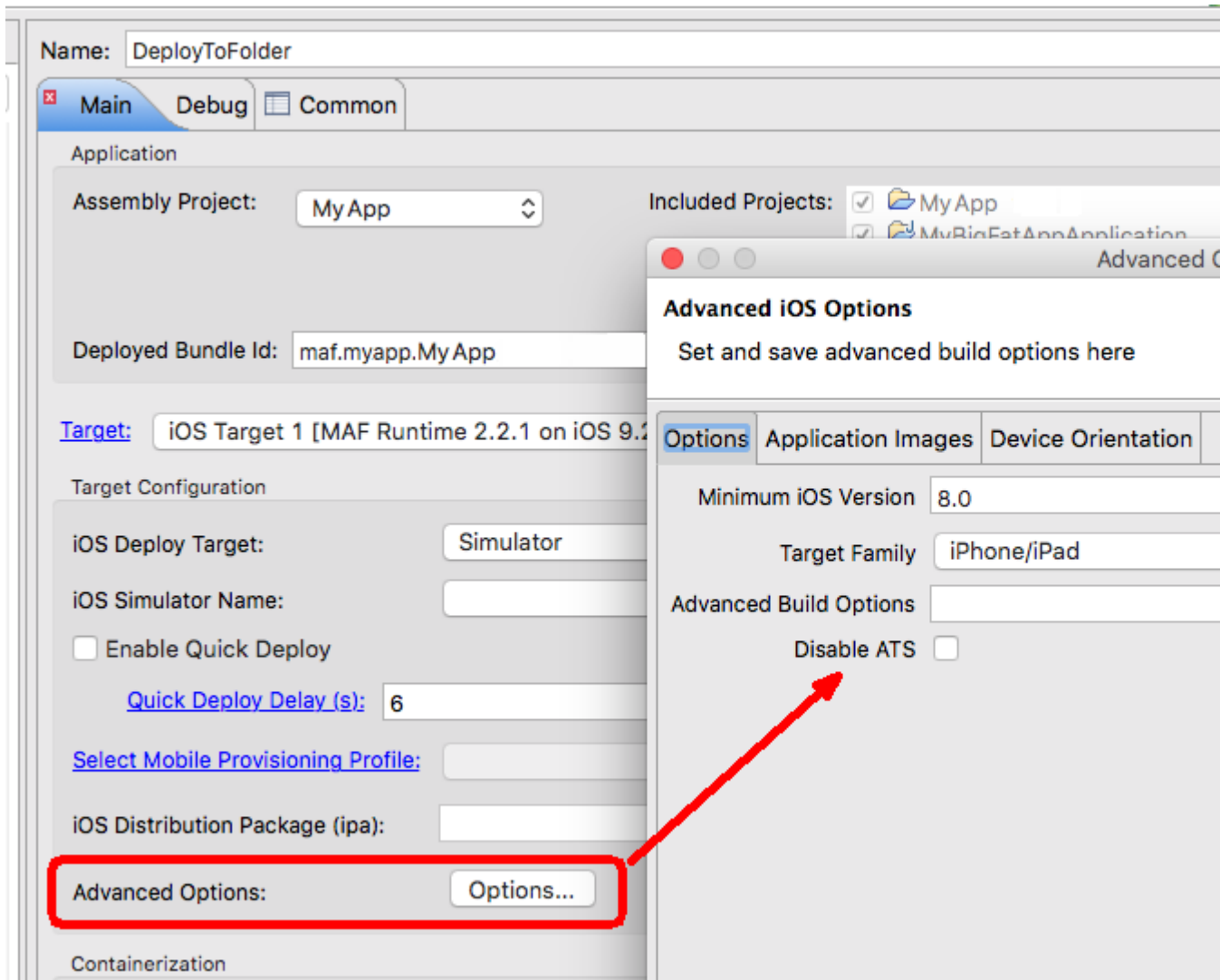
MAF applications also adhere to the default behavior enforced by Java 8's JVM to use the latest SSL version and cipher suites. While we encourage you to upgrade your servers to use these later versions, you can configure your MAF application to work around SSL errors you may encounter by using servers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

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
```

3.5 Maintaining Separate Xcode Installations for MAF 2.3.2 and MAF 2.2.0

MAF 2.3.0 and later requires Xcode 7.x. If you want to maintain one development environment only (for MAF 2.3.0 and later), install or upgrade to Xcode 7.x, as described in [How to Install Xcode and iOS SDK](#). Once you install or upgrade to Xcode 7.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 2.3.2 application to iOS. With this installation, Xcode 7.x replaces Xcode 6.x. No other changes are required. OEPE uses the active Xcode installation.

If you want to maintain separate development environments for MAF 2.3.2 (using Xcode 7.x) and MAF 2.2.0 or earlier (using Xcode 6.x), you can install both Xcode 6.x and Xcode 7.x on the same machine. See the following procedure for information about how you can accomplish this task.

For a complete list of supported versions of development and runtime tools, see the 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/>.

To maintain separate Xcode 7.x and Xcode 6.x installations:

1. Rename the preexisting Xcode.app installation for Xcode 6.x (For example, `Xcode6.app`.)
2. Install Xcode 7.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 7.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 7.x installation:
/Applications/Xcode.app
```

```
Xcode 6.x installation:
/Applications/Xcode6.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 7.x active:
sudo xcode-select -s /Applications/Xcode.app

//To make Xcode 6 active:
sudo xcode-select -s /Applications/Xcode6.app

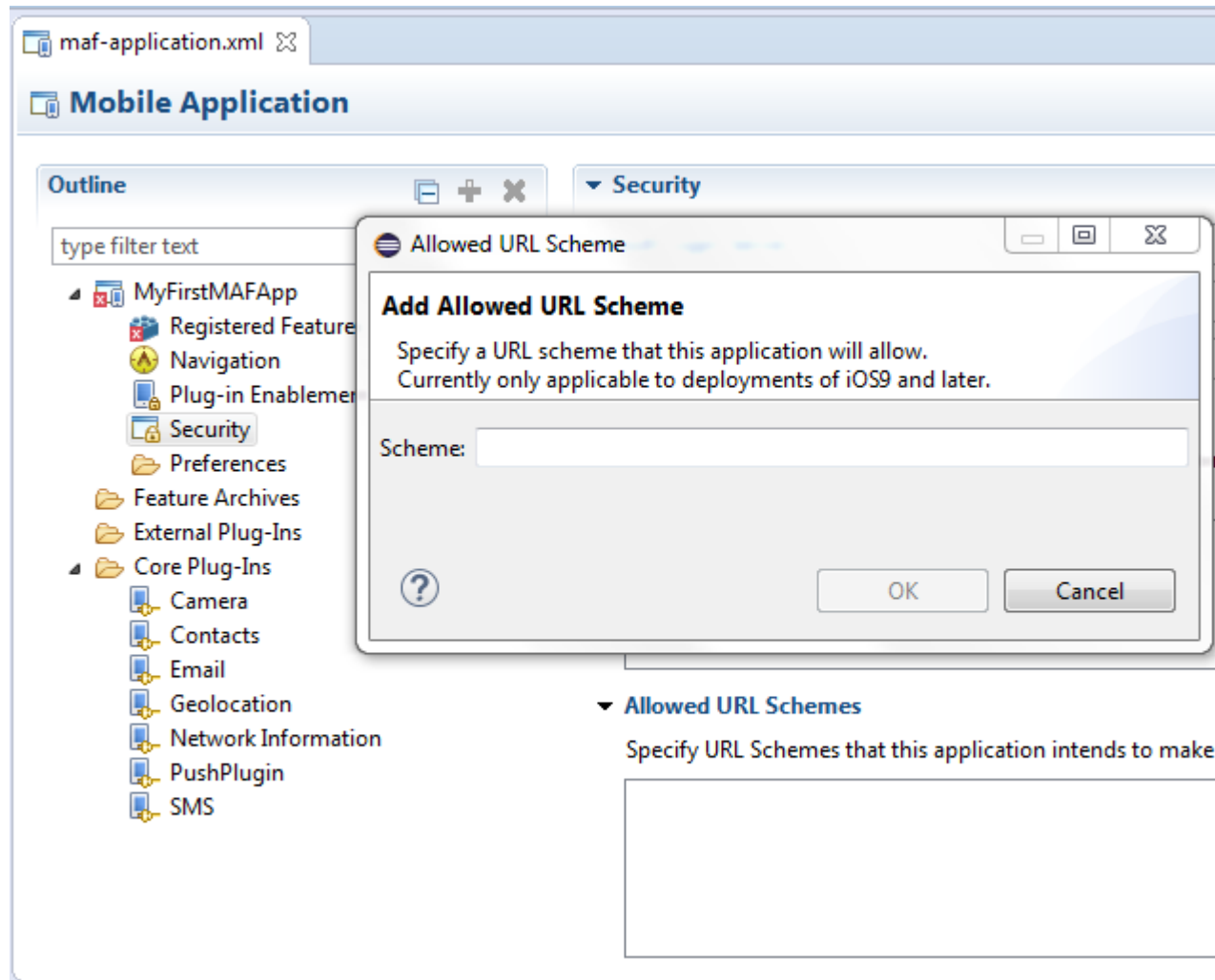
//To determine which instance of Xcode is currently active:
xcode-select --print-path
```

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

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



3.7 Migrating to JDK 8 in MAF 2.3.2

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 2.3.2 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 more 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 more 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 2.3.2

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 more 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 more 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

