Oracle® Retail Store Inventory Management

Mobile Application Framework (MAF) Guide
Release 16.0
E76226-01

December 2016

ORACLE

Oracle Retail Store Inventory Management Mobile Application Framework (MAF) Guide, Release 16.0

E76226-01

Primary Author: Bernadette Goodman, Tracy Gunston

Contributors: Bipin Pradhan

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

(i) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(ii) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(iii) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(iv) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

Send Us Your Comments

Oracle Retail Store Inventory Management Mobile Application Framework (MAF) Guide, Release 16.0

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

Preface

This Installation Guide describes the requirements and procedures to install this Oracle Retail Store Inventory Management (SIM) Mobile Application Framework (MAF) release.

Audience

This Installation Guide is written for administrators who help push Oracle Retail SIM MAF to the App Store.

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 in the Oracle Retail Store Inventory Management Release 16.0 documentation set:

- *Oracle Retail Store Inventory Management Batch Operations Guide*
- *Oracle Retail Store Inventory Management Configuration Guide*
- *Oracle Retail Store Inventory Management Extension Guide*
- *Oracle Retail Store Inventory Management Installation Guide*
- *Oracle Retail Store Inventory Management Integration Guide*
- *Oracle Retail Store Inventory Management Release Notes*
- *Oracle Retail Store Inventory Management Security Guide*
- *Oracle Retail Store Inventory Management Store User Guide*
- *Oracle Retail Store Inventory Management MAF Installation Guide*
- *Oracle Retail Store Inventory Management Wavelink Studio Client Guide*
- *Oracle Retail Store Inventory Management Data Model*
- *Oracle Retail Store Inventory Management Upgrade Guide*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create

- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 16.0) or a later patch release (for example, 16.0.1). If you are installing the base release or additional patch releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times **not** be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-**02** is an updated version of a document with part number E123456-**01**.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Technology Network

Oracle Retail product documentation is available on the following web site:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

(Data Model documents are not available through Oracle Technology Network. You can obtain them through My Oracle Support.)

Conventions

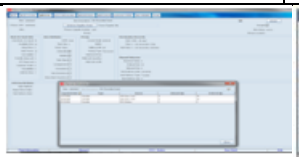
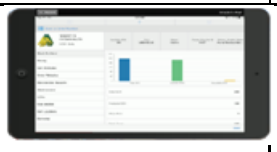
Navigate: This is a navigate statement. It tells you how to get to the start of the procedure and ends with a screen shot of the starting point and the statement "the Window Name window opens."

```
This is a code sample
    It is used to display examples of code
```

1

Overview

SIM UI Clients

	Desktop	Wavelink HH	MAF UI - Large	MAF UI - small
				
Technical	Java SWING	Wavelink Studio	Oracle - Mobile Application Framework	Oracle - Mobile Application Framework
Operating systems supported	Windows Desktop Version	Windows Mobile/CE, iOS, Android (Velocity client)	iOS, Android, Windows 10	iOS, Android, Windows 10
Device	PC/desktop, Tablet	Hardened Mobile devices, Tablet	Tablet	Hardened Mobile devices (MC 40/TC 70/and so on)
Used by	Store manager, Store Ops, Inventory Control	Store associates walking the shop floor	Store/district manager	Store associates walking the shop floor
User Interaction	Mouse, keyboard, Wedge Scanner	Sled scanner, touch	Bluetooth/Wedge scanner, touch	Bluetooth/Wedge scanner, touch, Barcode scanner (MC40)
Form factor - optimized	Large - Extra large (12"+)	Small (4"-5")	Medium - Large (7"+)	Small (4"-6")
Purpose	Admin, transaction approval, transaction management, research	Scanning barcodes, recording transaction information, mobility	Manager dashboard view, mobility	Scanning barcodes, recording transaction information, mobility

This document will refer to MAF (Mobile Application Framework) portion.

This chapter describes the architecture of SIM MAF UI.

Architecture

The SIM Mobile application is built using Oracle Mobile Application Framework (MAF). MAF is a cross-platform framework that uses web technologies (HTML5 and CSS) for the User Interface (UI), Java for application business logic, and Apache Cordova for access to the device features.

MAF defines a feature as a reusable, self-contained module of application functionality. A MAF feature can use one of three content types: AMX, Local HTML, or Server (Remote) HTML. The SIM feature is built using AMX.

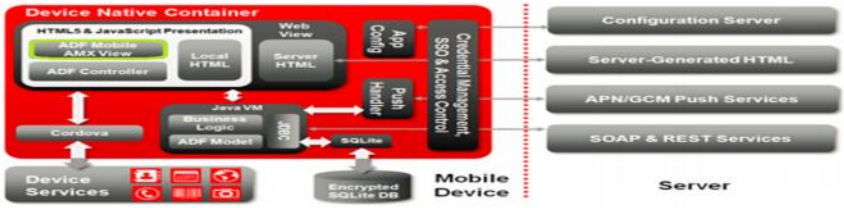
For an introduction to MAF, see Section 1.1, Introduction to Mobile Application Framework, in Developing Mobile Applications with Oracle Mobile Application Framework on the Oracle Help Center:

https://docs.oracle.com/middleware/maf212/mobile/develop-maf/maf-about.htm#ADFMF1150

Runtime Architecture

The runtime architecture of the SIM Mobile application is provided by MAF. For an overview of the runtime architecture of MAF, see Developing Mobile Applications with Oracle Mobile Application Framework on the Oracle Help Center.

Runtime Architecture



Security

Security is a top priority for mobile application development given that mobile devices are at a higher risk of loss or theft. For more information, see the *Oracle Retail Store Inventory Management Security Guide*.

Authentication

SIM Mobile uses MAF-based security to authenticate users to log in and view application features. MAF determines whether access to the application feature requires user authentication when an application feature is secured by an authentication server.

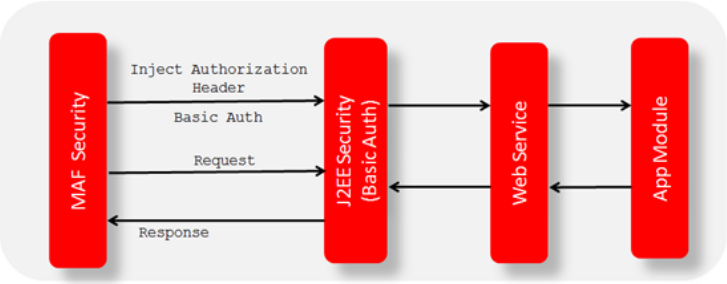
MAF supports the following authentication modes: Basic Auth, OAuth, and Web Single Sign On (SSO). MAF applications can use either the default login page provided by MAF or a customized login page that is written in HTML.

Service Authentication and Authorization

RESTful web services are secured using a basic authentication mechanism and access controlled using the standard J2EE authorization model. For more information on the J2EE authorization model, see the following web site:

http://docs.oracle.com/cd/E28280_01/web.1111/e13711/thin_client.htm#SCPRG133

Service Authentication and Authorization



Authorization

SIM Mobile implements similar access control functionality as other SIM platforms. Users' permissions are based upon assigned roles.

Deployment

> **Note**: The Basic Authentication application and Access Control Service are deployed together in the same EAR, but only one instance of the Access Control Service is used by SIM at runtime. For more information, see Application Configuration.

Service Endpoint Assembly:

SIM services are assembled as part of the application EAR file.

Typically, the services used are deployed on separate domains. It is also valid for the services to be deployed on a single domain.

2

Preinstallation Tasks

Technical Concepts

The installer should understand the following technical concepts:

- JDeveloper 12.2.1
- Application servers
- XML manipulation
- Apple Enterprise Development setup and deployment
- Universal Windows Platform development setup and deployment
- Android platform development and deployment
- Certificate creation and deployment

Technical Specifications

Runtime Support and Certification Matrix

A matrix of MAF certified/supported devices can be found here:
http://www.oracle.com/technetwork/developer-tools/maf/documentation/maf232certmatrix-3112841.html

RBGU Certified Devices

Mobile Operating System	Version	RGBU Certified Devices
iOS	9.x	▪ iPod Touch (5th gen) ▪ iPad Mini (1st gen)
iOS	8.x	▪ iPod Touch (5th gen) ▪ iPad Mini (1st gen)
Android	4.x	▪ Motorola MC40 ▪ Motorola TC70 ▪ Motorola TC55
Windows	10	▪ x64 Desktop

3

Installing and Setting up the Application

This chapter describes how to install and configure JDeveloper and SIM Mobile for deployment.

Prerequisites

Deploying SIM Mobile requires JDeveloper 12.2.1, as well as a few platform specific requirements:

Platform	Prerequisites
Android	▪ Android SDK ▪ .keystore file for App signing
iOS	▪ XCode 7.3.x ▪ Provisioning files and certificates ▪ For more Apple documentation see: https://developer.apple.com/
Universal Windows Platform	▪ Visual Studio Community Edition 2015 with other required components as specified by: http://docs.oracle.com/middleware/maf230/mobile/install/GUID-02517E1F-B7D9-442D-8A0B-2C97B912A966.htm#GUID-6A649FED-6941-4DD9-BA60-6EA6DAF0E82B ▪ .pfx file created and installed

Extracting Installer Artifacts

The artifacts required for installing Oracle Retail SIM Mobile are contained in the sim-all-src.zip/sim-mobile-client-amx-all-src.jar file. Unzip this file to find the following:

File	Contents
maf-2.2000.20151010-0201-RELEASE.zip	This is the MAF extension that must be installed in JDeveloper (see "Installing the MAF extension" below).
SimMobileArchive.maa	The main application archive. (Required to install or customize SIM Mobile)
PlatformMobileArchive.maa	The mobile platform archive. (Required for some customizations.)

Installing the MAF Extension

To install the MAF extension on already installed JDeveloper, see Installing Mobile Application Framework with JDeveloper.

When you reach "Installing the MAF Extension in JDeveloper" step 2, click the "Install from Local File" radio button and then locate and select the maf-2.2000.20151010-0201-RELEASE.zip file (from "Extracting Installer artifacts" above). Then skip to step 6.

Configuring JDeveloper for Deployment

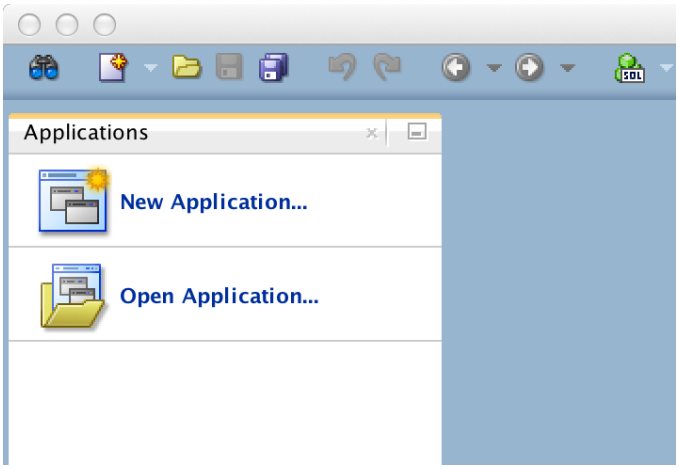
The configuration of JDeveloper is different depending on which platform(s) SIM Mobile is to be deployed on. Oracle MAF documentation details steps on setting up the various prerequisites and configurations based on target platform.

Target Platform	MAF Documentation
Android	http://docs.oracle.com/middleware/maf230/mobile/install/GUID-02517E1F-B7D9-442D-8A0B-2C97B912A966.htm#MAFIG161
Apple iOS	http://docs.oracle.com/middleware/maf230/mobile/install/GUID-02517E1F-B7D9-442D-8A0B-2C97B912A966.htm#MAFIG1397
Universal Windows Platform (Windows 10)	http://docs.oracle.com/middleware/maf230/mobile/install/GUID-02517E1F-B7D9-442D-8A0B-2C97B912A966.htm#MAFIG-GUID-6A649FED-6941-4DD9-BA60-6EA6DAF0E82B

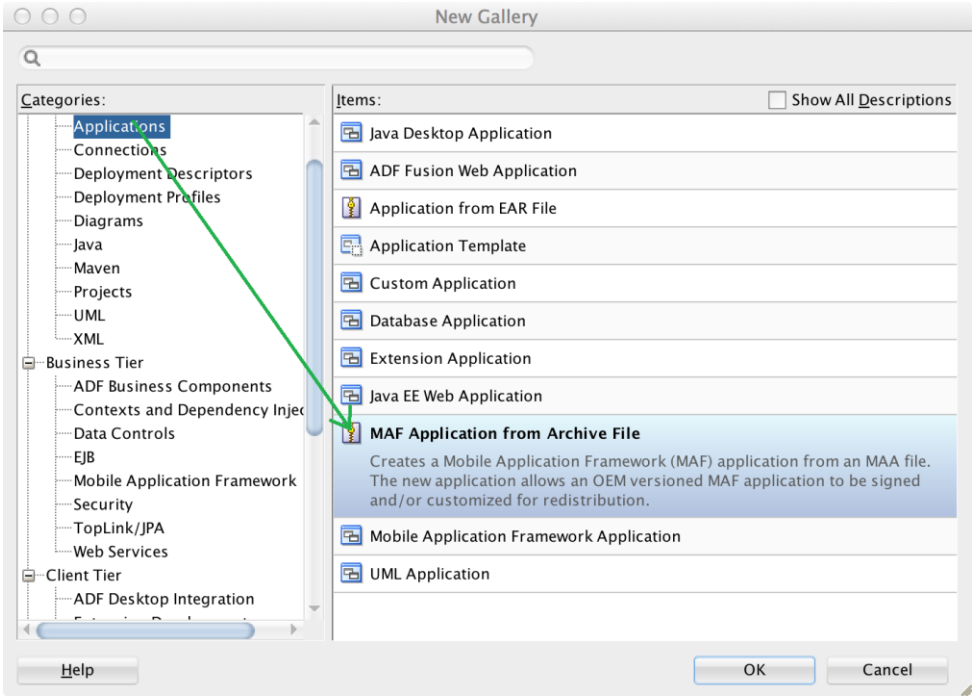
Creating a Workspace from the Archive

When you launch JDeveloper for the first time, you will encounter an empty workspace. The first thing that must be done is to create a new application from the provided .maa file. Use the steps that follow:

1. Open the **New Application** dialog (or open the general **New** dialog and select **Applications**).



2. Select **MAF Application from Archive File** and click **OK**.



3. In the wizard that opens, select the .maa file you want to create a custom application from (the main SIM Mobile app is packaged as SimMobileArchive.maa). For instructions on how to retrieve SimMobileArchive.maa, see section Extracting Installer Artifacts.
4. Enter a name for your application.
5. You can also change where the application will be saved (by default, <workspace>/<application name>).
6. On the Location page, enter the location information, for example:
   - MAA file : /Users/Myuser/Desktop/SimMobileArchive.maa
   - Application file : SimMobileCustom
   - Directory: /Users/Myuser/jDeveloper/mywork/SimMobileCustom
7. Click **Finish** when you have entered the necessary information.

Configuring Application Settings

This section describes the configuration that is required (and some that is optional) prior to building and deploying the SIM Mobile application.

Connection Configuration

Configuring the application includes allowing the application to authenticate users, connect to web services, and in certain cases, access remote images.

Feature	Connection (Type)
Configuration	ConfigServiceLogin (Login) ConfigService (URL)
SIM Mobile	SimMobileLogin (Login) SimMobileService (URL)

SIM Mobile is delivered with placeholder connections for each of the connections listed above. It is possible to build and install the application without updating the included connections and then use the Configuration feature to update the connections after the application is installed on a device. For more information, see Configuration Service Setup.

Another option is to update the placeholder connections with valid URLs prior to building the application so that the application is ready to run immediately after installation on a device. In either case, it is always possible to update the connections used on the device at a later time through the Configuration feature.

Timeout Configuration

Each connection contains an idle and session timeout values in seconds.

The **idle timeout** will log out the user after the configured amount of seconds has passed with zero activity.

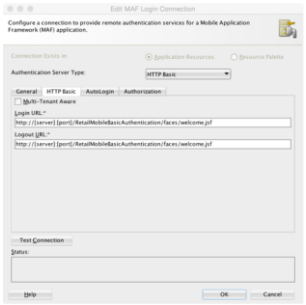
The **session timeout** will log out the user after the configured amount of seconds has passed.

> **Note:** The session timeout must be configured to a value less than that of the server.

Security Configuration

The Authorization feature uses the LoginServer connection and must be configured for HTTP Basic authentication. Configure HTTP Basic authentication on the HTTP Basic tab of the Edit MAF Login Connection window by replacing [server]:[port] with the correct values for your environment.

Edit MAF Login Connection Page HTTP Basic Tab



For more information on accessing and configuring login connections, see the Configuring MAF Connections section in the MAF documentation available at the following web site:

http://docs.oracle.com/middleware/maf212/mobile/develop-maf/maf-securing.htm#ADFMF1024

> **Note:** These updates must be made prior to building the application and cannot be updated through the Configuration feature.

Adding Remote Image URLs to the Application Whitelist

The SIM Mobile feature may show images on several screens. By default, these are read from the SIM Managed Server or from the device itself. Any customization of images from remote URLs must contain references in maf-application.xml to the remote domain.

For more information on adding domains to the whitelist in maf-application.xml, see the section How to Create a Whitelist (or Restrict a Domain) in Developing Mobile Applications with Oracle Mobile Application Framework for MAF 2.1.2.0.0 available at the following web site:

http://docs.oracle.com/middleware/maf212/mobile/develop-maf/maf-ui-remote-url.htm#BABHEHDE

Configuration Service Setup

If you are using the Configuration feature to update the connections.xml file on a mobile device after the application has been installed, it is necessary to host the connections.xml file at a secured location (HTTP Basic authentication).

The hosted connections.xml file should contain valid URLs for all connections being used by the application (including the ConfigService and ConfigServiceLogin connections). Some connections may not be in use (see above).

The connections.xml file must be named connections.xml and be located at a path that ends with the application identifier. For example, if the application identifier is com.company.SIM, the connections.xml could be located at the following URL:

http://server:port/SomeLocation/com.company.SIM/connections.xml

You need two URLs to complete the configuration process:

- Configuration URL: This is the base URL up to, but not including, the application identifier. For example:

http://server:port/SomeLocation

(Note that this is also the value that can be pre-populated in the ConfigService connection.)

- Configuration Login URL: This is a complete URL to a secured resource on the same domain as the connections.xml file. For example:

http://server:port/SomeLocation/SomeSecuredResource

(Note that this is also the value that can be pre-populated in the ConfigServiceLogin connection.)

If the application is deployed with placeholder (non-valid URLs) in the connections.xml file and the user is required to use the Configuration feature to get valid connections, it is recommended that you configure the application to prompt the user to set up configuration on the initial launch of the application (until configuration has been completed).

To configure the application to prompt the user:

1. Locate and open the adf-config.xml file in JDeveloper under Application Resources > Descriptors > ADF META-INF.
2. In the source view, locate the adf-property tag with the name value RETAIL_INITIAL_CONFIGURATION_REQUIRED.
3. Change the value to true.

Navigation Configuration

> **Note:** There is no need to update the navigation configuration unless changes in organization, labeling, and so on are desired.

The springboard is configured by a navigation.json file in the application controller project. The file has the following structure:

```
{
    bundles : {
        <alias> : <full bundle name>,
        ...
    },
    menus : {
        name : <Application name>,
        options : [
            {
                name : <Feature group name>,
                options : [
                    {
                        name : <Feature name>,
                        id : <ID of feature to link to>,
                        options : <Optional submenu>
                    },
                ...
                ]
            },
            ...
        ]
    }
    footers : [
        {
            label : <Label for footer item>,
            featureId : <ID of feature to link to>
        },
        ...
    ]
}
```

The bundles section allows the declaration of aliases to XLIFF bundles. Translated strings can then be used for name/label with [Alias.Key], similar to the usage in AMX pages (after using loadBundle). If an alias is not declared, the full bundle path must be used instead of the alias.

The menus section begins with the root node of the hierarchy. The name of this node is displayed on the switcher page of the springboard. The nodes under it are the feature groups to display on the switcher page. Finally, the next level of nodes are the features within the group. If no name is provided, the name of the feature referenced is used. If the options property is specified, another level of feature nodes can be placed in it to form a submenu (the springboard does not support nesting submenus), and name is required in this case. Tapping on a feature item in the group or a submenu allows you to navigate to the provided feature identifier.

The footers section allows for setting up feature links that should be displayed regardless of current feature group. A feature link is included by specifying the featureId and optionally a label.

Configuring Application Deployment

To configure SIM Mobile for deployment, use the following steps:

1. Open the workspace created above in Creating a Workspace from the Archive.
2. Open the application properties (Application → Application Properties…).
3. Deployment profiles are platform specific, and each relevant to your target platform(s) will need to be configured.

Target Platform	Deployment Profile Steps
Android	1. Under Deployment, select the **Android1** deployment profile and click the pencil icon. 2. Confirm that the application deploys in **Release Mode**. 3. Check the other settings as needed. 4. Click **OK**.
Apple iOS	1. Under Deployment, select the **iOS1** deployment profile and click the pencil icon. 2. Select iOS Options and confirm you have the correct Application Bundle Id and Application Archive Name (for more information about application bundle ids, refer to *Oracle Fusion Middleware Developing Mobile Applications with Oracle Mobile Application Framework* Documentation.) 3. Confirm that the application deploys in **Release Mode**. 4. Check the other settings as needed. 5. Click **OK**.
Universal Windows Platform (Windows 10)	1. Under Deployment, select the **Windows1** deployment profile and click the pencil icon. 2. Confirm that the application deploys in **Release Mode**. 3. Check the other settings as needed. 4. Click **OK**.

Deploying Application

Note: If SIM Mobile is already deployed to a device, it is recommended to delete it from the device before redeploying it again.

Details of application deployment can be found in the Oracle MAF documentation: https://docs.oracle.com/middleware/maf232/mobile/develop-maf/GUID-9579F7C9-E696-43A0-8087-8953187A0E81.htm#ADFMF990

To deploy SIM Mobile, use a deployment profile for the respective target platform.

Target Platform	Deployment Profile
Android	Android1
Apple iOS	iOS1
Universal Windows Platform (Windows 10)	Windows1

Note: Production deployments should always be in "Release" build mode with logging level of SEVERE. These are the default settings in the SIM Mobile maa.

4

Installation Order

This section provides a guideline as to the order in which the Oracle Retail applications should be installed. If a retailer has chosen to use some, but not all, of the applications the order is still valid less the applications not being installed.

> **Note:** The installation order is not meant to imply integration between products.

Enterprise Installation Order

1. Oracle Retail Merchandising System (RMS), Oracle Retail Trade Management (RTM)
2. Oracle Retail Sales Audit (ReSA)
3. Oracle Retail Extract, Transform, Load (RETL)
4. Oracle Retail Warehouse Management System (RWMS)
5. Oracle Retail Invoice Matching (ReIM)
6. Oracle Retail Price Management (RPM)
7. Oracle Retail Allocation
8. Oracle Retail Mobile Merchandising (ORMM)
9. Oracle Retail Customer Engagement (ORCE)
10. Oracle Retail Xstore Office
11. Oracle Retail Xstore Point-of-Service, including Xstore Point-of-Service for Grocery, and including Xstore Mobile
12. Oracle Retail Xstore Environment
13. Oracle Retail EFTLink
14. Oracle Retail Store Inventory Management (SIM), including Mobile SIM
15. Oracle Retail Predictive Application Server (RPAS)
16. Oracle Retail Predictive Application Server Batch Script Architecture (RPAS BSA)
17. Oracle Retail Demand Forecasting (RDF)
18. Oracle Retail Category Management Planning and Optimization/Macro Space Optimization (CMPO/MSO)
19. Oracle Retail Replenishment Optimization (RO)
20. Oracle Retail Regular Price Optimization (RPO)
21. Oracle Retail Merchandise Financial Planning (MFP)
22. Oracle Retail Size Profile Optimization (SPO)
23. Oracle Retail Assortment Planning (AP)
24. Oracle Retail Item Planning (IP)
25. Oracle Retail Item Planning Configured for COE (IP COE)
26. Oracle Retail Advanced Inventory Planning (AIP)
27. Oracle Retail Integration Bus (RIB)
28. Oracle Retail Service Backbone (RSB)
29. Oracle Retail Financial Integration (ORFI)

30. Oracle Retail Bulk Data Integration (BDI)
31. Oracle Retail Integration Console (RIC)
32. Oracle Commerce Retail Extension Module (ORXM)
33. Oracle Retail Data Extractor for Merchandising
34. Oracle Retail Clearance Optimization Engine (COE)
35. Oracle Retail Analytic Parameter Calculator for Regular Price Optimization (APC-RPO)
36. Oracle Retail Insights, including Retail Merchandising Insights (previously Retail Merchandising Analytics) and Retail Customer Insights (previously Retail Customer Analytics)
37. Oracle Retail Order Broker

5

Extending SIM MAF UI

Overview

The SIM Mobile application can be customized in a number of ways covered in this chapter:

- Application Branding
- Application Skinning
- Application Localization
- Adding UI Elements
- Removing UI Elements
- Altering Logic Elements

And lastly, features can be added and removed. However, the entire SIM product is defined as a single feature within the application, and the process of adding a new feature requires a full MAF license (as noted later in this chapter). Even though licensing is available, SIM does not suggest adding or removing features from the application, but rather adding and removing workflows within the SIM feature.

Configuration

This section describes the configuration that is required (and some that is optional) prior to building and deploying the SIM Mobile application.

Application Configuration

Configuring the application includes allowing the application to authenticate users, connect to web services, and in certain cases, access remote images.

Feature	Connection (Type)
Configuration	ConfigServiceLogin (Login) ConfigService (URL)
SIM Mobile	SimMobileLogin (Login) SimMobileService (URL)

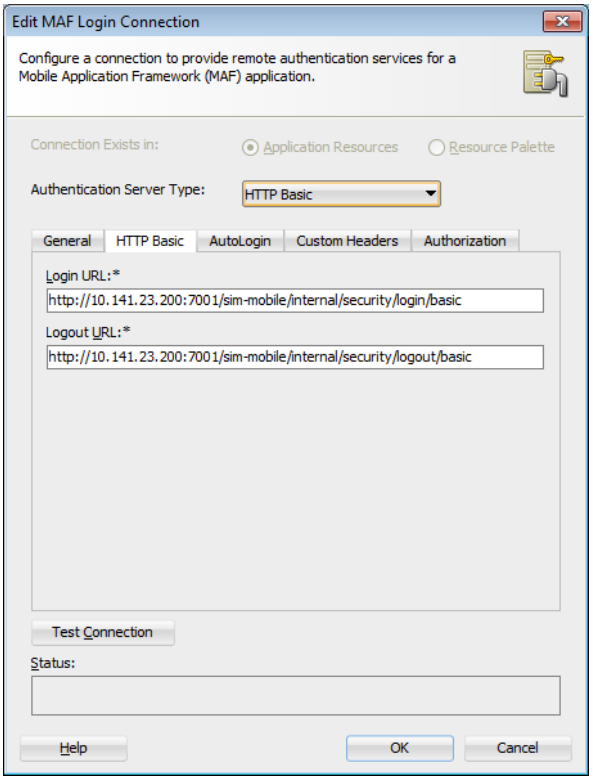
SIM Mobile is delivered with placeholder connections for each of the connections listed above. It is possible to build and install the application without updating the included connections and then use the Configuration feature to update the connections after the application is installed on a device. For more information, see Configuration Service Setup.

Another option is to update the placeholder connections with valid URLs prior to building the application so that the application is ready to run immediately after installation on a device. In either case, it is always possible to update the connections used on the device at a later time through the Configuration feature.

Security Configuration

The Authorization feature uses the LoginServer connection and must be configured for HTTP Basic authentication. Configure HTTP Basic authentication on the HTTP Basic tab of the Edit MAF Login Connection window by replacing [server]:[port] with the correct values for your environment.

Edit MAF Login Connection Page HTTP Basic Tab



For more information on accessing and configuring login connections, see the Configuring MAF Connections section in the MAF documentation available at the following web site:

http://docs.oracle.com/middleware/maf212/mobile/develop-maf/maf-securing.htm#ADFMF1024

> **Note:** These updates must be made prior to building the application and cannot be updated through the Configuration feature.

Adding Remote Image URLs to the Application Whitelist

The SIM Mobile feature may show images on several screens. By default, these are read from the SIM Managed Server or from the device itself. Any customization of images from remote URLs must contain references in maf-application.xml to the remote domain.

For more information on adding domains to the whitelist in maf-application.xml, see the section How to Create a Whitelist (or Restrict a Domain) in Developing Mobile Applications with Oracle Mobile Application Framework for MAF 2.1.2.0.0 available at the following web site:

http://docs.oracle.com/middleware/maf212/mobile/develop-maf/maf-ui-remote-url.htm#BABHEHDE

Configuration Service Setup

If you are using the Configuration feature to update the connections.xml file on a mobile device after the application has been installed, it is necessary to host the connections.xml file at a secured location (HTTP Basic authentication).

The hosted connections.xml file should contain valid URLs for all connections being used by the application (including the ConfigService and ConfigServiceLogin connections). Some connections may not be in use (see above).

The connections.xml file must be named connections.xml and be located at a path that ends with the application identifier. For example, if the application identifier is com.company.SIM, the connections.xml could be located at the following URL:

http://server:port/SomeLocation/com.company.SIM/connections.xml

You need two URLs to complete the configuration process:

- Configuration URL: This is the base URL up to, but not including, the application identifier. For example:

  http://server:port/SomeLocation

(Note that this is also the value that can be pre-populated in the ConfigService connection.)

- Configuration Login URL: This is a complete URL to a secured resource on the same domain as the connections.xml file. For example:

  http://server:port/SomeLocation/SomeSecuredResource

(Note that this is also the value that can be pre-populated in the ConfigServiceLogin connection.)

If the application is deployed with placeholder (non-valid URLs) in the connections.xml file and the user is required to use the Configuration feature to get valid connections, it is recommended that you configure the application to prompt the user to set up configuration on the initial launch of the application (until configuration has been completed).

To configure the application to prompt the user:

1. Locate and open the adf-config.xml file in JDeveloper under Application Resources > Descriptors > ADF META-INF.
2. In the source view, locate the adf-property tag with the name value RETAIL_INITIAL_CONFIGURATION_REQUIRED.
3. Change the value to true.

Navigation Configuration

> **Note:** There is no need to update the navigation configuration unless changes in organization, labeling, and so on are desired.

The springboard is configured by a navigation.json file in the application controller project. The file has the following structure:

```
{
    bundles : {
        <alias> : <full bundle name>,
        ...
    },
    menus : {
        name : <Application name>,
```

```
            options : [
                {
                    name : <Feature group name>,
                    options : [
                        {
                            name : <Feature name>,
                            id : <ID of feature to link to>,
                            options : <Optional submenu>
                        },
                    ...
                    ]
                },
                ...
            ]
        }
        footers : [
            {
                label : <Label for footer item>,
                featureId : <ID of feature to link to>
            },
            ...
        ]
    }
```

The bundles section allows the declaration of aliases to XLIFF bundles. Translated strings can then be used for name/label with [Alias.Key], similar to the usage in AMX pages (after using loadBundle). If an alias is not declared, the full bundle path must be used instead of the alias.

The menus section begins with the root node of the hierarchy. The name of this node is displayed on the switcher page of the springboard. The nodes under it are the feature groups to display on the switcher page. Finally, the next level of nodes are the features within the group. If no name is provided, the name of the feature referenced is used. If the options property is specified, another level of feature nodes can be placed in it to form a submenu (the springboard does not support nesting submenus), and name is required in this case. Tapping on a feature item in the group or a submenu allows you to navigate to the provided feature identifier.

The footers section allows for setting up feature links that should be displayed regardless of current feature group. A feature link is included by specifying the featureId and optionally a label.

User Interface Customization

This section describes the supported customizations of the Oracle Retail SIM Mobile application. Customizations require familiarity with developing mobile applications using Oracle Mobile Application Framework (MAF).

For more information on the specifics about MAF customizations, see Chapter 10 Customizing MAF Application Artifacts with Metadata Services (MDS) available at the following web site:

http://docs.oracle.com/middleware/maf212/mobile/develop-maf/maf-apps-customize-mds.htm#ADFMF25118

Also see Chapter 18, Customizing MAF AMX Application Feature Artifacts, in Developing Mobile Applications with Oracle Mobile Application Framework available at the following web site:

http://docs.oracle.com/middleware/maf212/mobile/develop-maf/maf-mds-ui-components.htm#ADFMF24124

Note: For more information on the scope of customizations allowed under the MAF Foundation license that is provided with Oracle Retail SIM Mobile, see the Restricted Use Licenses chapter in the *Oracle Retail Licensing Guide*.

Understanding Metadata Services

For more information on MDS and the MAF artifacts that can be customized, see Developing Mobile Applications with Oracle Mobile Application Framework available at the following web site:

http://docs.oracle.com/middleware/maf212/mobile/develop-maf/toc.htm

Understanding JDeveloper Roles

The JDeveloper IDE runs using a given role. Usually, developers use the Studio Developer role to develop mobile applications with MAF. However, MDS-based customizations must be done using the Customization Developer role.

The Customization Developer role limits what you can do. For example, source code generally cannot be edited directly and new files cannot be created. When changes are made to files using the IDE, the changes are saved separately from the actual source by MDS. MDS applies the changes on top of the source documents. This allows the customizations to be preserved when the source documents are updated.

Customization Setup

There are several setup steps that must be completed before one can begin customizing the SIM Mobile application with MDS. MDS has a notion of customization layers that must be set up before customization can begin. The supported customization layers must be configured in the CustomizationLayerValues.xml file. Java customization classes must be created for the layers you support. The customization classes need to be added to the SIM Mobile classpath and then referenced in the adf-config.xml file in the SIM Mobile workspace.

For more information, see Developing Mobile Applications with Oracle Mobile Application Framework available at the following web site:

http://docs.oracle.com/middleware/maf212/mobile/develop-maf/toc.htm

Note: Oracle Retail recommends that customization classes be created in a separate JDeveloper workspace that produces a JAR that is consumed by the SIM Mobile workspace. For more information, see Developing Mobile Applications with Oracle Mobile Application Framework.

Customizing the Application Id

You will have to update the Application Id in order to deploy your company's version of the SIM Mobile application. There are two approaches to customizing the Application Id. The recommended approach depends on whether you plan on performing additional customizations or not. If you do not plan on performing additional MDS-based customizations, you can simply update the Application Id in maf-application.xml under the Studio Developer role. Afterwards, update the Application Bundle Id referenced by

the deployment profiles you are using to match your new Application Id (if they do not match already).

> **Note:** If you later decide to upgrade SIM Mobile, you will have to revert your changes to the Application Id in maf-application.xml to properly perform the upgrade. After upgrading the application, you can reapply the change to maf-application.xml and the deployment profiles you are using.

If you do plan on customizing the application, Oracle Retail recommends you customize the Application Id using JDeveloper's Customization Role and MDS. That way, you can upgrade your workspace without having to revert your Application Id change temporarily.

To customize the Application Id:

1. Open JDeveloper using the Customization Developer role.
2. Open the maf-application.xml file.
3. Update the Application Id.
4. Save your changes.
5. After the Application Id has been updated in the maf-application.xml file, you should update any deployment profiles you are using to ensure their Application Bundle Ids match your new Application Id. Upon an upgrade, you may need to reapply your Application Bundle Id to the deployment profiles, since deployment profile changes are not saved by MDS; however, the change made to the Application Id in maf-application.xml does not need to be reapplied since it was saved by MDS.

> **Note:** Always ensure that the Application Id in the maf-application.xml and the Application Bundle Id in your deployment profiles match. They are used for different purposes within MAF applications, but for proper functioning of the SIM Mobile application, they must be set to the same value.

Customizing Application Branding

The application branding consists of the application icon as shown in various contexts and the splash screens when launching the application. The application branding can be customized by replacing the icons and images that are referenced by the application.

The following steps assume that a workspace has already been created from the Oracle Retail SIM Mobile MAA file and that the appropriate-sized icons or images have been added to the workspace created from the MAA file:

1. Open JDeveloper in the Studio Developer role.
2. Follow the steps outlined in Developing Mobile Applications with Oracle Mobile Application Framework for more information on how to configure your custom images.

> **Note:** Since MDS is not aware of changes made to this image configuration, any customizations are overridden by upgrading to a newer version of Oracle Retail SIM Mobile.

Customizing Application Skins

The overall look and feel of the SIM Mobile application is controlled by a skin. Since MAF supports MDS customizations of the maf-skins.xml and maf-config.xml files, it is possible to apply a custom skin to the application. If the SIM Mobile application is upgraded to a newer version at a later date, the skinning customizations are still applied on top of the upgraded version by MDS.

The following steps assume that a workspace has already been created from the Oracle Retail SIM Mobile MAA file:

1. Open JDeveloper in the Studio Developer role.
2. Open the SIM Mobile workspace created from the delivered MAA file.
3. Create a new CSS skin file in the SIMApplicationController project. For more information on how to create CSS skin files and how MAF skinning works, see Chapter 7, Skinning MAF Applications, in Developing Mobile Applications with Oracle Mobile Application Framework.
4. Switch JDeveloper to the Customization Developer role by selecting the Tools menu and then Switch Roles > Customization Developer.
5. When JDeveloper has restarted in the Customization Developer role, open the maf-skins.xml file.
6. In the Structure Pane, right click the adfmf-skins node and select Insert Inside adfmf-skins > skin.
7. Fill in the fields in the popup. Note that the style-sheet-name should reference the CSS skin file created in Step 3.
8. If the skin should be versioned, in the Structure Pane, right click the skin element just created, and select Insert Inside skin > version.
9. Fill out the popup with the skin version information. For more details on configuring skins in maf-skins.xml, see Chapter 7, Skinning MAF Applications, in Developing Mobile Applications with Oracle Mobile Application Framework.
10. Save the changes.
11. Open the maf-config.xml file.
12. Update the skin-family element with the name of the custom skin family that should be applied to the entire application.
13. Update the skin-version element if the custom skin was defined with a different version. If the custom skin does not have a version, remove the skin-version element.
14. Save the changes.
15. Deploy the application to iOS Simulator to verify that the new skin is being picked.

After upgrading to a new SIM Mobile version, the custom skin changes are still applied over the base application.

Customizing String Resources

In order to support localization, Oracle Retail SIM Mobile references application string resources in the XLIFF resource bundles.

Since Oracle MAF does not support MDS customizations of XLIFF resource bundles, the Customization Developer role does not allow you to make changes to the XLIFF resource bundles. Any changes made in the Studio Developer role to the XLIFF resource bundles delivered in the SIM Mobile application are overridden when the application is upgraded at a later date, so this approach is not recommended.

In order to add custom string resources to the SIM Mobile application in a future-proof way, you must create new resource bundles under the Studio Developer role, add strings to these new bundles, and then reference strings from these new resource bundles in customizations to AMX pages or other customizable artifacts under the Customization Developer role. For more information about this process, see Enabling Customizations in Resource Bundles in Developing Mobile Applications with Oracle Mobile Application Framework.

Customizing the Application Name

The application name cannot be customized, so for most use cases, it is not recommended to create a new application-level resource bundle as described in the MAF documentation. Instead, create new resource bundles in the projects where you want to add your custom strings.

Customizing Application Strings in the SIM Mobile Feature

The SimMobileViewController project contains the string resources used by the SIM feature. If you want to change strings used in the feature, create a new resource bundle (while in the Studio Developer role) under the SimMobileViewController project, add your custom strings to it, and then in the JDeveloper Customization Developer role, customize the files to reference these new strings.

Springboard Navigation

The SimMobileApplicationController project contains application name string resources referenced by the application Springboard. They are referenced in the navigation.json file. To customize the name of features as they show up on the Springboard, create a new resource bundle (while in the Studio Developer role) under the SimMobileApplicationController project, add your custom strings to it, and then update the navigation.json file to reference your new strings.

To update the navigation.json file, you must first add a new property that maps the key you decide to use for your new resource bundle's basename in the bundles object. For example, if you created a new bundle whose basename is customer.custom.bundle.CustomBundle, you would add a property key that maps to a bundle basename:

customBundle : customer.custom.bundle.CustomBundle

The complete bundles object may look like the following example:

```
"bundles" : {
"SimMobileApplicationControllerBundle" :
"oracle.retail.sim.mobile.client.application.SimMobileApplicationC
ontrollerBundle",
"simmobileviewcontrollerBundle" :
 "oracle.retail.sim.mobile.client.SimMobileViewControllerBundle"
}
```

Next, you need to update the name attribute of the application name you plan to update. If the new customBundle contains the string under the ID NEW_APP_NAME, the reference would follow this format: [customBundle.NEW_APP_NAME] where customBundle is the reference to the resource bundle you defined in the bundles.

Note that the navigation.json file must be updated under the Studio Developer role since you cannot modify this file under the Customization Developer role. Additionally, the customizations to the navigation.json are not managed by MDS, so they are lost whenever you upgrade the SIM Mobile application.

Application Level

The SimMobileViewController project contains the string resources used by the SIM Mobile application for some of its application-level features such as the About page. To add your own custom strings to these features, create a new resource bundle (while in the Studio Developer role) under the SimMobileViewController project, add your custom strings to it, and then in the JDeveloper Customization Developer role, customize the files in the project to reference these new strings.

> **Note:** Some strings displayed on the UI may not be customizable through MDS since the data could originate from web services or be programmatically constructed.

Removing Features from the Application

Oracle Retail SIM Mobile can be customized to remove features. Features that have been removed are not accessible by users when the application is deployed.

To remove features from the application:

1. Open the SIM Mobile workspace under the Customization Developer role.
2. Open the maf-application.xml file.
3. Select the Feature References tab.
4. Select the feature that you wish to remove from the application.
5. Click the X icon just above the table of features to delete the selected feature.
6. Save your changes.

> **Note:** Any updates to the navigation.json file are overridden by SIM Mobile upgrades.

Adding New Features to the Application

Oracle Retail SIM Mobile can be customized to add new custom features. The recommended approach to adding new features to SIM Mobile is to develop the feature in a separate workspace, deploy a Feature Archive (FAR) file containing the feature, and add the FAR as a library to the SIM Mobile application.

For more information on FARs, see Reusing MAF Application Content of Developing Mobile Applications with Oracle Mobile Application Framework available at the following web site:

http://docs.oracle.com/middleware/maf212/mobile/develop-maf/maf-apps-feature-archive.htm#ADFMF25114

> **Note:** Adding new features to the application requires a MAF license. For more information on restricted licenses, see the Restricted Use Licenses chapter of the *Oracle Retail Licensing Guide*.

To add a new feature:

1. Open JDeveloper in Studio Mode.
2. Create a new MAF application for your custom feature development.
3. Develop your custom feature.

4. Test your custom feature by deploying it to an iOS Simulator before adding it to the SIM Mobile application to verify it is working as expected. This is a recommended step.
5. Create a FAR deployment profile as described in the MAF documentation.

   Note: If your feature references connections from the connections.xml file, change the Connections Include option from Connection Name Only to Connection Details (excluding secure content) in the MAF Feature Archive Deployment Profile Properties window. When you later add the FAR you generate to an application, the connection details (that is, URLs) are copied into that application's connections.xml file.

6. Deploy your feature as a FAR.
7. Add the FAR as an application library as described in the MAF documentation to the SIM Mobile workspace.
8. Save your changes.
9. Switch to the JDeveloper Customization Developer role.
10. Open the maf-application.xml file.
11. Add a Feature Reference in maf-application.xml for the feature coming from your FAR.

    Note: Oracle Retail recommends that the platform.mobile.authorization feature remain as the first feature in the list of Feature References.

12. If you want the feature to appear on the springboard, update the navigation.json file to reference it. The order that features appear on the springboard comes from the order they are defined in the navigation.json file. In order to add a reference to your feature on the springboard, insert a JSON object with the following structure to the desired location in the navigation.json file:

    ```
    {"name": "The name of your feature", "id": "the feature ID of your feature."}
    ```

Customizations to the maf-application.xml file are preserved across application upgrades, but changes to the navigation.json file are not. After upgrading, you may need to add the FAR as an application library to the SIM Mobile application again. This again updates any connections in the connections.xml file that may have been removed during the upgrade process.

Customizing the User Interface

Since the SIM Mobile UI is built using MAF artifacts, many of them can be customized. For a full list of components that can be customized, see Chapter 18, Customizing MAF AMX Application Feature Artifacts, available at the following web site:

http://docs.oracle.com/middleware/maf212/mobile/develop-maf/maf-mds-ui-components.htm#ADFMF24124

Following are examples of UI customizations that can be performed:

Adding a New Component to the User Interface

To add a new UI component to the UI:

1. Open the SIM Mobile application under the JDeveloper Customization Developer role.
2. Open the AMX page or page fragment you want to customize with a new UI component.
3. Use one of the standard ways of adding UI Components to the page (right click a component in the Structure pane and select one of the insert options, drag and drop a component from the Component pane to the location in the source where you wish to add the component, and so on). For more information on adding UI Components to a page, see Chapter 13, Creating the MAF AMX User Interface, available in Developing Mobile Applications with Oracle Mobile Application Framework.
4. Save your changes.

Updating Attributes of User Interface Components

If you want to change the text label of a component on the UI or change one of its other attributes:

1. Open the SIM Mobile application under the JDeveloper Customization Developer role.
2. Open the AMX page or page fragment containing the component you wish to customize.
3. Find the component you wish to change in the AMX source code.
4. Click anywhere in the source code of the component you wish to change. The Properties pane updates to display the properties of the selected component.
5. Update the properties in the Properties pane that you wish to change. Note that several key properties appear to be disabled in the Properties pane when under the JDeveloper Customization Developer role (for example, the text property of the amx:commandButton component). Although the property input field is disabled, you can usually still access the menu to the right of the property input field to change its value using the Expression Builder or Select Text Resource options and make updates to the property input field using those UIs.
6. Save your changes.

Removing a Component from the User Interface

If you want to remove a component from the UI (for example, a field on the UI that is not important to your application users):

1. Open the SIM Mobile application under the JDeveloper Customization Developer role.
2. Open the AMX page or page fragment containing the component you wish to remove.
3. Find the component you wish to remove in the Structure pane.
4. Right click the component.
5. Select Delete from the menu that appears.
6. Save your changes.

Customizing Platform Features

Some features in SIM Mobile, such as the application Springboard, come from a FAR file generated from a separate Mobile Platform workspace. In order to customize these Platform features, you must make the customizations in the Platform workspace first. The following steps assume you have created a workspace from the PlatformMobileArchive.maa file and have prepared it for customization:

1. Open JDeveloper in the Customization Developer role.
2. Open the workspace you created from the PlatformMobileArchive.maa file.
3. Make any of the above supported customizations to the features in the PlatformMobileViewController project.
4. Save your changes.
5. Right click the PlatformMobileViewController project.
6. Select Deploy > Platform Mobile Features from the menu.
7. To deploy the feature archive JAR file, click Finish in the Deploy PlatformMobileFeatures window.
8. Overwrite the existing PlatformMobileFeatures.jar file in the lib folder of your SIM Mobile workspace with the PlatformMobileFeatures.jar you just created.
9. Open the SIM Mobile workspace.
10. Deploy the SIM Mobile application.

> **Note:** Whenever you upgrade the SIM Mobile workspace to a newer version, you need to redeploy your customized PlatformMobileFeatures.jar file and overwrite the existing PlatformMobileFeatures.jar file in the lib folder.

Additional Customizations in adf-config

Logic Factories

Logic factories are used to instantiate logic objects that are designed to manage the business logic and functionality supported by the screen. There should be a logic factory in each separate functional area or workflow. Each of these factories allows the user to follow the standard factory pattern within a MAF deployment to sub-class or override all the business logic within SIM that is execute on the MAF client.

Classification of logic objects:

- Reader: Readers object contains all the methods that load information from the server.
- Writer: Writer object contains all the methods that writes information to the server.
- Filter: Filter objects contains all the methods that filter down information from filter bars.
- Model: Model objects contains methods specific to or general about particular business objects. They are mainly used for delegate transaction level APIs.
- Yes/No Handlers: These handlers are used to display a confirmation choice box with buttons to confirm or cancel a particular action.

The following table summarizes the logic factory properties defined in adf-config file:

PROPERTY NAME	DESCRIPTION
ACTIVITY_LOCK _FACTORY_IMPL	Defines custom ActivityLockLogicFactoryInterface implementation
BARCODE_FACTORY_IMPL	Defines custom BarcodeLogicFactoryInterface implementation.
CARTON_LOOKUP _FACTORY_IMPL	Defines custom CartonLookupLogicFactoryInterface implementation.
CORE_LOGIC_FACTORY_IMPL	Defines custom CoreLogicFactoryInterface implementation.
SIM_MONEY_FACTORY_IMPL	Defines custom SimMoneyFactoryInterface implementation.
FULFILLMENT_ORDER _FACTORY_IMPL	Defines custom FulfillmentOrderLogicFactoryInterface implementation.
FULFILLMENT_ORDER_DELIVERY _FACTORY_IMPL	Defines custom FulfillmentOrderDeliveryLogicFactoryInterface implementation.
FULFILLMENT_ORDER_PICK _FACTORY_IMPL	Defines custom FulfillmentOrderPickLogicFactoryInterface implementation.
INVENTORY_ADJUSTMENT _FACTORY_IMPL	Defines custom InventoryAdjustmentLogicFactoryInterface implementation.
ITEM_FACTORY_IMPL	Defines custom ItemLogicFactoryInterface implementation.
LOCATION_FACTORY_IMPL	Defines custom LocationLogicFactoryInterface implementation.
NOTE_FACTORY_IMPL	Defines custom NoteLogicFactoryInterface implementation.
NOTIFICATION_FACTORY_IMPL	Defines custom NotificationLogicFactoryInterface implementation.
OPERATIONAL_VIEW _FACTORY_IMPL	Defines custom OperationalViewLogicFactoryInterface implementation.
PRINT_FACTORY_IMPL	Defines custom PrintLogicFactoryInterface implementation.
QUICK_RECEIVING _FACTORY_IMPL	Defines custom QuickReceivingLogicFactoryInterface implementation.
SCREEN_LOGIC_FACTORY_IMPL	Defines custom ScreenLogicFactoryInterface implementation.
SHIPMENT_FACTORY _IMPL	Defines custom ShipmentLogicFactoryInterface implementation.
STOCK_COUNT_FACTORY_IMPL	Defines custom StockCountLogicFactoryInterface implementation.
STORE_LOOKUP_FACTORY_IMPL	Defines custom StoreLookupLogicFactoryInterface implementation.

PROPERTY NAME	DESCRIPTION
SUPPLIER_LOOKUP _FACTORY_IMPL	Defines custom SupplierLogicFactoryInterface implementation.
TRANSACTION_FACTORY_IMPL	Defines custom TransactionLogicFactoryInterface implementation.
TRANSFER_DELIVERY _FACTORY_IMPL	Defines custom TransferDeliveryLogicFactoryInterface implementation.
TRANSFER_SHIPMENT _FACTORY_IMPL	Defines custom TransferShipmentLogicFactoryInterface implementation.
UIN_FACTORY_IMPL	Defines custom UinLogicFactoryInterface implementation.
UOM_FACTORY_IMPL	Defines custom UomLogicFactoryInterface implementation.

Apart from the logic factories, following are the additional configuration options defined in adf-config file:

PROPERTY NAME	DESCRIPTION
RETAIL_INITIAL_ CONFIGURATION_REQUIRED	Boolean property with "true"/"false" as possible values. If this property is set, the application will request for connection configuration at first run.
RETAIL_NETWORK_CONNECTIVITY_ CHECK_RATE	Number property which defines how often to check if the device is online.
SPRINGBOARD_VALUE_ LOOKUP_INTERVAL	Number property to define how often to refresh springboard entries.
SPRINGBOARD_NAVIGATION_IMPL	To define custom SpringboardNavigation implementation which can be used to define additional springboard entries.
MOBILE_TASK_FLOW_IMPL	To define custom TaskFlow implementation. To be used for adding custom mobile task flows for Springboard entries in the SIM client feature to point at.
MOBILE_SERVICE_FACTORY_IMPL	To define custom MobileServiceFactoryInterface implementation. Service objects are specific to functional areas.
MOBILE_DAO_FACTORY_IMPL	To define custom MobileDaoFactoryInterface implementation.
MOBILE_BO_FACTORY_IMPL	To define custom MboFactoryInterface implementation.
WRAPPER_FACTORY_IMPL	To define custom WrapperFactoryInterface. Wrapper objects are used to display information on screens.
MOBILE_MAPPER_FACTORY_IMPL	To define custom MobileMapperFactoryInterface implementation. Mapper objects are used to do conversion between Business Objects and JSON Objects

PROPERTY NAME	DESCRIPTION
MOBILE_ADDRESS_FACTORY_IMPL	To define custom MobileAddressFactoryInterface implementation
MOBILE_RULE_FACTORY_IMPL	To define custom RuleLogicFactoryInterface implementation.
CURRENCY_DEFAULT_TYPE	To define the default currency type.
BACKGROUND_THREAD_COUNT	To define the background thread count.
DEVICE_CAMERA_SCAN	To enable/disable device camera scan.
SOUND_SCAN_ENABLED	To enable disable sound while scanning.
SOUND_INFORMATION_ENABLED	To enable/disable sound for information messages.
SOUND_WARNING_ENABLED	To enable/disable sound for warning messages.
SOUND_ERROR_ENABLED	To enable/disable sound for error messages.
SCAN_FOCUS_ITEM_DETAIL	To enable/disable auto focus on item detail screen.
VIBRATION_ENABLED	To enable/disable vibration.
REFRESH_RATE_BARCODE_ATTRIBUTES	To define cache length in milliseconds for barcode attributes.
REFRESH_RATE_CONFIG	To define cache length in milliseconds for configuration.
REFRESH_RATE_INV_ADJUST_REASON	To define cache length in milliseconds for inventory adjustment reasons.
REFRESH_RATE_ITEM_DIFF	To define cache length in milliseconds for item diffs.
REFRESH_RATE_ITEM_IMAGE	To define cache length in milliseconds for item images.
REFRESH_RATE_NONSELLABLE_ QTY_TYPE	To define cache length in milliseconds for non sellable quantity types.
REFRESH_RATE_STORE	To define cache length in milliseconds for store.
REFRESH_RATE_STORE_PRINTER	To define cache length in milliseconds for store printers.
REFRESH_RATE_UOM_CONVERSION	To define cache length in milliseconds for uom conversion factors.

Upgrading to a New Version

Customizations are applied on top of the new upgraded version of SIM Mobile. However, the following are examples of updates that need to be reapplied after upgrading:

- The addition of any libraries or JARs to the application-level Libraries and Classpath. Note this includes the following:
  - The JARs containing your customization classes.
  - Any FARs that were added to add new feature content to the SIM Mobile application.

- Any changes to the navigation.json file.
- Any changes to the application branding.
  - Any images you added should be preserved in the upgraded version, however, the application-level configuration to use your new images will be overwritten.
- Any changes to the connections.xml file.
- Any changes made under the Studio Developer directly to delivered resource bundles.

> **Note:** Making these kinds of changes is not recommended since it is not upgrade-safe.

For more information on upgrading a MAF Application that is built from a MAA file such as SIM Mobile Section, see Upgrading a MAF Application with Customizations available at the following web site:

http://docs.oracle.com/middleware/maf212/mobile/develop-maf/maf-apps-customize-mds.htm#ADFMF24079

> **Note:** Although JDeveloper creates a backup copy of the workspace, it is usually a best practice to create your own backup in case the JDeveloper backup fails for some reason.

Unsupported Customizations

The following non-exhaustive list provides some examples of customizations that are not supported:

- Adding new pages to a feature:
  - Although MDS supports customizations to task flows and AMX pages, it currently is not very upgrade-friendly to add new pages to a feature. The Customization Developer role does not allow new pages to be added, so they would have to be added under the Studio Developer role. Upon upgrading, any new pages you have added are preserved. However, adding pages usually changes the DataBindings.cpx file, which is overridden by the upgrade process.
  - The recommended customization is to add a new feature to the application instead of trying to add new pages to a given feature.

> **Note:** Adding new features to the application requires a MAF license. For more information on restricted licenses, see the Restricted Use Licenses chapter of the *Oracle Retail Licensing Guide*.

- Modifying ReST service calls:
  - Similar to modifying business logic, service calls cannot be customized to add additional request parameters, accept different responses, and so on.

MDS Customizations and Configuration Services

If the SIM Mobile application is deployed with configuration services enabled, and you run the configuration services, any changes in subsequent deployments are not picked up by the application at runtime. In order to make sure changes in subsequent deployments of the application are picked up at runtime, first delete the existing

installation of the application on the device before deploying the latest version. (Note that this is necessary only if you run the configuration services.)

JDeveloper Bugs

You many encounter issues with the JDeveloper Customization Developer role. These bugs may interfere with your ability to customize the application. Following are some common issues you may encounter and recommended workarounds:

Unable to Customize the Application

Sometimes the JDeveloper UI indicates that changes can be made to files that support customization, however, it does not seem to respond to changes. For example, sometimes the menu to the right of a field in the Properties pane is unavailable, and you need to access that menu to select a text resource. Usually restarting JDeveloper in the Customization Developer role fixes these kinds of issues.

The Customization Developer role does not let you directly modify source code in the source editor. If you are unable to modify the source code in the source editor, that is the expected behavior.

Exception Stacktraces

At various times during customization (for example, switching between certain files, deployment, and so on), JDeveloper may display an exception stack trace dialog for an error that has occurred. In most cases, the dialog allows you to file a bug for the error or ignore it. Although the error has occurred, usually upon dismissing the window you are able to successfully continue with your customization.

Internationalization

Internationalization is the process of creating software that can be translated more easily. Changes to the code are not specific to any particular market. Oracle Retail applications have been internationalized to support multiple languages.

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated include the following:

- Graphical user interface (GUI)

Error messages

The following components are not translated:

- Documentation (online help, release notes, installation guide, user guide, operations guide)
- Batch programs and messages
- Log files
- Configuration tools
- Reports
- Demonstration data
- Training materials

Translating Text Resources

Oracle Retail SIM Mobile stores text resources in XLIFF (XML Localization Interchange File Format) resource bundles, the standard for Oracle Mobile Application Framework (MAF). For more information about XLIFF, see the following web site:

http://docs.oasis-open.org/xliff/xliff-core/xliff-core.html

See the Developing Mobile Applications with Oracle Mobile Application Framework documentation for MAF's requirements for localized XLIFF files. New localized versions of delivered resources bundles can be added to the workspaces built from the Retail Mobile Platform and SIM Mobile MAA files.

When the SIM Mobile application is deployed to an iOS device, the version of the resource bundles that is used by the application depends on the language set on the device.

Delivered XLIFF Resource Bundles

The following sections describe the XLIFF resource bundles used by the mobile application:

Retail Mobile Resource Bundles

The following table lists the resource bundles that are packaged in the Retail Mobile Platform MAA file:

Base XLIFF Resource Bundle	Text Resource Descriptions
PlatformMobile/PlatformMobileApplication Controller/src/oracle/retail/apps/platform/mobile/PlatformMobileBundle.xlf	Default string resources referenced by Retail Mobile Platform AMX fragments.
PlatformMobile/PlatformMobileViewControl ler/src/oracle/retail/apps/platform/mobile /PlatformMobileViewControllerBundle.xlf	String resources for the Configuration Services feature, the Springboard feature, and so on.

SIM Mobile Platform Resource Bundles

The following table lists the resource bundles that are packaged in the SIM Mobile MAA file:

Base XLIFF Resource Bundle	Text Resource Descriptions
SimMobile/.adf/META-INF/SimMobileBundle.xlf	Text resource for the application name.
SimMobile/SimMobileViewController/src/oracle/retail/sim/mobile/SimMobileViewControllerBundle.xlf	Text resources for the SIM Mobile feature.
SimMobile/SimApplicationController/src/oracle/retail/sim/mobile/client/application/SimMobileApplicationControllerBundle.xlf	Text resources for the application names as shown on the springboard.

Changing Language on a Deployed Application

Oracle MAF applications choose the resource bundle to load based on the iOS language settings. For example, if the language is set to Spanish in iOS Settings and there is a localized Spanish version of the XLIFF resource bundle deployed with the application, the localized Spanish XLIFF resource bundle is used at runtime instead of the base English XLIFF resource bundle.

> **Note:** In order to run the SIM Mobile application in Brazilian Portuguese, the device language must be set to Brazilian Portuguese and the device region setting should be set to Brazil. After updating the language in the iOS settings, the application should be closed and reopened for the new language settings to take effect.

Language & Region Page