Oracle® Retail Mobile Merchandising

Implementation Guide
Release 15.0
E65651-02

January 2016

ORACLE

Oracle® Retail Mobile Merchandising Implementation Guide, Release 15.0

Primary Authors: Harish Ramamurthi / Steve Line / Daniel Balm / Matthew Scheele / Glenn Gonzales/ Gopal Edara / Rakhee Prabhudesai

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 Mobile Merchandising, Implementation Guide, Release 15.0

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 Implementation Guide describes the requirements and procedures to extend and customize this Oracle Retail Mobile Merchandising release.

Audience

This Implementation Guide is intended for Oracle Retail Mobile Merchandising application integration and implementation staff.

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 Mobile Merchandising Release 15.0 documentation set:

- *Oracle Retail Mobile Merchandising Installation Guide*
- *Oracle Retail Mobile Merchandising Release Notes*
- *Oracle Retail Mobile Merchandising User Guide*
- *Oracle Retail Mobile Merchandising Security 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, 15.0) or a later patch release (for example, 15.0.1). 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 also 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

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction

Oracle Retail Mobile Merchandising provides on-the-go visibility into certain Oracle Retail Merchandising application transactions and provides the ability, in some cases, to take action on existing transactions. The specific functions supported within Mobile Merchandising are:

- Recent Allocations
- Sales Audit (ReSA) Dashboard
- Sales Audit Store Search
- Invoice Matching

Key Features of Mobile Merchandising

Oracle Retail Mobile Merchandising provides the following features:

Recent Allocation

- Look Up Recent Allocations
- View Allocation Details
- Approve/Reserve Allocation
- View Item Metrics

Sales Audit

- Store Search
- View Store Day Summary Dashboard
- View Store Day
- View Store Details
- Share Store Day/Details via Email

Invoice Match

- View Invoice Match Metrics
- View Supplier Sites
- View Supplier Site Details / Performance
- Share Supplier Site Details via Email
- View Supplier Site Invoices
- View Employees
- View Employee Details
- View Employee Invoices
- View Invoices tied to Role

Second Screen Display

- Link display to Merchandising web apps

Skills Needed for Implementation

You need an understanding of the following applications and technical concepts.

Applications

You should understand the interface requirements of the integrated applications and data sources. You need this knowledge for the following applications:

- Oracle Retail Allocation release 15.0
- Oracle Retail Sales Audit release 15.0
- Oracle Retail Invoice Matching release 15.0

Technical Concepts

You should understand the following technical concepts:

- JDeveloper 12.1.3
- Technical architecture of the Oracle Retail Mobile Merchandising
- Application servers
- Java coding, including REST Java coding concepts
- XML manipulation
- Apple Enterprise Development setup and deployment
- Certificate creation and deployment

For customization of the user interface, you should also understand the following technical concepts:

- Mobile Application Framework (MAF) 2.2
- CSS programming

Technical Specifications

The following section covers the technical specifications.

Runtime Support and Certification Matrix

Mobile Operating System	Version	Mobile Application Framework (MAF) Certified Devices	MAF Supported Devices	RGBU Certified Devices
iOS	9.x	iPhone 6 (9.0.1) iPad Air 2nd gen (9.0.1)	iPhone (4S, 5, 5C, 5S, 6, 6 Plus, 6S, 6S Plus) iPod Touch (5th and 6th generation) iPad (2nd, 3rd and 4th generation) iPad Mini (1st, 2nd, 3rd and 4th generation) iPad Air (1st and 2nd generation)	iPad 4th gen (9.1)

Mobile Operating System	Version	Mobile Application Framework (MAF) Certified Devices	MAF Supported Devices	RGBU Certified Devices
iOS	8.x	iPhone 6 Plus (8.1.2) iPhone 6 (8.1) iPhone 6 (8.0.2) iPhone 5S (8.4.1) iPhone 5 (8.1) iPad 4th gen (8.4.1) iPad 4th gen (8.1.2) iPad Air 2nd gen (8.4.1) iPad Air 1st gen (8.3) iPad Air 1st gen (8.1) iPad Mini 3rd gen (8.1.2) iPod Touch 5th gen (8.3)	iPhone (4S, 5, 5S, 5C, 6, 6 Plus) iPod Touch (5th and 6th generation) iPad (2nd, 3rd, and 4th generation) iPad Mini (1st, 2nd, and 3rd generation) iPad Air (1st and 2nd generation)	iPod Touch (5th generation) (8.3) iPad Mini 1st gen (8.3)
Android	5.x	Google Nexus 9 (5.0) Google Nexus 7 (5.1.1) Google Nexus 7 (5.0.2) Google Nexus 6 (5.1) Samsung Galaxy S6 Edge (5.0.2) Samsung Galaxy S6 (5.0.2)	Any device running Android 5.0 or above with Minimum 1 GHz processor / 1 GHz Dual Core or better (Recommended) and at least 1 GB of total RAM	
Android	4.x	Asus Transformer Infinity (4.2.1) Google Nexus 7 (4.4.4) Google Nexus 7 (4.4.3) Google Nexus 7 (4.4.2) Google Nexus 4 (4.4.4) Motorola Xoom (4.1.2) Samsung Galaxy S5 (4.4.2) Samsung Galaxy S2 (4.1.2) Samsung Galaxy Note 3 (4.4.2) Samsung Galaxy Note 4 (4.4.4) Samsung Galaxy Tab 8 (4.4.2) Samsung Galaxy Tab 10.1 (4.0.3) Samsung Galaxy Tab 10.1 (4.3.0) Samsung Galaxy Tab Pro 12.2 (4.4.2)	Any device running Android 4.0.3 or above with Minimum 1 GHz processor / 1 GHz Dual Core or better (Recommended) and at least 1 GB of total RAM	Asus Nexus 7 (4.4.2)

Build Time Support

Mobile Platform	Mobile SDK Version	Native Development Tool Version	JDeveloper Version	MAF Version
iOS	iOS SDK 8.x	Xcode 6. x	Oracle JDeveloper 12.1.3	2.2

2

Understanding Application Architecture

This chapter discusses the architecture of the Oracle Retail Mobile Merchandising application.

Overview

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

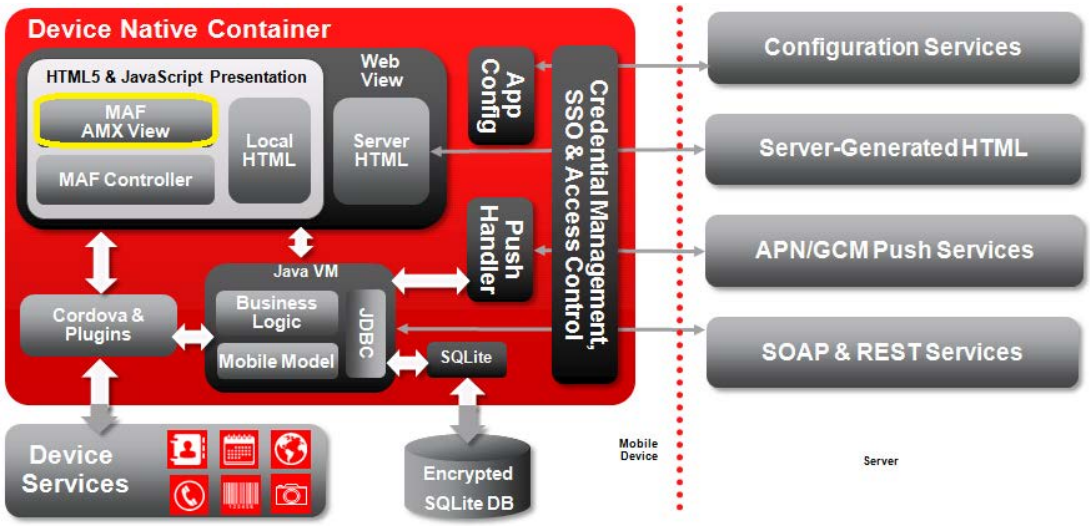
MAF defines a "feature" as "a reusable, self-contained module of application functionality." A MAF feature can use one of three content types: AMX, Local HTML or Server (Remote) HTML. Mobile Merchandising features are built using AMX.

For an introduction to MAF see Section 1.1, "Introduction to Mobile Application Framework" in the online book "Developing Mobile Applications with Oracle Mobile Application Framework".

Runtime Architecture

The runtime architecture of Mobile Merchandising is provided by MAF.

For an overview of the runtime architecture of MAF, see Section 1.2, "About the MAF Runtime Architecture" in the online book "Developing Mobile Applications with Oracle Mobile Application Framework".



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 Mobile Merchandising Security Guide*.

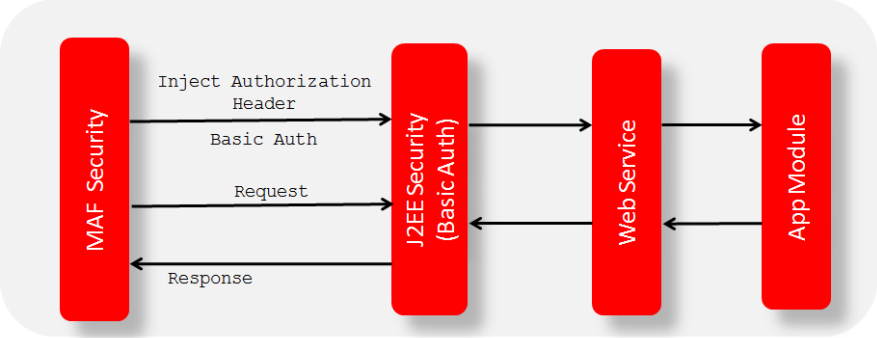
Authentication

Mobile Merchandising applications use MAF-based security to authenticate users to login /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 consumed by Oracle Retail Mobile Merchandising application are secured using a basic authentication mechanism and are access controlled using the standard J2EE authorization model. For more information on J2EE authorization model, see the 'Securing Web Application' section available at http://docs.oracle.com/cd/E28280_01/web.1111/e13711/thin_client.htm#SCPRG133.

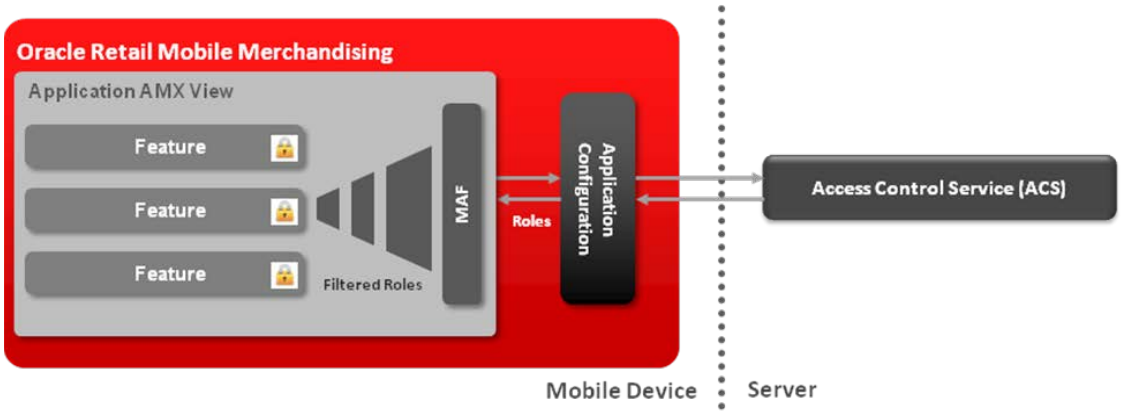


Authorization

Mobile Merchandising implements MAF-based access control functionality to secure role-based access to Mobile UI features:

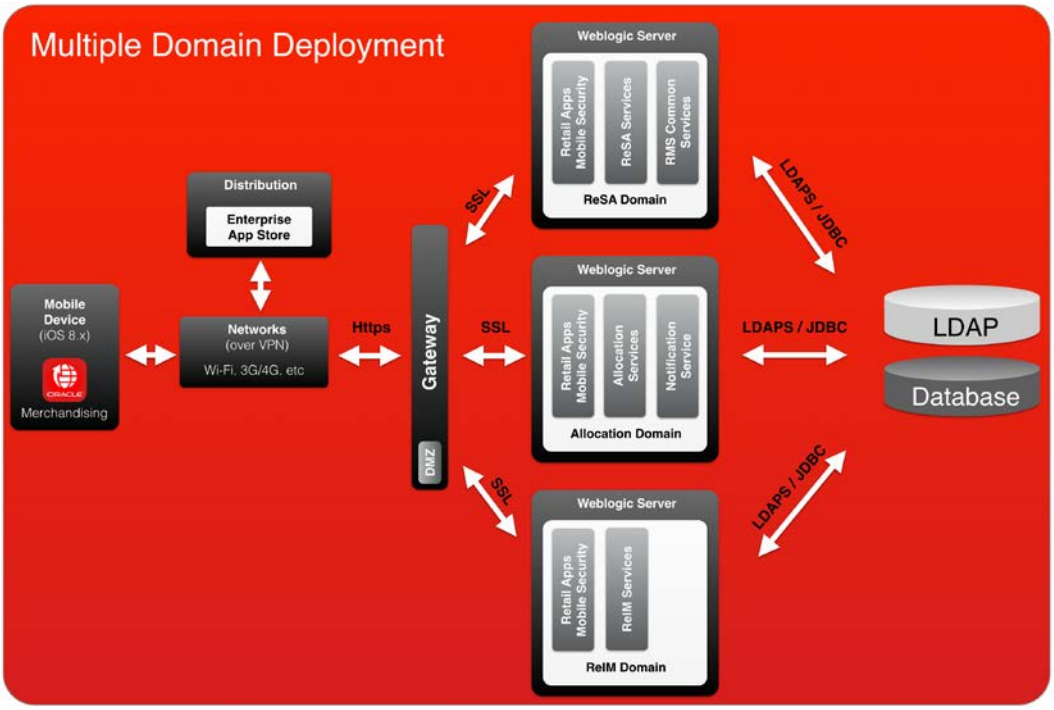
- MAF determines access to a feature based on the constraints defined on a feature. These constraints can be based on user roles or user privileges.
- MAF provides the ability to configure an Access Control Service (ACS) to get roles and privileges for a given user. MAF also provides Request and Response specifications to implement the ACS based on REST standards.
- The MAF also provides support to validate feature role constraints against the roles returned by ACS. Features are enabled/disabled based on the verification result.

The diagram below describes an Access Control Service (ACS) based role filtering on MAF features.



Deployment

The following diagram describes the deployment details of the Mobile Merchandising application.



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

Service Endpoint Assembly:

1. Allocation services are assembled as part of the Allocation application EAR file.
2. ReSA services are assembled as part of the ReSA application EAR file.
3. ReIM services are assembled as part of ReIM Application Ear file.
4. RMS Services are assembled as part of RMS application Install.
5. The Notification and Access Control service is deployed as part of every application's (ReSA, Allocation, ReIM, and RMS) installer.
6. Second Screen service is split between the common install and the websocket relay EAR
7. Push Notifications services are provided in a separate EAR file.

Notification and Access Control service is deployed as part of every application's (ReSA, Allocation, ReIM, and RMS) installer.

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

3

Configuration

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

Application Configuration

Configuring the application includes allowing the application to authenticate users, connect to Web services, generate URLs for in-context-launching to the ReSA Portal desktop application, and in certain cases, access remote images. The exact configuration required depends on which features are deployed (You do not need to configure connections for features that you are not using.).

The following table lists the connections required by each feature, along with their connection type.

Feature	Connection (Type)
Authorization	MerchMobileLoginServer (Login)
Recent Allocations	AllocationLoginServer (Login) (optional) AllocMobile (Service)
ReIM Dashboard	ReimLoginServer (Login) (optional) ReimService (Service)
ReSA Dashboard	ResaLoginServer (Login) (optional) ResaService (Service) RMSCommonService (Service) ResaPortal (desktop application URL)
ReSA Store Search	ResaLoginServer (Login) (optional) ResaService (Service) RMSCommonService (Service) ResaPortal (desktop application URL)
Configuration	ConfigServiceLogin (Login) ConfigService (URL)
Notifications	NotificationService (Service)
Second Screen	SecondScreenSocket (Service) SecondScreenService (Service)

Note: The term, Optional, in the above table indicates Login connections could be removed (or ignored) when all the services are deployed to the same domain.

Mobile Merchandising 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.

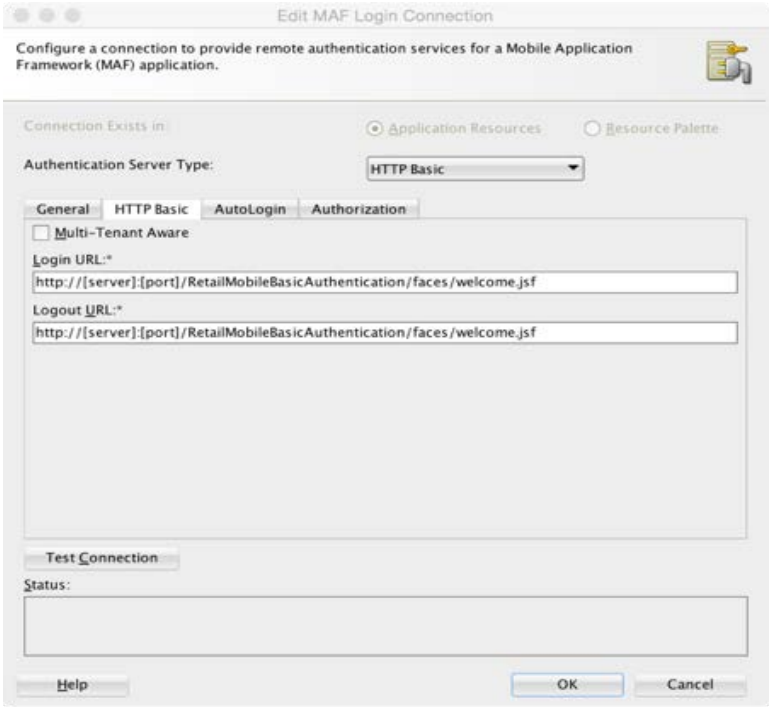
Another option is to update the placeholder connections with valid URLs prior to building the application so that the application is ready to run immediately after installation on a device. In either case, it is always possible to update the connections used on the device at a later time through the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

The following sections describe how to configure the connections.xml file manually. It is also possible to create a new connections.xml file from a template within the Configuration feature by tapping the "New" button on the Config Setup page.

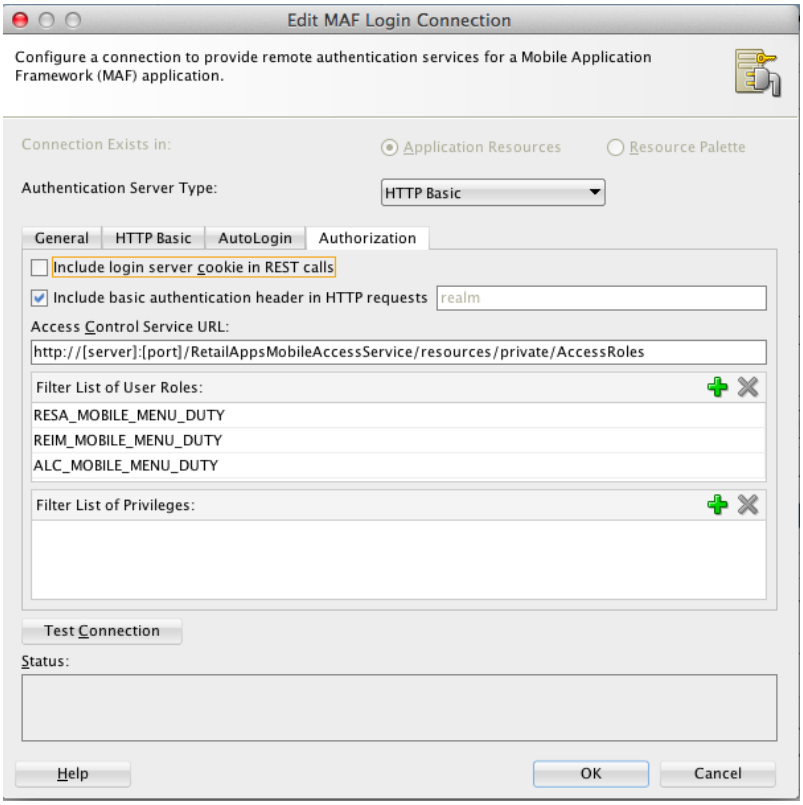
Security Configuration

The Authorization feature uses the **MerchMobileLoginServer** connection and must be configured for HTTP Basic authentication as well as Authorization.

1. 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.



2. Configure Authorization on the Authorization tab of the Edit MAF Login Connection window by replacing "[server]:[port]" with the correct values for your environment.



3. The **ResaLoginServer** connection should be configured for HTTP Basic authentication if the ReSA services are not on the same domain as the **MerchMobileLoginServer** connection.
4. In the same way, the **AllocationLoginServer** connection should be configured for HTTP Basic authentication if the Allocation services are not on the same domain as the **MerchMobileLoginServer** connection.
5. In the same way, the **ReimLoginServer** connection should be configured for HTTP Basic authentication if the Reim services are not on the same domain as the **MerchMobileLoginServer** connection.

For more information on accessing and configuring login connections, see the 'Configuring MAF Connections' section in the MAF documentation available at https://docs.oracle.com/middleware/maf220/mobile/develop-maf/GUID-3975EBAB-DB9E-4BC2-B814-619F30AF897D.htm#ADFMF1024

If the ReSA services are on the same domain as the **MerchMobileLoginServer** connection, then make the following updates to the application configuration.

1. Go to the **Security** tab of the maf-application.xml overview editor and select **<application login server>** for both of the ReSA features. (resa.mobile.dashboard and resa.mobile.storesearch).
2. Find the **ResaService** connection within the connections.xml file and set the **adfCredentialStoreKey** to **MerchMobileLoginServer**. If desired, remove the **ResaLoginServer** connection from the connections.xml file.
3. If desired, remove the **ResaLoginServer** connection from the connections.xml file.

Likewise, if the Allocation services are on the same domain as the **MerchMobileLoginServer** connection then make the following updates to the application configuration:

1. Go to the **Security** tab of the maf-application.xml overview editor and select **<application login server>** for the Recent Allocations feature (alc.mobile.RecentAllocations).
2. Find the **AllocMobile** connection within the connections.xml file and set the **adfCredentialStoreKey** to **MerchMobileLoginServer**. If desired, remove the **AllocationLoginServer** connection from the connections.xml file.
3. If desired, remove the **AllocationLoginServer** connection from the connections.xml file.

Likewise, if the ReIM services are on the same domain as the **MerchMobileLoginServer** connection then make the following updates to the application configuration:

1. Go to the **Security** tab of the maf-application.xml overview editor and select **<application login server>** for the ReIM Dashboard feature (reim.mobile.dashboard).
2. Find the **ReimService** connection within the connections.xml file and set the **adfCredentialStoreKey** to **MerchMobileLoginServer**.
3. If desired, remove the **ReimLoginServer** connection from the connections.xml file.

Likewise, if the Notification services are on the same domain as the **MerchMobileLoginServer** connection then make the following updates to the application configuration:

1. Go to the **Security** tab of the maf-application.xml overview editor and select **<application login server>** for the Notifications feature (platform.mobile.notifications).
2. Find the **NotificationService** connection within the connections.xml file and set the **adfCredentialStoreKey** to **MerchMobileLoginServer**.

Likewise, if the RMS Common services are on the same domain as the **MerchMobileLoginServer** connection then make the following updates to the application configuration:

1. Find the **RMSCommonService** connection within the connections.xml file and set the **adfCredentialStoreKey** to **MerchMobileLoginServer**.

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

Service Configuration

Open the connections.xml file, find each of the following URL connections, and update the [server]:[port] portion of the URL to refer to the server and port of the deployed services.

Service Connection	Placeholder URL
AllocMobile	http://[server]:[port]/Allocation14/services/private
ReimService	http://[server]:[port]/ReimRestService/services/private
ResaService	http://[server]:[port]/ResaReSTServices/services/private
RMSCommonService	http://[server]:[port]/RMSService/services/private
NotificationService	http://[server]:[port]/RetailAppsReSTServices/services/private/Notifications

Service Connection	Placeholder URL
SecondScreenSocket	http://[server]:[port]/RetailAppsWebSocketRelayServer/sockets
SecondScreenService	http://[server]:[port]/RetailAppsReSTServices/services/private
PushNotificationService	http://[server]:[port]/RetailPushServices

By default, ORMM is configured to work with version 15.0 of the services for each application (Allocation, ReSA, and ReIM). However, if there is a need to modify the version of the services that ORMM is going to be running against, the version number is stored as a configurable setting within the application and can be modified at installation time.

To modify the service version setting, take the following steps:

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_SERVICE_VERSION**.
3. Change the value to a version that you want the application ReST Services to run against. For example, 15.0.1 or 15.0.2.

The service version number configured in adf-config.xml is a global application configuration value. It sets the service version for all Web service calls made by Mobile Merchandising. By default, the **RETAIL_SERVICE_VERSIONING_ENABLED** property is set to **True** to enable a service validation check between the Mobile client and services.

To Modify/skip the service versioning check, take the following steps:

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_SERVICE_VERSIONING_ENABLED**.
3. Change the value to **False** to disable service versioning validation.

Mobile Merchandising calls the secure Web services. In MAF, the mapping between secure Web services and policies is saved in the wsm-assembly.xml file and configured through the maf-application.xml file. For more information on the policies supported by MAF and how to configure them, see '15.7 Accessing Secure Web Services in Developing Mobile Applications with Oracle Mobile Application Framework' available at http://docs.oracle.com/middleware/maf220/mobile/develop-maf/GUID-74D1067F-9040-4490-BA9A-172E376C478F.htm#ADFMF-GUID-BF2C9878-C87A-4304-81EB-1466559492B5.

Mobile Merchandising ships with a policy configuration that works with HTTP Basic authentication. To implement another authentication type, such as Web SSO authentication, you must change your Web service to policy mappings. For more information, see '15.7 Accessing Secure Web Services in Developing Mobile Applications with Oracle Mobile Application Framework' available at http://docs.oracle.com/middleware/maf220/mobile/develop-maf/GUID-74D1067F-9040-4490-BA9A-172E376C478F.htm#ADFMF-GUID-BF2C9878-C87A-4304-81EB-1466559492B5.

Note: After updating your Web service connections in the connections.xml file, the mapping between your Web service connection and its security policy (or policies) is not automatically updated by MAF and appears blank in the maf-application.xml file. You can simply add the policy (or policies) again. After doing so, there may be multiple entries for that particular Web service connection in wsm-assembly.xml file. It is recommended that you manually remove any entries that are no longer applicable.

Configuring Application Connections

The ReSA features reference a connection to the ReSA Portal desktop application. The connection is named ResaPortal. The connection can be found in the connections.xml file and modified the same way as the Web service connections.

The connection should be set to a value as follows:

https://<server address>:<port>/ResaPortal/faces/Home

where <server address> and <port> refer to the server address and port number respectively of the location where the ReSA Portal application is deployed.

Adding Remote Image URLs to the Application Whitelist

The Recent Allocations feature shows item images on several screens. MAF applications only allow connections to external domains that have been added to a whitelist. All connections in the connections.xml file are automatically added to the whitelist. If your remote images are coming from a domain other than a connection defined in your connections.xml file, those domains must be added to the whitelist in maf-application.xml of the Merchandising Mobile workspace. If they are not added to the whitelist, the images will not render in the application. 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)' of 'Developing Mobile Applications with Oracle Mobile Application Framework' for MAF 2.2 available at http://docs.oracle.com/middleware/maf220/mobile/develop-maf/GUID-0C7FD01B-6A71-406F-A995-84483F96A910.htm#ADFMF1478.

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.MerchMobile*", then the connections.xml could be located at URL: http://server:port/SomeLocation/com.company.MerchMobile/connections.xml.

You need two URLs to complete the configuration process:

1. The “Configuration URL” which 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.)
2. The “Configuration Login URL” which 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 and the user is required to use the Configuration feature to get valid connections, then 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 do this, take the following steps:

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**.

Connectivity Configuration

Mobile Merchandising requires a network connection. If you open the application without an active network connection, the application displays a page notifying you of the network connectivity requirements of the application as well as the current network connectivity status of the device.

This value is updated with the frequency specified by the **RETAIL_NETWORK_CONNECTIVITY_CHECK_RATE** property in adf-config.xml. You may want to review the device settings to ensure a network connection (for example, Wi-Fi, data, etc.) is configured properly. Once the device has a recognized network connection, the network status value updates to reflect that the application has a connection. You may tap the **Retry** button to proceed using the Mobile Merchandising application.

To change the rate at which the network connectivity updates status, take the following steps:

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_NETWORK_CONNECTIVITY_CHECK_RATE**.
3. Change the value to the desired network connectivity check rate. The value must be entered in milliseconds.

Navigation Configuration

> **Note:** There is no need to update the navigation configuration unless changes in organization, labeling, etc. 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>
            value: <EL pointing to a value to display on item>,
            imgSrc: <Path to image to use as icon for value>,
            cacheValue: <True/false to cache the value for a set time>,
            secured: <True/false to only show the item when logged in>

        },
        ...
    ]
}
```

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 the current feature group. A feature link is included by specifying the featureId and optionally a label. If no label is provided, the name of the feature referenced is used.

Additionally, you can specify the following optional properties:

imgSrc - The imgSrc property specifies the path to an image that is displayed along with the footer link. The path to the image needs to be prefixed with "../../<project>/public_html/" where <project> is the folder name of the project containing the image you are referencing.

value - The value property references a value that you want to display along with the footer link (e.g. a count of unread messages). By default, the value refreshes every time the springboard is opened and when you enter/exit the feature group list in the springboard. The value should be given in EL expression syntax e.g. #{applicationScope.someValue}.

> **Note:** You may see a different syntax in the delivered navigation.json file; referencing custom values in this way is not supported.

cacheValue - The cacheValue property is set to true or false and controls how often the value set in the value property is refreshed. Since the call to get the value may take some time, it is recommended to set the cacheValue property to true. When true, the value only refreshes when the earlier specified value refresh conditions are met and a refresh rate interval has been exceeded. This prevents the value from refreshing too often. The refresh rate is specified in milliseconds by the SPRINGBOARD_SERVICE_CALL_INTERVAL property in the adf-config.xml file. This property is edited similar to how the other adf-config.xml properties are edited. The default value refresh behavior is observed if this value is omitted or set to false.

secured - The secured property controls the visibility of the footer node. If the secured property is set to true, the feature link will only be displayed for logged in users. If the secured property is set to false or not present, the feature link will always be visible if there are no other feature constraints that would cause it to not be visible.

Notifications Configuration

Notifications has a feature to navigate to the source of a notification based on mappings in the notifications.json file in the application controller project. This file has the following structure:

```
{
    <Application code> : <Feature ID>
}
```

It consists of a single object mapping the appCode of received notifications to the corresponding feature in the mobile application. Currently, only Recent Allocations supports this feature.

Second Screen Configuration

In addition to setting the connections, Second Screen requires whitelisting the domain(s) you will be connecting to the web applications on. This is done on the security pane of maf-application.xml. You will need to make sure there are whitelist entries for both the applications themselves and the SSO login. The * wildcard is supported for whitelist entries.

4

User Interface Customization

This chapter describes the supported customizations of the Oracle Retail Mobile Merchandising 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 http://docs.oracle.com/middleware/maf220/mobile/develop-maf/GUID-9557B4D0-15BD-4A89-A9DC-9E39596F95ED.htm#ADFMF25118 and Chapter 18 'Customizing MAF AMX Application Feature Artifacts of Developing Mobile Applications with Oracle Mobile Application Framework' available at http://docs.oracle.com/middleware/maf220/mobile/develop-maf/GUID-03158967-8E75-4521-8374-89FBF36D0729.htm#ADFMF24124.

Note: For more information on the scope of customizations allowed under the MAF Foundation license that is provided with Oracle Retail Mobile Merchandising, see the 'Restricted Use Licenses' chapter of 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 http://docs.oracle.com/middleware/maf220/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 Mobile Merchandising application with MDS. MDS has a notion of customization layers that must be setup 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 Mobile Merchandising classpath and then referenced in the adf-config.xml file of the Mobile Merchandising workspace.

For more information, see 'Developing Mobile Applications with Oracle Mobile Application Framework' available at http://docs.oracle.com/middleware/maf220/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 Mobile Merchandising workspace. See 'Developing Mobile Applications with Oracle Mobile Application Framework' for more information.

Customizing the Application Id

You will have to update the Application Id in order to deploy your company's version of the Mobile Merchandising 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 Mobile Merchandising, 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, take the following steps:

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 profile(s) match. They are used for different purposes within MAF applications, but for proper functioning of the Mobile Merchandising 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 Mobile Merchandising 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 '27 Deploying MAF Applications' of 'Developing Mobile Applications with Oracle Mobile Application Framework' available at http://docs.oracle.com/middleware/maf220/mobile/develop-maf/GUID-9579F7C9-E696-43A0-8087-8953187A0E81.htm#ADFMF-GUID-9579F7C9-E696-43A0-8087-8953187A0E81 for information on how to customize application images in an iOS deployment profile.

> **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 Mobile Merchandising.

Customizing Application Skins

The overall look and feel of the Mobile Merchandising 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 Mobile Merchandising 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 Mobile Merchandising MAA file:

1. Open JDeveloper in the Studio Developer role.
2. Open the Mobile Merchandising workspace created from the delivered MAA file.
3. Create a new CSS skin file in the **MerchMobileApplicationController** project. See Chapter 7 'Skinning MAF Applications' of 'Developing Mobile Applications with Oracle Mobile Application Framework' for more information on how to create CSS skin files and how MAF skinning works.
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' of '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 Mobile Merchandising version, the custom skin changes are still applied over the base Merchandising application.

Customizing String Resources

In order to support localization, Oracle Retail Mobile Merchandising 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 Mobile Merchandising 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 Mobile Merchandising application in a future-proof way, you must create new resource bundles under the Studio Developer role, add strings to these new bundles, configure the Resource Bundle settings in the project's properties to include your new bundle in the Bundle Search, and then reference strings from these new resource bundles in customizations to AMX pages or other customizable artifacts under the Customization Developer role.

Customizing 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 Strings in Allocation Feature

The AllocMobileViewController project contains the string resources used by the Recent Allocations feature. If you want to change strings used in the Recent Allocations feature, create a new resource bundle (while in the Studio Developer role) under the AllocMobileViewController 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.

Customizing Strings in ReIM Feature

The ReimMobileViewController project contains the string resources used by the Invoice Matching feature. If you want to change strings used in the ReIM feature, create a new resource bundle (while in the Studio Developer role) under the ReimMobileViewController 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.

Customizing Strings in ReSA Feature

The ResaMobileViewController project contains the string resources used by the Store Day Summary Dashboard and Store Search features. If you want to change strings used in the ReSA features, create a new resource bundle (while in the Studio Developer role) under the ResaMobileViewController 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.

Customizing Merchandising Mobile

Springboard Navigation

The MerchMobileApplicationController 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 MerchMobileApplicationController 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 like so: “customBundle” : “ customer.custom.bundle.CustomBundle”. The complete bundles object may look like the following:

```
"bundles" : {
"merchMobileApplicationControllerBundle" :
"oracle.retail.apps.merch.mobile.application.MerchMobileApplicatio
nControllerBundle",
"resamobileviewcontrollerBundle" :
"oracle.retail.apps.resa.mobile.ResaMobileViewControllerBundle",
"allocmobileviewcontrollerBundle" :
"oracle.retail.apps.alc.mobile.AllocMobileViewControllerBundle",
"reimmobileviewcontrollerBundle" :
"oracle.retail.apps.reim.mobile.ReimMobileViewControllerBun
dle"
}
```

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 Mobile Merchandising application.

Application Level

The MerchMobileViewController project contains the string resources used by the Mobile Merchandising 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 MerchMobileViewController 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 via MDS since the data could originate from Web services or are programmatically constructed.

Removing Features from the Application

Oracle Retail Mobile Merchandising 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, take the following steps:

1. Open the Mobile Merchandising 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.

The following are optional steps that can be taken to remove references to the feature from the navigation.json file that is used to render the Springboard. These steps are optional because the only features that are rendered on the Springboard are the features that are referenced in maf-application.xml's Feature References.

1. Open the Mobile Merchandising workspace under the Studio Developer role. (The navigation.json file cannot be customized under the Customization Developer role.)
2. Open the navigation.json file.
3. Find the object whose identifier property maps to the feature identifier of the feature you deleted.
4. Delete the found object from the JSON structure.
5. Save your changes.

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

Adding New Features to the Application

Oracle Retail Mobile Merchandising can be customized to add new custom features. The recommended approach to adding new features to Mobile Merchandising 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 Mobile Merchandising application.

For more information on FARs, see the section 'Reusing MAF Application Content of Developing Mobile Applications with Oracle Mobile Application Framework' available at http://docs.oracle.com/middleware/maf220/mobile/develop-maf/GUID-393B8F68-13D3-402A-B6D6-761FC6944B81.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, take the following steps:

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 Mobile Merchandising 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 (i.e. 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 Mobile Merchandising 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."}
    ```

    Note: Changes to the navigation.json file must be done under the Studio Developer role and will not be persisted across upgrades by MDS.

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 Mobile Merchandising 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 Mobile Merchandising 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 http://docs.oracle.com/middleware/maf220/mobile/develop-maf/GUID-03158967-8E75-4521-8374-89FBF36D0729.htm#ADFMF24124. The following subsections provide 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, take the following steps:

1. Open the Mobile Merchandising application under the JDeveloper Customization Developer role.
2. Open the AMX page or page fragment you wish 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, etc. For more information on adding UI Components to a page, see Chapter 13, 'Creating the MAF AMX User Interface' available at '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, take the following steps:

1. Open the Mobile Merchandising 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 (e.g. 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), take the following steps:

1. Open the Mobile Merchandising 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 Mobile Merchandising, 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. Click **Finish** in the Deploy PlatformMobileFeatures window to deploy the feature archive JAR file.
8. Overwrite the existing PlatformMobileFeatures.jar file in the lib folder of your Mobile Merchandising workspace with the PlatformMobileFeatures.jar you just created.
9. Open the Merchandising workspace.
10. Deploy the Merchandising application.

> **Note:** Whenever you upgrade the Mobile Merchandising 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.

Upgrading to a New Version

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

- The addition of any libraries or JARs to the application-level Libraries and Classpath. Note this includes the following:
  - The JAR(s) containing your customization class(es).
  - Any FAR(s) that were added to add new feature content to the Mobile Merchandising application.
- Any changes to the *navigation.json* file.
- Any changes to the application branding.
  - Any images you've 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 Mobile Merchandising Section, see 'Upgrading a MAF Application with Customizations' available at

http://docs.oracle.com/middleware/maf220/mobile/develop-maf/GUID-9557B4D0-15BD-4A89-A9DC-9E39596F95ED.htm#ADFMF24079.

Note: Although JDeveloper creates a backup copy of the workspace, it is recommended that you create your own backup in case the JDeveloper backup fails for some reason.

Enabling Optional Features

Push Notifications

To activate push notifications for Mobile Merchandising, make the following changes to the configuration:

- Set the PushNotificationsService connection to point to your deployment of RetailPushServices
- Enable the PushPlugin in maf-application.xml
- Set ENABLE_PUSH_NOTIFICATIONS to true in adf-config.xml
- Set JDeveloper to build the application using the push-enabled provisioning profile you created during the installation steps
- Set GCM_SENDER_ID to the Sender ID value you generated during installation if you will be deploying to Android devices

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.
- Modifying business logic:
  - Most business logic is implemented outside of the artifacts that are customizable by MDS, so this is unsupported.
- Modifying ReST service calls:
  - Similar to modifying business logic, service calls cannot be customized to add additional request parameters, accept different responses, etc.

MDS Customizations and Configuration Services

If the Mobile Merchandising 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.)

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.

Recall that 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 (e.g. switching between certain files, deployment, etc.), 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.

5

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 Mobile Merchandising 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 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 Mobile Merchandising MAA files.

When the Mobile Merchandising 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 Platform 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/PlatformMobileApplicationController/src/oracle/retail/apps/platform/mobile/PlatformMobileBundle.xlf	Default string resources referenced by Retail Mobile Platform AMX fragments.
PlatformMobile/PlatformMobileViewController/src/oracle/retail/apps/platform/mobile/PlatformMobileViewControllerBundle.xlf	String resources for the Configuration Services feature, the Springboard feature, etc.

Mobile Merchandising Resource Bundles

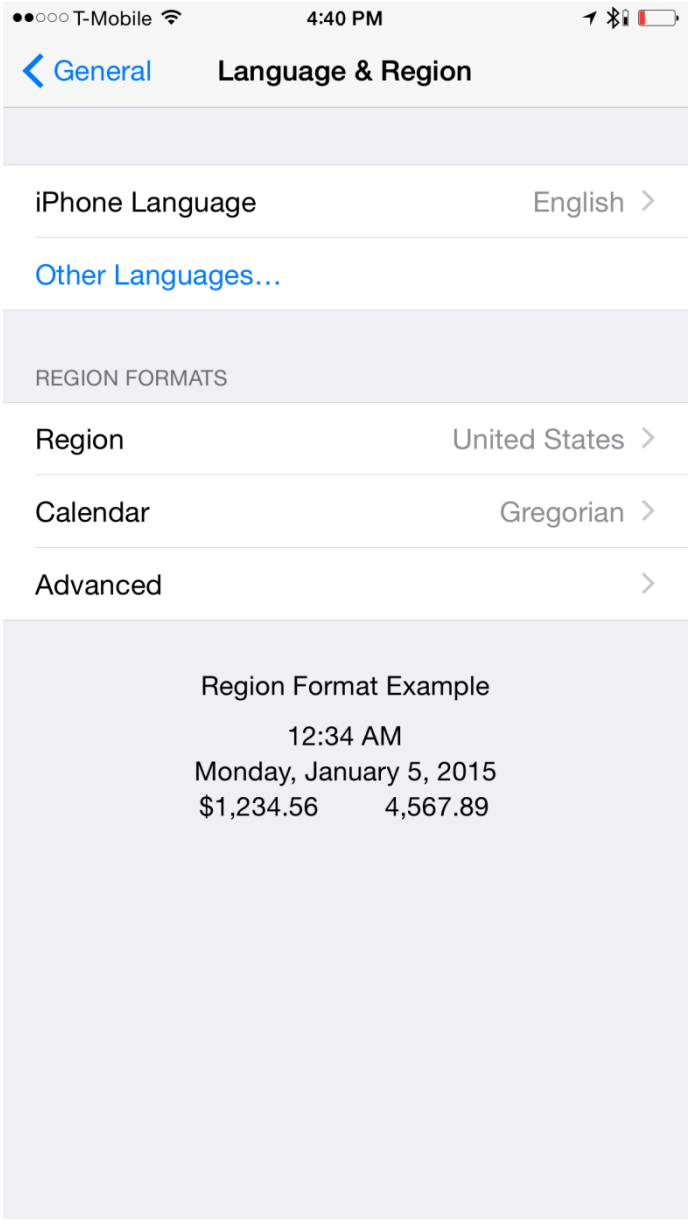
The following table lists the resource bundles that are packaged in the Mobile Merchandising MAA file.

Base XLIFF Resource Bundle	Text resource descriptions
MerchMobile/.adf/META-INF/MerchMobileBundle.xlf	Text resource for the application name.
MerchMobile/AllocMobileViewController/src/oracle/retail/apps/alc/mobile/AllocMobileViewControllerBundle.xlf	Text resources for the Recent Allocations feature.
MerchMobile/MerchMobileApplicationController/src/oracle/retail/apps/merch/mobile/application/MerchMobileApplicationControllerBundle.xlf	Text resources for the application names as shown on the springboard.
MerchMobile/MerchMobileViewController/src/oracle/retail/apps/merch/mobile/MerchMobileViewControllerBundle.xlf	Text resources for the Merchandising features (e.g. About feature).
MerchMobile/ReimMobileViewController/src/oracle/retail/apps/reim/mobile/ReimMobileViewControllerBundle.xlf	Text resources for the ReIM feature.
MerchMobile/ResaMobileViewController/src/oracle/retail/apps/resa/mobile/ResaMobileViewControllerBundle.xlf	Text resources for the ReSA features.

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 Merchandising 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 Merchandising application should be closed and reopened for the new language settings to take effect.


●●○○○ T-Mobile
4:40 PM
General
Language & Region
iPhone Language
English
Other Languages…
REGION FORMATS
Region
United States
Calendar
Gregorian
Advanced
Region Format Example
12:34 AM
Monday, January 5, 2015
$1,234.56 4,567.89