CAMPUS MOBILE IMPLEMENTATION GUIDE
# Table of Contents

1. **Background** ................................................................................................................................. 5  
   1.1  Introduction ................................................................................................................................. 5  
   1.2  Overview of Campus Mobile ...................................................................................................... 5  
   1.3  Overview of ADF Mobile ............................................................................................................ 6  

2. **Campus Solutions** ......................................................................................................................... 10  
   2.1  Campus Solutions Configuration ............................................................................................... 10  
   2.2  Integration Broker Configuration ............................................................................................ 10  
   2.3  Security Configuration ............................................................................................................. 10  
       2.3.1  Authentication Service ..................................................................................................... 10  
       2.3.2  Authorization Service ..................................................................................................... 11  
   2.4  About the Entity Registry ........................................................................................................ 11  
       2.4.1  Entity Profiles ................................................................................................................ 11  
   2.5  Configuring Notifications ........................................................................................................ 11  
       2.5.1  Framework ..................................................................................................................... 11  
       2.5.2  Consumers ..................................................................................................................... 12  
       2.5.3  Templates ....................................................................................................................... 13  
       2.5.4  Preferences .................................................................................................................... 14  
   2.6  Message Catalog Considerations ............................................................................................ 14  

3. **Campus Mobile** ............................................................................................................................ 16  
   3.1  Overview .................................................................................................................................... 16  
   3.2  Using JDeveloper ....................................................................................................................... 16  
       3.2.1  Installing and Setting up JDeveloper ................................................................................ 16  
       3.2.2  Creating a New Mobile Application From the Delivered Mobile Application Archive File ...... 17  
   3.3  Configuring Your Mobile Application ....................................................................................... 17  
       3.3.1  Web Services ................................................................................................................... 17  
       3.3.2  Security ........................................................................................................................... 18  
       3.3.3  Properties ....................................................................................................................... 19  
       3.3.4  Currency Symbols .......................................................................................................... 19
3.7.5 Supporting Your Mobile Application

Figure 1-1 ADF Mobile Runtime Architecture

Figure 3-1 Application Navigator
1. BACKGROUND

1.1 INTRODUCTION

Campus Mobile is a new student self service interface designed for smartphone devices. It is delivered as part of Campus Solutions Self Service and is deployed as an iOS or Android application. The application enables students to have constant access to high-value transactions.

This document provides implementers with the information required to:

- Configure your existing Campus Solutions environment(s) to facilitate mobile access. Refer to chapter 2.
- Configure, brand, customize and extend the delivered Campus Mobile application archive in order to create and distribute a mobile application that is tailored to the needs of your students and organization. Refer to chapter 3.

Readers should progress through this document in the chapter order, since later chapters require knowledge of information described in earlier chapters.

Multiple pieces of documentation have been developed or updated to assist in the implementation effort.

See Campus Mobile Documentation, on My Oracle Support

See Working With the Notifications Framework, part of CS_Bundle_29_Campus_Community_9_0.pdf on My Oracle Support

See PeopleSoft Campus Community Fundamentals 9.0 PeopleBook, “Setting Up Entity Registry”

1.2 OVERVIEW OF CAMPUS MOBILE

Campus Mobile is built using Oracle’s ADF Mobile framework, which enables deployment to both iOS and Android as a native application.

The application features high-value transactions to which students desire to have constant, always-available access. Features of Campus Mobile include the ability to: view class schedule, view grades, view finances, view to do list, view holds, view schedule of classes, enroll in classes, and manage the enrollment shopping cart.

The Campus Mobile smartphone application is a secure, simplified, and personalized platform for students to conduct key Campus Self Service processes. You can configure, brand, customize and extend the Campus mobile application to suit your requirements.

Campus Mobile uses the delivered Campus Solutions web services to provide real-time read and write access to data within Campus Solutions. For Campus Mobile, RESTful versions of the web services are deployed, leveraging PeopleTools 8.52 functionality. RESTful web services require lower overhead than SOAP-based services, thus lowering the bandwidth requirement for delivering
data to a mobile device. To optimize the web service performance, Campus Mobile takes advantage of the Entity Profiling feature new to Campus Solutions. Entity Profile gives us the ability to create and use a mobile profile that limits the amount of data delivered in the web service payload to the subset of data that Campus Mobile needs; further lowering the bandwidth requirement for delivering data to a mobile device.

In parallel to Campus Mobile, the Campus Solutions Notification Framework provides proactive communication to students and enhances the student mobile experience. Campus Mobile enables students to manage how they receive notifications.

Note: Certain notifications are described in Campus Mobile documentation, but Campus Mobile is not necessarily a prerequisite to the implementation of these notifications. These notifications enhance the mobile experience for students and provide proactive information from Campus Solutions and thus they have been developed parallel to Campus Mobile.

1.3 OVERVIEW OF ADF MOBILE

The Oracle Application Development Framework (Oracle ADF) is an end-to-end Java enterprise framework that simplifies application development by providing out-of-the-box infrastructure services and a visual and declarative development experience.

ADF Mobile extends Oracle ADF, enabling you to create an application that can house different types of content and runs on mobile devices powered by the iOS and Android platforms. Within the application, the contained functional areas are referred to as application features and represent one or more specific pieces of the application's functionality.

An extension of Apache Cordova (see http://cordova.apache.org), the ADF Mobile architecture enables HTML5, as well as ADF-defined pages and task flows to be rendered in the same downloadable application.

ADF Mobile consists of the following parts:

- **Model** provided by Oracle ADF.
- **View** expressed as HTML or XML.
- **Controller** defined with XML.
- **Java runtime** powered by an embedded Java VM bundled with each application.

Figure 1-1 shows the overall runtime architecture of ADF Mobile.
The following elements comprise the ADF Mobile runtime architecture:

- **Device-Native Container** represents an application container, or template, compiled as a device-native application binary. This container provides the runtime environment for an ADF Mobile application to run as an on-device, native application in the mobile device's operating system (iOS or Android). Besides hosting the client-side components for an ADF Mobile application, it provides navigation utilities, such as a springboard and navigation bar, which enable access to particular application features.

- **Web View** is a part of the Device-Native Container that uses a mobile device's web engine to display and process web-based content. In an ADF Mobile application, the web view is the primary mechanism to render and deliver the application user interface.

- **Server HTML** represents a web-based user interface that is generated on the server and delivered as a web page to the ADF Mobile application. The application HTML code, business logic, and page flow logic are generated on a remote server. Server HTML can access device-native services, such as the camera, through the JavaScript API supported by Cordova, as long as it is running inside an ADF Mobile application.

---

*Figure 1-1 ADF Mobile Runtime Architecture*
**Note:** Campus Mobile does not make use of Server HTML. All application code is contained within the mobile application.

- **Local HTML** represents web pages developed using JDeveloper or third-party tools that are directly embedded within an ADF Mobile application. These pages are delivered as a part of the ADF Mobile application. Local HTML files can access device-native features through the JavaScript APIs supported by Cordova.

- **ADF Mobile AMX Views** are based on the ADF Mobile AMX technology that delivers a JSF-like development experience to working with an HTML5-based user interface. ADF Mobile AMX views are defined using UI and code editors provided by JDeveloper. These views are embedded into an ADF Mobile application and deployed to a mobile device. At runtime, the JavaScript engine in the web view renders ADF Mobile AMX view definitions into HTML5 components. Of the implementation approaches provided by ADF Mobile, application features developed using the ADF Mobile AMX components provide the most authentic device-native user experience through their extensive support of animation and gestures.

- **ADF Controller** is represented by a mobile version of the Oracle ADF controller that supports a subset of Oracle ADF task flow components available to a server-based Oracle ADF application. Both bounded and unbounded Oracle ADF task flows are supported, as well as a subset of events and scopes that are supported by the server-based ADF.

- **Java** provides a Java runtime environment for an ADF Mobile application. This Java Virtual Machine (JVM) is implemented in device-native code, and is embedded (or compiled) into each instance of the ADF Mobile application as part of the native application binary. The JVM is based on the JavaME Connected Device Configuration (CDC) specification.

- **Managed Beans** are Java classes that can be created to extend the capabilities of ADF Mobile, such as providing additional business logic for processing data returned from the server. Managed beans are executed by the embedded Java support, and therefore must conform to the JavaME CDC specifications.

- **ADF Model** in an ADF Mobile application supports a subset of business logic components available to a server-based Oracle ADF application. ADF model contains the binding layer that connects the business logic components with the user interface. In addition, the binding layer provides the execution logic to invoke REST- or SOAP-based web services.

- **Application Configuration** refers to services that allow key application configurations to be downloaded and refreshed. For example, URL endpoints for a web service or remote URL connection. Application configuration services download the configuration information from a WebDav-based server-side service.

**Note:** Campus Mobile does not make use of a remote Application Configuration. Configuration files are contained within the mobile application.
• **Credential Management and Access Control** refers to client-side services that provide security-related services for an ADF Mobile application. For example, a local credential store that securely caches user credentials to support an offline authentication or access control services that display or hide application features based on user access privileges.

• **Apache Cordova** (formerly known as PhoneGap) is an open-source code library that provides a common JavaScript API to access various mobile **Device Services**, such as the camera. Cordova provides a majority of the device services integration for an ADF Mobile application. Cordova JavaScript APIs are further abstracted as device data controls in the JDeveloper design time for ADF Mobile AMX-based views, allowing for integration of device services by dragging and dropping data controls to their ADF Mobile AMX views.

• **Local Data** refers to data stores that reside on the device. In ADF Mobile, these are implemented as encrypted SQLite databases. Create Retrieve Update Delete (CRUD) operations are supported to this local data store through the Java layer, using JDBC-based APIs.

• On the server side, the **Configuration Server** refers to a WebDav-based server that hosts configuration files used by the application configuration services. The configuration server is delivered as a reference implementation. Any common WebDav services hosted on a common J2EE server can be used for this purpose.

  **Note:** Campus Mobile does not make use of a Configuration Server. Configuration files are contained within the mobile application.

• On the server side, **ADF Mobile Browser** and **ADF Faces Rich Client** refer to frameworks for developing server-side applications that can be used for implementation of the remote URL ADF Mobile application feature.

  **Note:** Campus Mobile does not make use of ADF Mobile Browser or ADF Faces Rich Client. All application code is contained within the mobile application.
2. CAMPUS SOLUTIONS

2.1 CAMPUS SOLUTIONS CONFIGURATION

Campus Mobile requires a minimum PeopleTools level of PeopleTools 8.52. For information on Campus Solutions maintenance level requirements specific to a Campus Mobile release version, please refer to My Oracle Support document 1557671.1.

Note that configuration changes made within Campus Solutions under Set Up SACR > Common Definitions > Self Service are typically applicable to Self Service pages only and may not affect the functioning of the Enrollment Web Services or Campus Mobile itself.

2.2 INTEGRATION BROKER CONFIGURATION

The configuration of Integration Broker required to enable Campus Mobile are described in the Campus Self Service PeopleBook chapter Setting Up Campus Mobile. In particular, refer to sections “Setting Up Integration Broker for Campus Mobile,” “Activating Campus Mobile Service Operations and Handlers,” and “Generate Routings for the Following Service Operations.”

2.3 SECURITY CONFIGURATION

Users log into the Campus Mobile app using the same credentials that they use for Campus Self Service access through PIA. These credentials are passed to PeopleTools through the Authentication and Authorization services, which are described below.

As delivered, all features within Campus Mobile are protected with Permission List ‘HCCPCSSA3000’. This is a new Permission List created to support authorization in Campus Mobile. By default, this Permission List is mapped to the newly delivered “CS Mobile Student” Role, and a user must have this role to be able to use the application.

If you want to use a different role to secure access to Campus Mobile features, you should map Permission List ‘HCCPCSSA3000’ to that role.

If you want to use a different Permission List to secure access to Campus Mobile features, you must customize Campus Mobile as described in section 3.3.2.3 and map that Permission List to whichever role you want to use for this purpose.

In addition, you must assign the Permission List that is used to secure access to Campus Mobile features to each of the web service operations called from Campus Mobile. This is described in the Campus Self Service PeopleBook chapter Setting Up Campus Mobile, “Setting Up Security for Campus Mobile.”

2.3.1 AUTHENTICATION SERVICE

Campus Mobile can authenticate against any basic authentication server.
Campus Solutions provides a RESTful web service operation (SCC_USERREG_AUTHENTICATE) that is used exclusively to support authentication in Campus Mobile. Nothing is returned in the payload.

Any URL that is protected with basic authentication can be used as the login URL. You may implement your own URL and use it instead of the default one, in which case you must modify the configuration in your mobile application (refer to section 3.3.2.2).

### 2.3.2 Authorization Service

Since Campus Mobile makes use of the user’s Permission Lists to control access to its features, an Access Control Service must be deployed to return the user’s Permission Lists to Campus Mobile.

An Access Control Service (ACS) is a RESTful web service with JSON that enables users to download their user roles and privileges through a single HTTP POST message.

Campus Solutions provides a RESTful web service operation (SCC_USERREG_AUTHORIZE_POST) that is used exclusively to support authorization in Campus Mobile. It accepts input and returns the result in JSON message format.

You may implement your own ACS and use it instead of the default one, in which case you must modify the configuration in your mobile application (refer to section 3.3.2.3).

### 2.4 About the Entity Registry

#### 2.4.1 Entity Profiles

Campus Mobile uses Entity Profiling of the Enrollment Web Services in order to minimize the payload provided by the EWS. A “Mobile” profile is delivered that is designed to provide only the necessary fields to Campus Mobile.

You can view the delivered Entity Profiles for Campus Mobile, “Mobile” and “Mobile Cart” in the Entity Profile component (Set Up SACR > System Administration > Entity > Entity Profile).

For more information on the Entity Registry and Entity Profiles see the Campus Community PeopleBook chapter Setting Up Entity Registry.

### 2.5 Configuring Notifications

#### 2.5.1 Framework

The Campus Solutions Notifications Framework provides a consistent, extensible communication mechanism that Campus Solutions product areas can use to enable notifications to interested parties. Notifications are delivered through a notifications “Channel” – email, SMS, push, alerts, or worklists – based on a transaction performed by a consuming application (called a “Consumer”).
Email and SMS notifications may be received by users on their smartphones, whether or not Campus Mobile is installed. Push notifications are sent to a user’s device only when they have the Campus Mobile app installed and associated with their user ID through login. Alert and worklist notifications are not displayed in Campus Mobile.

For more information about the Notifications Framework, refer to the Campus Community PeopleBook chapter *Working With the Notifications Framework*.

### 2.5.2 Consumers

Campus Solutions is delivered with a number of preconfigured Consumers, some of which have been developed concurrently with Campus Mobile. These are identified with “Campus Mobile” in the name:

- Campus Mobile
- Campus Mobile Grade EMAIL
- Campus Mobile Grade PUSH
- Campus Mobile Grade SMS
- Campus Mobile WL Confirmation EMAIL
- Campus Mobile WL Confirmation PUSH
- Campus Mobile WL Confirmation SMS

All Consumers are delivered as inactive. To enable these Consumers, refer to the Campus Self Service PeopleBook chapter *Setting Up Campus Mobile*, “Setting Up For Notifications.”

You are able to modify the delivered Consumers or write your own.

It is important to understand that email and SMS notifications do not require implementation of Campus Mobile, whereas push notifications do. Conversely, you are not required to enable these Consumers in order to implement Campus Mobile.

#### 2.5.2.1 Final Grade Posting Consumers

The “Final Grade Posting” Consumers notify a student when the student’s final grade has been posted for a class.

Each student will receive a notification for every instance of having final grades posted for a class.

#### 2.5.2.2 Moved from Waitlist to Enrolled Consumers

The “Moved from Waitlist to Enrolled” Consumers notify a student when the student has been automatically enrolled into a class from the wait list.

The “Moved from Waitlist to Enrolled” Consumers rely on being triggered by a waitlist job that is delivered as inactive. For instructions on how to activate this waitlist job, please refer to the Student Records PeopleBook chapter *Using Enrollment Related Processes* for more information.

#### 2.5.2.3 Campus Mobile Consumer

The “Campus Mobile” Consumer notifies a student when:
• A new checklist has been assigned to the student, or if any checklist items on an existing checklist have been updated. (Note: This does not include the addition of a new checklist item to an existing checklist.)
• A negative service indicator has been assigned to, or removed from, a student’s record or an existing negative service indicator has been updated.
• The student’s enrollment appointment is about to open.

The “Campus Mobile” Consumer makes use of the Notification Framework’s batch processing capability. For information on how to configure the batch processing jobs, refer to the Campus Self Service PeopleBook chapter Setting Up Campus Mobile, “Defining and Scheduling Campus Mobile Batch Notifications.”

Checklist notifications are sent for checklists that are added or updated after the last batch execution date and prior to the date of the batch process execution.

Service indicator notifications are sent for changes occurring since the time of the last batch process execution.

Enrollment appointment notifications are sent for appointments that will open the day after the batch process execution.

It is expected that batch processing will occur once per day, with the following implications:

1. Executing the batch process more than once per day may lead to duplicate enrollment appointment or checklist notifications.
2. Executing the batch process less than once per day may lead to missed enrollment appointment notifications, as well as delayed checklist and service indicator notifications.
3. Notifications are sent to students at the time of batch process execution.

2.5.3 TEMPLATES

Every notification sent by a Consumer is based on a “Template,” which is used to create the notification. The template defines attributes of the notification, such as the message that is delivered to the user using email or SMS.

The following templates are delivered for use by the Consumers mentioned above:

- CHECKLIST_EMAIL
- CHECKLIST_SMS
- CHECKLIST_UPDATED_PUSH
- ENRL_FRM_WL_EMAIL
- ENRL_FRM_WL_SMS
- ENROLLED_FROM_WAITLIST_PUSH
- ENROLLMENT_APPT_EMAIL_OPEN
- ENROLLMENT_APPT_OPEN_PUSH
- ENROLLMENT_APPT_SMS_OPEN
• FINAL_GRADES_POSTED_PUSH
• FINAL_GRADES_CHANGED_PUSH
• FINAL_GRADES_REMOVED_PUSH
• GRD_EMAIL_ADD
• GRD_EMAIL_CHNG
• GRD_EMAIL_REM
• GRD_SMS_ADD
• GRD_SMS_CHNG
• GRD_SMS_REM
• SERVICE_INDICATOR_ASSGND_PUSH
• SERVICE_INDICATOR_REMOVED_PUSH
• SI_ADD_OR_CHG_EMAIL
• SI_ADD_OR_CHG_SMS
• SI_DELETE_EMAIL
• SI_DELETE_SMS

All Templates are delivered as inactive. To enable these Templates, refer to the Campus Self Service PeopleBook chapter Setting Up Campus Mobile, “Setting Up For Notifications.”

You are able to modify the delivered Templates or create your own.

2.5.4 PREFERENCES

Every user must specify an email address for receiving email-based notifications, and a cell phone number for receiving SMS-based notifications. This can be done through Campus Solutions Self Service > Personal Information > Notifications Preferences.

This page also allows users to enable or disable either email- or SMS-based notifications.

Campus Solutions Self Service also provides a page for administrators to configure the notification preferences for all users through Self Service > Campus Community > Personal Information > Biographical > Personal Attributes > Notifications Preferences.

Use of these pages is described in the Campus Community PeopleBook chapter Working With the Notifications Framework.

Since users will not receive email or SMS notifications until such information has been entered, you may consider implementing a batch update process, or communicating to all users that they should enter their notification preferences.

Users manage the receipt of push notifications within their smartphone’s operating system.

2.6 MESSAGE CATALOG CONSIDERATIONS

Campus Mobile uses the EWS for enrollment transactions and the EWS uses Message Catalog 14641 (refer to the EWS guide) for messages that are displayed to users. Most of the delivered messages
are applicable to any user interface and thus are suitable for Campus Mobile. However there are certain messages that should be evaluated by each institution to determine their applicability to Campus Mobile, since they refer to Campus Self Service UI components. Since these messages are currently utilized by both Campus Self Service and Campus Mobile, institutions may want to consider modifying the existing messages toward a more generic wording that does not include instructions specific to one delivery platform. Refer to the table below.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>14641 – 7</td>
<td>Class %1 is full. If a wait list is available, click Add Another Class to return to step 1. Click the class link, select the wait list option and resubmit your request.</td>
</tr>
<tr>
<td>14641 – 38</td>
<td>%1. You must obtain permission to take this class. If you have a permission number, click Add Another Class, click the class link, enter the number and resubmit.</td>
</tr>
<tr>
<td>14641 – 223</td>
<td>Class %1 and its alternate (%2) are full. If a wait list is available, click Add Another Class to return to step 1. Click the class link, select the wait list option and resubmit your request.</td>
</tr>
</tbody>
</table>
3. CAMPUS MOBILE

3.1 OVERVIEW

The features of Campus Mobile are described in the Campus Self Service PeopleBook chapter *Using Campus Mobile*.

Campus Mobile is delivered as an unsigned Mobile Application Archive (.maa) file, which gives you the ability to:

- Configure your mobile application. (refer to section 3.3.)
- Brand your mobile application. (refer to section 3.4.)
- Customize your mobile application. (refer to section 3.5.)
- Extend your mobile application. (refer to section 3.6.)
- Manage your mobile application. (refer to section 3.7.)

To do this, you will need to use Oracle’s JDeveloper IDE, which is described in section 3.2.

3.2 USING JDEVELOPER

Oracle JDeveloper is an integrated development environment (IDE) that simplifies the development of Java-based enterprise or mobile applications. JDeveloper is the development environment for the Oracle Application Development Framework (Oracle ADF). JDeveloper 11g R2 (11.1.2.4.0) provides support for developing ADF Mobile-based applications.

The ADF Oracle® Fusion Middleware Mobile Developer's Guide for Oracle Application Development Framework 11g Release 2 (11.1.2.4.0), herein referred to as the *ADF Mobile Developer's Guide*, provides a comprehensive guide to developing Oracle ADF-based mobile applications using Oracle JDeveloper. This document can be located on the Oracle Technology Network > Developer Tools > ADF Mobile.

The subsequent sections of this document provide information about how to configure, brand and customize the delivered Campus Mobile application archive using JDeveloper, making reference to applicable sections within the ADF Mobile Developer’s Guide.

3.2.1 INSTALLING AND SETTING UP JDEVELOPER

Navigate to [http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html](http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html), then download and install Oracle JDeveloper 11g R2 (11.1.2.4.0). For additional JDeveloper and ADF Mobile release level requirements, as well as other requirements specific to a Campus Mobile release version, please refer to My Oracle Support document 1557671.1.

Follow chapter 3 of the ADF Mobile Developer's Guide to set up JDeveloper and the required development tools for iOS or Android (or both).
3.2.2 Creating a New Mobile Application From the Delivered Mobile Application Archive File

In JDeveloper, select New Application > Mobile Application from Archive File, then browse to the CampusMobile.maa file contained within your Campus Solutions release folder. You will now see an application called CampusMobile in your Application Navigator, similar to the screen shot below.

![Application Navigator](image)

Figure 3-1 Application Navigator

Chapter 4 of the ADF Mobile Developer's Guide provides an overview of the structure of the mobile application that you have just created, including a description of the important files and how to navigate the application structure within JDeveloper. Bear in mind that the description in chapter 4 of the ADF Mobile Developer's Guide is related to an application that is created from scratch. For further information about what happens when you create an application from a mobile application archive file, refer to section 17.7.2 of the ADF Mobile Developer's Guide.

As delivered, the Campus Mobile application has Oracle branding and some dummy configurations. The subsequent chapters of this guide describe how you can configure, brand, customize and even extend your mobile application.

You may want to deploy and test your mobile application at this point before implementing any branding or customizations. Before doing so, you must at a minimum configure the web service endpoints and ensure that your security model is reflected within the app configuration (refer to section 3.3). Having done this, you can deploy to an installed emulator or to a connected device (refer to section 3.7.2).

3.3 Configuring Your Mobile Application

3.3.1 Web Services

All web service end points are defined in the Application Resources > Descriptors > ADF META-INF > connections.xml file. Within this file, you must replace each instance of the string “hostname” with the correct path to the server that hosts your web services, such as “myserver.myinstitution.edu”.

Oracle Corporation Confidential - For Oracle internal use only
To ensure secure communications to and from the mobile device, it is important to use secured web services (i.e. https and not http) that have an SSL certificate signed by a certified CA (since a self-signed certificate will not work).

Having configured your mobile application this way, your mobile application will connect to the web service end points that you have defined. If for any reason one or more of these end points change, you will need to release a new version of your mobile application.

3.3.2 Security
Chapter 18 of the ADF Mobile Developer's Guide provides an overview of the security framework within ADF Mobile and describes how to configure the security mechanisms within an ADF Mobile application.

3.3.2.1 Login Page
Campus Mobile is delivered to you with a custom login page, described by the file ApplicationController > Web Content > login.html. It is a simple HTML page that provides username and password text entry boxes, a Login button and a Remember me checkbox.

For information about customizing the delivered login page, refer to section 3.5.1.

3.3.2.2 Authentication
Campus Mobile authenticates against the delivered authentication REST web service that is described in section 2.3.1. The location of this web service is defined in the connections.xml file, within the node LoginConnection, as elements <login> and <logout>.

You must define the correct web service end point in these elements.

Following successful authentication, the user's credentials are stored in the encrypted on-device credentials store. Campus Mobile injects the user's credentials into the header of subsequent web service calls.

3.3.2.3 Authorization
Most features within Campus Mobile are constrained for access by the Permission List ‘HCCPCSSA3000’. This is defined in the View Controller > Application Sources > META-INF > adfmf-feature.xml file. If you want to make use of a different Permission List, perhaps even using different permission lists to constrain access to different features, change the value for each feature in this file.

Campus Mobile calls the delivered authorization REST web service that is described in section 2.3.2. The location of this web service is defined in the connections.xml file, within the node LoginConnection, as element <accessControl>.

You must define the correct web service end point in this element.

3.3.2.4 Timeouts
ADF Mobile provides for both an Idle Timeout and a Session Timeout, as defined within the <idleTimeout> and <sessionTimeout> elements, respectively, within the LoginConnection node of connections.xml.

The Idle Timeout defines the time for an application feature to remain idle before the user must re-authenticate. After this period expires, the user is timed-out of all the application features that are secured by the login connection. In this situation, the user is prompted with the login page when they access the feature again. If the Session Timeout has not yet occurred, the user's credentials are authenticated against the local credential store (otherwise the user's credentials will be authenticated against the authentication service). The default value of the Idle Timeout is 30 minutes.

The Session Timeout defines the time that a user can remain logged into an application feature. After this period expires, the user is prompted with the login page and the user's credentials are authenticated against the authentication service. The default value is 8 hours.

3.3.3 Properties

There is a single properties file (CampusSettings.properties) that holds configuration properties used within the app:

- GCM_SENDER_ID – To make use of push notifications to Android devices, you must create a Google API project (refer to http://developer.android.com/google/gcm/gs.html). You must specify the project number here, which will be used to validate the GCM sender ID during the device registration process for GCM.
- HOLDS_SERVICE_DURATION – To avoid loading the user's hold (negative service indicator) data every time the user views the springboard, the data is loaded in a background thread that is called at regular intervals. This value specifies the number of minutes in between each reload of the user's hold data. The default value is 5 (minutes).

3.3.4 Currency Symbols

Campus Solutions sends monetary amounts to Campus Mobile with an associated currency code, such as “USD” or “AUD.” Campus Mobile converts the currency code into a current symbol based on what is defined in the resource bundle (ViewControllerBundle.xlf). You should ensure that all currencies you expect to be displayed in your app are configured in the resource bundle.

3.4 Branding Your Mobile Application

Before distributing the Campus Mobile to your students, you will want to brand it as your own application, considering attributes such as:

- The application id, name and icon
- Your company or institution logo and color scheme
- Information about the app
- App URL scheme
• Versioning of the app.

These changes are made within JDeveloper and are described in the sections below.

3.4.1 Changing the Application ID

Both Android and iOS development guidelines require that every app has an identifier (called “package” for Android apps and “Bundle ID” for iOS apps) that is unique across all applications in the Apple App Store, or an application marketplace, as appropriate. The restrictions on this identifier are similar, but slightly different, for the two operating systems. This identifier is never displayed to your end users and you can set it for each operating system within the JDeveloper deployment descriptors, using the Application Bundle Id attribute. Refer to section 3.7.2 for more information about deployment descriptors.

However, since the naming conventions are similar, you may prefer to use the same unique identifier for both Android and iOS. This can be achieved by setting the Id attribute in the adfmf-application.xml file. At deployment time, if you have not entered a specific value for Application Bundle Id within a deployment descriptor, this attribute will be read from adfmf-application.xml.

Campus Mobile is delivered with the Id attribute set to “com.oraclecorp.internal.CampusMobile”. You should change this to something like “edu.myinstitution.MyApp,” which must match the provisioning profile issued by Apple (for iPhone app development).

By default, the name of the delivered package file (for example, MyApp.apk or MyApp.ipa) will be derived from the Id attribute specified in adfmf-application.xml. You can override this in the deployment descriptors.

3.4.2 Changing the Application Name

The configuration file adfmf-application.xml holds the value for the name of the application that will be displayed to users in the Apple App Store (or an application marketplace) and on the users’ devices.

As delivered, the application name is read from the resource bundle CampusMobileBundle.xlf, in which it is set to “Campus”. To use your own name, change the value of APP_NAME within this file to the name you want to use.

The reason for defining the application name in the resource bundle is to allow for localization of the name. You can learn more about localization in section 3.5.5.

3.4.3 Changing the Application Icon

Both Android and iOS have different requirements for the app icon. You should read the guidelines (Android Developers > Design > Iconography and iOS Developer Library > Custom Icon and Image Creation Guidelines > App Icon) before developing your own app icon. Both operating systems
require the definition of icon image files for different form factors, which will be scaled accordingly on user devices.

The delivered Campus Mobile app icon has two versions, one for Android and one for iOS, both 114x114 pixels. Whilst ADF Mobile facilitates the definition of different sized image files for different device form factors, the same 114x114 image file has been replicated with different filenames for each operating system. These image files are located android and ios folders under Application Resources > Resources. Once you have developed your own app icon, you can replace the delivered image files with your own, using the same file names.

Be aware that Apple also requires a 512x512 image file for use within iTunes.

Note that ADF Mobile allows for the iOS gloss effect on the icon to be turned on or off. This can be configured within the iOS deployment profile under Application > Application Properties> Deployment > IOS_MOBILE_NATIVE_archive > Edit > iOS Options > Application Images. As delivered, it is turned off.

### 3.4.4 Inserting Your Logo

As delivered, the Oracle logo appears on the splash screen and the ‘about’ page.

The splash screen is one whole image, for which there exists multiple image files for different device form factors. These image files are located in the same folders as the app icons, under Application Resources > Resources. Once you have developed your own splash screen image, you can replace the delivered image files with your own.

The logo used on the ‘about’ page is an image file (logo.png) stored within the ViewController> Web Content > resources > images. You can replace the existing file with your own or relocate it on the page using the .spacerAbout entry in the custom.css style sheet.

A small version of the Oracle logo is used as the Home button that, when tapped, returns the user to the springboard. This is stored as an image file (home.png within ViewController > Web Content > resources > icons) which you can replace with your own. You may want to use a small version of your own logo to ensure a consistent branding throughout the app. Otherwise you may want to use a generic home icon.

### 3.4.5 Inserting Your Color Scheme

The overall visual color scheme presented throughout the app is determined by the skin referenced within adfmf-config.xml and a few background images. The use of Cascading Style Sheet (CSS) skins ensures that all application components share a consistent look and feel.

The delivered skin ("campusMobile") is an extension of the default skin provided by ADF Mobile, which includes a predominantly black page background, with black header and footer, along with standard iOS coloring for page components, such as tables, lists and buttons.
The skin extensions are defined in `common.css` and `custom.css`, which are located in `ViewController > Web Content > resources > styles`. They are registered in the `adfmf-skins.xml` file.

You may want to change the skin of the app so that it uses your institution’s colors and works well together with your logo. To do this, you can either extend the delivered skin or insert your own skin. Chapter 5.11 of the ADF Mobile Developer's Guide describes how to do this.

Note that the login page (`login.html`) is displayed prior to the initialization of the ADF Mobile framework and is therefore not affected by the skin. If you change the skin of the app, you should also consider changing the appearance of the login page.

The springboard and ‘about’ pages make use of background images that are different to the one defined in the skin. Portrait and landscape versions of these image files are located in `ViewController > Web Content > resources > images`. You can replace the existing files with your own. On each of the springboard and ‘about’ pages stretch the background image to fit the user's device. They also define the location of the logo on the page. You can modify these pages (`springboard.amx` and `about.amx` within `ViewController > Web Content > oracle.cs.cm.view.springboard`) if your logo would be better suited to a different location on the new background image.

### 3.4.6 Inserting Your App Info

As delivered, the ‘about’ page displays a background image, a logo, the app name and version number, an Oracle copyright notice, and build number (in fine print). You should change the copyright notice to your own message by modifying the value for `COPYRIGHT_LBL` with the resource bundle `ViewControllerBundle.xlf`. You may want to remove the build number, which denotes the underlying Campus Mobile build.

If you want to make more extensive changes to the ‘about’ page, you could modify the delivered file (`about.amx` in `ViewController > Web Content > oracle.cs.cm.view.springboard`), or you could write your own page. This could be a convenient place to display any end user license agreement. If you write your own, you will need to update the reference within `springboard-task-flow.xml` in the same location.

### 3.4.7 Inserting Your App URL Scheme

With the release of v1.1 patch 4, ADF Mobile now supports the launch of an ADF Mobile-based app using a URL scheme. This means that your app can be launched from another app using a URL. To facilitate this, you must define the URL scheme to be used within the `admf-application.xml` configuration file. More information can be found in section 5.4 of the *ADF Mobile Developer’s Guide*.

### 3.4.8 Maintaining The App Version
The app version is defined in adfmf-application.xml. As delivered, this will indicate the version of Campus Mobile, but since this version is displayed to your users, you should maintain your own app version.

Although you can enter any text in this field, since iOS only allows the app version to contain one or more period-separated integers (for example, you can specify “10.2.5,” but not “V1”), you should follow the iOS-approved format to ensure compatibility across both Android and iOS.

If you want to use a different app version on Android, you can configure the Version Name attribute of the Android deployment descriptor separately. Note that the Android deployment descriptor also contains a Version Code attribute, which is not displayed to users and is used to determine whether one version of your app is more recent than another.

Note that the Description and Vendor fields in adfmf-application.xml can be left empty, as they are not used.

3.5 Customizing Your Mobile Application

3.5.1 Customizing the Login Page

As delivered, Campus Mobile contains a login page, login.html, located in ApplicationController > Web Content. You may want to modify the look and feel of this page to include your logo or additional information for your users. To do this, you should create a new HTML page (perhaps a copy of the delivered one), which should be referenced as the custom login page within adfmf-application.xml.

3.5.2 Customizing the Springboard and Navigation Bar

As delivered, Campus Mobile contains a springboard and a navigation bar.

ADF Mobile provides you with the ability to customize the functionality of the springboard and navigation bar using JDeveloper. For example, you could –

- Turn off the springboard or navigation bar
- Change the way the springboard is displayed
- Create your own springboard
- Display a toggle button that enables the user to display the springboard or navigation bar

Chapter 5.4 of the ADF Mobile Developer’s Guide describes how to make such customizations.

3.5.3 Customizing the App Features

Every ADF Mobile app contains one or more features. A feature may be accessed directly from the springboard or navigation bar, or it may be accessed from another feature.

As delivered, Campus Mobile contains the following features:

- Schedule
• Search
• Cart
• Grades
• Profile
• Finances
• To do List
• Springboard (used to provide a custom springboard implementation)
• Redirector (used to display the springboard if the user shakes the phone)
• Map (used to provide a custom map implementation)

The first seven in the list are displayed on the springboard and navigation bar. The others are not.

ADF Mobile provides you with the ability to customize the display and functionality of the delivered features using JDeveloper. For example, you could –

• Remove features from (or add to) the springboard or navigation bar
• Change the order of the features displayed on the springboard and navigation bar
• Change the names and icons of features (refer to section 3.5.5)
• Change feature security and constraints
• Change feature lifecycle and task flows

Chapter 5.5 of the ADF Mobile Developer's Guide describes how to make such customizations.

Note that the delivered functionality of Campus Mobile is defined by the features, so changing attributes such as security, constraints, lifecycle or task flows is not recommended.

3.5.4 Maps Integration

The delivered Campus Mobile application has a Map feature that enables users to display the location of a classroom on a map if the classroom's building location has been entered into Campus Solutions.

ADF Mobile supports the use of Oracle Maps or Google Maps. As delivered, Campus Mobile uses Oracle Maps and supports the display of the classroom location and the user’s current location.

If you change to Google Maps, your users will be able to access Google Street View and additional layers, such as satellite. It would also be possible to extend the app to provide additional information, such as directions.

To configure the app to use Google Maps, you must modify adf-config.xml as follows:

1. Replace `<adf-property name="mapProvider" value="oraclemaps"/>` with `<adf-property name="mapProvider" value="googlemaps"/>`.
2. Add `<adf-property name="geoMapKey" value="MapKey"/>` where yourKey is your institution’s (or company's) Google Maps business license key.
3. Add `<adf-property name="geoMapClientId" value="ClientId"/>` where yourClientId is your institution's (or company's) client id for your Google Maps business license.


5. Remove `<adf-property name="baseMap" value="ELOCATION_MERCATOR.WORLD_MAP"/>`.

Refer to section 3.6.1 for details of how to extend the app to make use of additional Google Maps functionality.

### 3.5.5Icons, Labels, Resource Bundles and Localization

All icons and labels displayed within Campus Mobile are referenced within the resource bundle ViewController.xlf. This approach not only enables you to easily replace an icon or label with your own, but also facilitates localization. In this way, you can support more than one locale within the app. Refer to section 5.10 of the ADF Mobile Developer’s Guide for more information.

Campus Mobile determines the resource bundle to be used based on the locale specified in the device’s settings.

If you don’t need to support multiple locales within the app, you can simply make changes to labels directly within ViewController.xlf and you can replace the image files directly in the folder ViewController > Web Content > resources > icons.

You should read the guidelines (Android Developers > Design > Iconography and iOS Developer Library > Custom Icon and Image Creation Guidelines) before developing the icon set to be used in your mobile application.

### 3.5.6Customizing the Generic Error Message

If Campus Mobile receives an invalid web service response (or no response within the timeout period), a generic error message is displayed to the user. As delivered, this error message states “Connection Error. Please try again. If the problem persists, please contact your IT helpdesk.” You may want to customize this message by including a phone number or email address that they can contact in such situations. To do this, modify the value for the key `CONNECTION_ERROR_LABEL` in ViewController.xlf.

### 3.5.7Customizing Pages Within the App

ADF Mobile AMX is a sub-framework within ADF Mobile that provides a set of UI components that enable you to create an application feature whose behavior is identical on all supported platforms and provides access to device functionality. JDeveloper allows you to drag these components into an editor from the Component palette or from the Data Control palette. JDeveloper also allows you to create task flows that define the navigation between ADF Mobile AMX pages.
Each page within Campus Mobile is defined by an AMX file, such as grades.amx for the Grades feature. Using JDeveloper, you have a lot of flexibility for changing the layout of any page within the app.

Before making such changes, please read sections 6, 7 and 8 of the ADF Mobile Developer’s Guide to ensure that you have a thorough understanding of AMX.

For information about changing the data displayed on a page within the app, refer to section 3.6.2.

It is not recommended to change the task flows within the app, other than to update the page names if you customize pages. Otherwise you may break the designed flow of the app, potentially introducing bugs and even leading to data integrity issues.

3.5.8 Customizing the Android App Permissions

On iOS, when the app attempts to access a device service (such as a camera or contacts), the user is prompted to provide permission.

On Android, the required device access permissions for the application cannot be automatically determined. The user is prompted during installation to provide permission to access the required device services. All required permissions are listed in a file (AndroidManifest.xml) that is delivered within the JDeveloper ADF Mobile Framework extension and copied into the Android app when JDeveloper builds it.

To modify this list of permissions, follow these steps:

2. Edit Android/template/AndroidManifestTemplate.xml, and remove the permissions that are not required by the application.
3. Zip it back up, ensuring the zip file has the exact same directory structure as the original zip file. Any extra layers of directory in the zip file will result in failure.
4. Restart JDeveloper and deploy your app.

3.6 Extending Your Mobile Application

3.6.1 Providing Additional Map Functionality

If you change to using Google Maps (refer to section 3.5.4) and want to make use of other Google Maps API features, such as providing directions, you must write your own JavaScript and incorporate it into the show-location.amx page (refer to section 3.5.7 regarding customizations to delivered AMX pages).

3.6.2 Changing the Displayed Data

In addition to changing the layout of a delivered AMX page (refer to section 3.5.7 regarding customizations to delivered AMX pages), you may want to alter the data that is displayed on a delivered AMX page.
To remove data, simply remove the component that displays the data. This may be useful if you do not make use of a particular field during a transaction and therefore you don’t want to confuse your users by displaying the field.

Within the AMX page definition, you have flexibility about how data is displayed. For example, you may want to display GPA to only 1 decimal place instead of 3.

In order to display additional data on a page, you must first ensure that the data is delivered to the app. This may be as simple as altering the delivered entity profile for the relevant web service (refer to the Entity Registry chapter(s) of the Campus Community PeopleBook). Or you may need to customize the web service.

To ensure that the data in the web service response is made available to the relevant AMX page(s) for display, you need to write a data control. The delivered Data Controls cannot be modified or extended. For more information about web service data controls, refer to chapters 8 and 9 of the ADF Mobile Developer’s Guide.

Once you have created the web service data control, you can then update your AMX page to display the additional data.

3.6.3 Providing Additional Features

One of the benefits of using Oracle ADF Mobile technology is the ease with which you can extend Campus Mobile by adding new features. As described in section 1.3, each of AMX, Local HTML and Remote HTML features may coexist within an ADF Mobile application, which means that you can leverage existing web pages and web services without having to learn ADF Mobile AMX.

For example, if you already have a web page that has been designed for the mobile browser (or even if it hasn’t), you can very easily add a Remote HTML feature that displays the web page within Campus Mobile. For more information on this, read chapter 12 of the ADF Mobile Developer’s Guide.

Alternatively, if you have existing web services, you can easily add a Local HTML feature, writing your own HTML code (in JDeveloper or not) that calls the web services. Such pages may still access the device functionality. For more information refer to section 5.9.1 of the ADF Mobile Developer’s Guide.

3.7 Managing Your Mobile Application

3.7.1 Developing Your Mobile Application

Before making any changes to the delivered app, you should consider formalizing your development processes. In particular, you should maintain a record of all the changes so that they can be reapplied when you upgrade to later revisions of Campus Mobile.

A typical development lifecycle includes:
• Requirements gathering and definition
• Design
• Implementation (refer to sections 3.2 to 3.6)
• Deployment (refer to section 3.7.2)
• Testing and debugging (refer to section 3.7.3)
• Distribution (refer to section 3.7.4)
• Support (refer to section 3.7.5)

3.7.2 **Deploying Your Mobile Application**

Chapter 17 of the ADF Mobile Developer’s Guide describes in detail the entire deployment process.

The delivered Campus Mobile application has a default Android deployment profile and an iOS default deployment profile. You can make use of these for your testing and also for the eventual creation of mobile application files for distribution to your students. Otherwise you can modify these or create new deployment profiles to deploy with different attributes.

The delivered deployment profiles are configured to deploy the app in Release mode and in the case of iOS, as an iPhone app (which means that the app can be deployed to an iPad but will display as iPhone size). Note that changing to Debug mode will cause your mobile application to run much slower. Changing to an iPhone/iPad app will cause the app to stretch the UI if deployed to an iPad (for which it is not designed). Of course, changing to an iPad app will preclude deployment of the app to an iPhone for your users.

The delivered deployment profiles are unsigned. You must configure the deployment profiles with your signing properties. For Android, refer to section 17.2.3.4 of the ADF Mobile Developer’s Guide. For iOS, refer to sections 17.2.4.2 and 17.4.4 of the ADF Mobile Developer’s Guide.

To enable your app to register for push notifications on iOS devices, you must ensure that your production provisioning profile includes the entitlement specific to push notifications. The same is true for your testing provisioning profile, if you want to test push notifications. Note that this can only be done with a device, not using the iOS simulator.

To enable your app to register for push notifications on Android devices, you must create a Google API project and record the project number in the CampusSettings.properties configuration file, as described in section 3.3.3.

To deploy your mobile application for testing, select Application > Deploy, then select either the Android or iOS deployment profile. You must then select to deploy to a connected device or an installed emulator. Once the deployment has finished, you will be able to test your mobile application.

Note that the first launch of the app on Android may take a while as it unpacks the app.

3.7.3 **Testing and Debugging Your Mobile Application**
More information about testing and debugging your mobile application can be found in chapter 19 of the ADF Mobile Developer's Guide.

3.7.3.1 Logging
As delivered, Campus Mobile only logs ‘SEVERE’ messages. This is recommended for your production release of your mobile application. However, for debugging purposes, you can lower the message severity for logging. To do this, modify the value for .level in the file Application Resources > Descriptors > META-INF > logging.properties. The accepted severity values are: SEVERE, WARNING, INFO, FINE, FINER, FINEST. For more information about configuration of logging, refer to section 19.4 in the ADF Mobile Developer’s Guide.

Log files are written to the following locations:

- Android Device: /sdcard
- Android Emulator: DDMS > /mnt/sdcard
- iOS Device: Application > Documents/logs
- iOS Simulator: Mac Console > Applications > Utilities > Console

3.7.3.2 Debugging
Debugging properties are located in cvm.properties. In particular, to enable Java debugging, the parameter java.debug.enabled must be set to true.

To enable remote debugging, the app should be deployed in Debug mode.

3.7.3.3 Certificates
JDeveloper creates a file within the Application Resources > Security folder that identifies a set of certificates from well-known and trusted sources. If the certificate used by your SSL web services is not contained within this set, or not signed by a CA within this set, you may notice an error ‘Certificate was issued by an unrecognized entity’ in the log file. Refer to section 18.5 of the “ADF Mobile Developer’s Guide” for information on how to add your certificate to the trusted set.

3.7.4 Distributing Your Mobile Application
Deploying the application to the production environment typically involves publishing to an enterprise server, the Apple App Store, or an application marketplace, such as Google Play. After you publish the ADF Mobile application, end users can download it to their mobile devices.

Refer to section 17.3.3 of the ADF Mobile Developer’s Guide for information about publishing an Android application. Refer to section 17.4.5 of the ADF Mobile Developer’s Guide for information about publishing an iOS application.

3.7.5 Supporting Your Mobile Application
You should consider the support and maintenance ramifications of delivering a mobile application to your students.
You may want to set up a support phone number or email, which could be referenced in the mobile application's 'about' page (refer to section 3.4.6) or generic error message (refer to section 3.5.6).

You may consider providing a mechanism for students to give feedback about the app, which could be used to inform additional development of your mobile application.